WARWICK
THE UNIVERSITY OF WARWICK

**A Thesis Submitted for the Degree of PhD at the University of Warwick**

**Permanent WRAP URL:**
http://wrap.warwick.ac.uk/78612

**Copyright and reuse:**
This thesis is made available online and is protected by original copyright.
Please scroll down to view the document itself.
Please refer to the repository record for this item for information to help you to cite it.
Our policy information is available from the repository home page.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk

**warwick.ac.uk/lib-publications**

# SCHEDULING AND CONTROL IN THE BATCH PROCESS INDUSTRY USING HYBRID KNOWLEDGE BASED SIMULATION

by

William Richard Goodall

Submitted for the Degree of Ph.D.

at

The University of Warwick

in

The Department of Engineering

September 1993

BRITISH LIBRARY

**IMAGING SERVICES NORTH**

Boston Spa, Wetherby
West Yorkshire, LS23 7BQ
www.bl.uk

# CONTAINS PULLOUTS

CONTENTS

ILLUSTRATIONS

ACKNOWLEDGEMENTS

Mr.R.Roy for his supervision and support of this work

Mr.T.Chadwick and the staff of Eden Vale Minsterley for their co-operation and help.

Mr.and Mrs.R.W.Goodall and my wife Annette for their support during the whole period of this work.

DECLARATION

No portion of the material in this thesis has been used before.

## SUMMARY

This thesis relates to the area of short term scheduling and control in batch process plants. A batch process plant consists of individual plant items linked by a pipe network through which product is routed. The structure of the network and the valve arrangements which control the routing severely constrains the availability of plant items for configuration in routes when a plant is operating. Current approaches to short term scheduling contain simplifying assumptions which ignore these constraints and this leads to unrealistic and infeasible schedules. The work undertaken investigates the use of techniques from the areas of Artificial Intelligence (AI) and Discrete Event Simulation (DES) in order to overcome these simplifying assumptions and develop good schedules which can be implemented in a plant.

The main divisions of work cover a number of areas. The development of a representation scheme for batch plant networks, and procedures for reasoning about the constraints imposed by their structure to infer the actual availability of plant items for routing purposes at any time. The development of a dynamic rule-based route configuration procedure which takes into account the constraints on plant item availability. The development of an activity scheduling framework for batch plants based on this. The development of a dynamic simulation model to take account of finite capacity constraints in a batch plant. The integration of these elements in a hybrid structure to make best use of the techniques available from the areas of AI and DES.

The representation scheme and procedures developed for reasoning about the constraints in a plant network enable the simplifying assumptions of other approaches to be overcome so that the system can produce good feasible schedules. The hybrid structure is a practical one to take for implementation and enables the best use of techniques from AI and DES.

# CHAPTER 1 INTRODUCTION

## 1.0.THE AREA OF RESEARCH

This research was initiated to investigate the use of hybrid systems based on procedural Discrete Event Simulation (DES) and declarative knowledge based programming techniques from the area of Artificial Intelligence (AI) for the purpose of short term scheduling and control in the batch process industry. The aim of the research was to demonstrate that the use of a hybrid approach could produce a tool for this that overcomes particular difficulties with respect to the constraints introduced through the typical network structure of batch process plants. This thesis is therefore written for a reader who has some familiarity with the batch process industry, and the principles and techniques used in DES and AI.

The rest of this chapter will briefly describe a number of areas as an introduction to the remainder of the thesis. These areas are:

1. The characteristics of the batch process industry and batch process plants which give rise to the particular scheduling problems which have been addressed.
2. Overall production planning and control of batch plants and the importance of short term scheduling and control within this.
3. The application of techniques from Operations Research, DES, and AI to short term scheduling and control.

## 1.1.THE SEMI-CONTINUOUS BATCH PROCESS INDUSTRY

This area of manufacturing encompasses industries such as dairy processing, pharmaceuticals, food, and brewing. It is generally characterised by the production of a diverse range of final products starting from a few basic raw

materials, often in liquid or powder form, which are processed in batches of varying size at one or more stages to produce the final product. In many cases the final product processing stage consists of filling and packing a bulk product into large numbers of discrete units such as packs of yoghurt. A processing stage could involve a chemical reaction which occurs in a batch reaction vessel, or it could involve moving the product through a plant item which carries out the process. This second type of processing may involve splitting of a product into one or more components or the blending of one or more component into a single product. Processing of this type tends to be intermittent in that one or more intermediate product batches will be processed through a plant item with gaps in between the batches. Therefore it is termed semi-continuous processing. Because of these characteristics short term scheduling and control effort is aimed at carrying out the production of intermediates at each stage in a timely manner to supply the requirements of all following stages in a process.

### 1.1.1.Process Plant Structure

The equipment which is in a batch process plant can be classified into a number of plant item types including storage and reaction vessels, and semi-continuous process plant items such as separators, heat exchangers, in line mixers, and evaporators. These are all linked by a complex network of pipes with product routing and movement controlled by pumps and a large number of valves. A simplified picture of the network structure in a batch plant is given by a Process Flow Diagram an example of which is shown in Figure 1.1. The network structure and possible open/ shut valve configurations will constrain the feasible routing between plant items. Also because different production activities which involve product routing often share some route components such as a semi-continuous process plant item or are controlled by a common set of valves this can constrain whether a potentially feasible route can be set up or not.

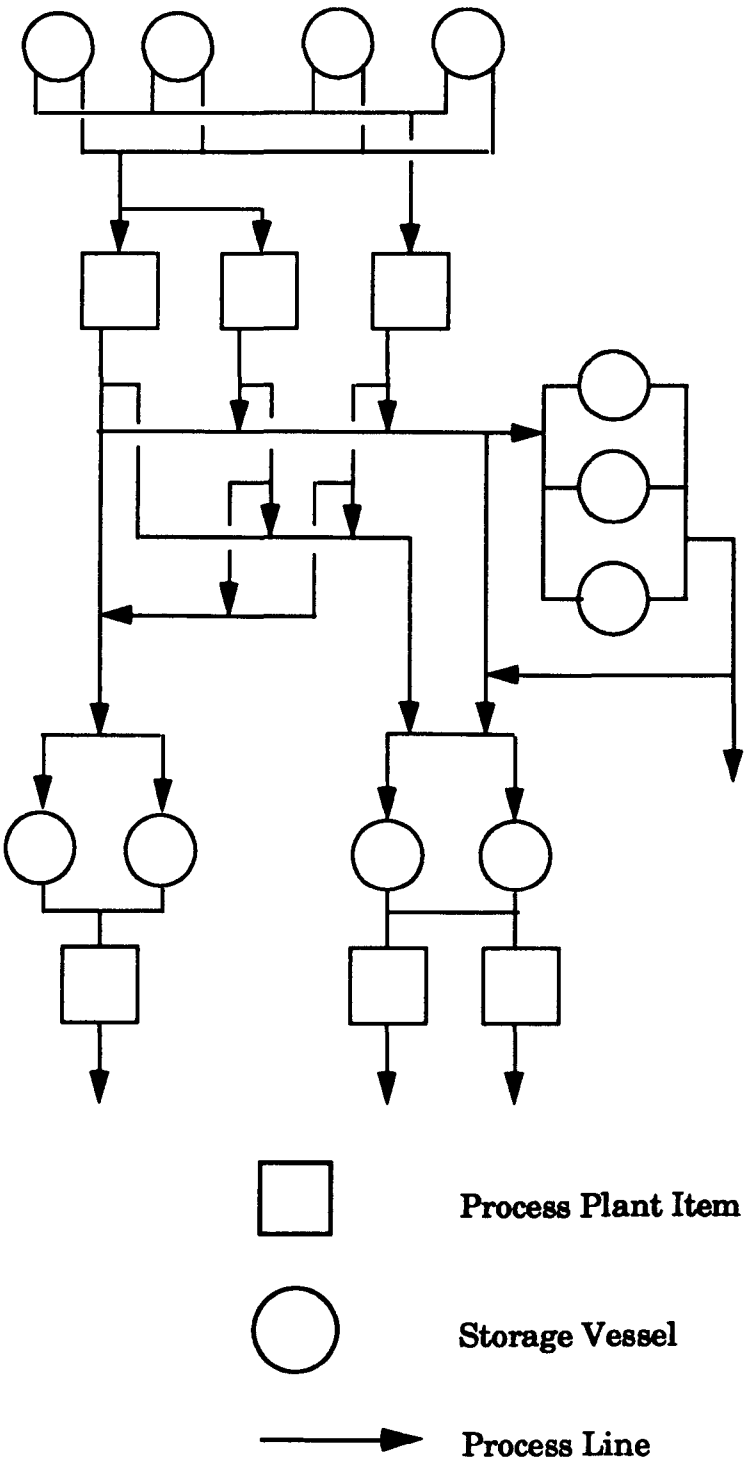



Figure 1.1. Example Process Flow Diagram

There is a high capital investment in the equipment and process control systems which means there is often a requirement for a large degree of operational flexibility and high utilisation from it even though individual product requirements may be small so that long production runs of a single product are not possible. In order to achieve high levels of utilisation the efficient scheduling and management of the production of batches within the plant is taking on progressively more importance as discussed by Simmons [1]. The development of techniques for scheduling and automation of the management function is therefore the subject of a lot of recent research. However, the effects of the plant network structure and the constraints on product routing have not so far been properly addressed in this context.

## 1.2.PRODUCTION PLANNING AND CONTROL

The production planning and control process can be split into an integrated hierarchy of functions which address different levels of detail and have a correspondingly suitable time horizon. In 1981 the American Production and Inventory Control Society (APICS) published a report containing a framework for such a hierarchy for the process industry [2]. This consisted of three levels of Resource Requirements Planning, Production Planning, and Production Scheduling. To this can be added two lower levels of Batch Management and Process Control to give the hierarchy as shown in Figure 1.2.

### Resource Requirements Planning

The APICS report described Resource Requirements Planning as concerned with long term decision making using long range forecasts based on aggregated information to develop '..acquisition plans for facilities, workforce, raw materials, energy and information.' The sort of time horizon covered by this planning function is usually measured in years. The results from this

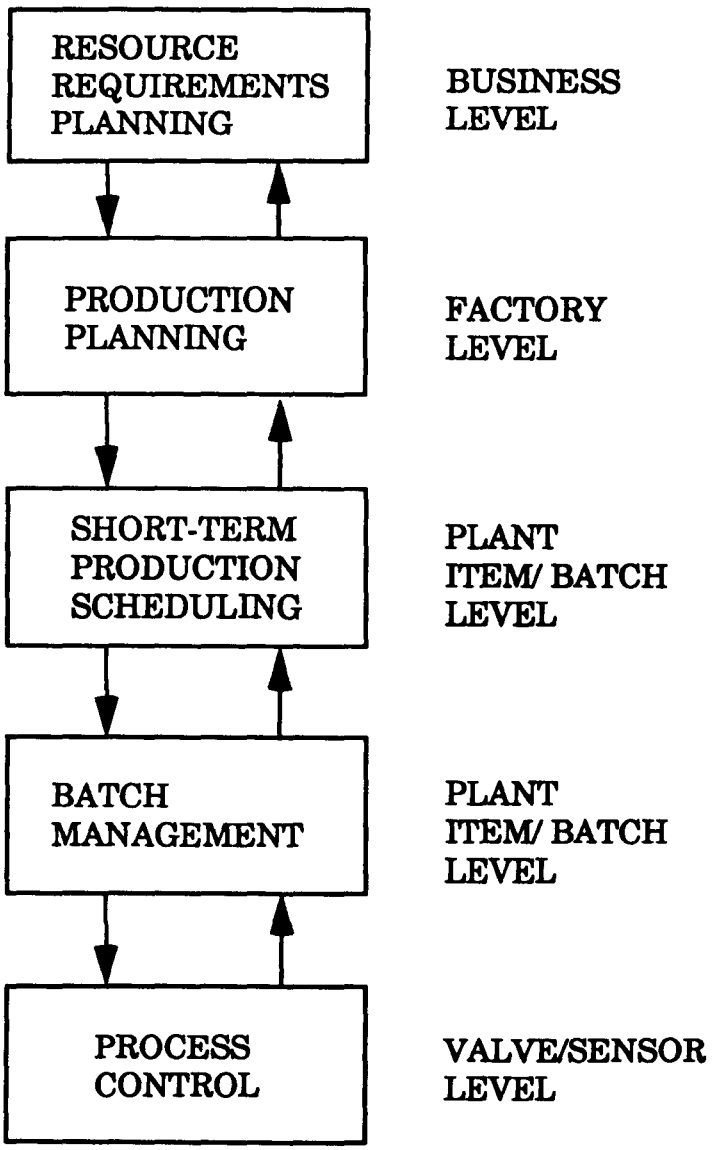



Figure 1.2. Production Planning and Control Hierarchy

level of planning can result in constraints on lower levels through the resources which are available to meet production demands.

## Production Planning

The Production Planning function is concerned with the development of '..intermediate range operating plans in response to demand forecasts so as to optimize the utilisation of resources.' Typical time horizons for this sort of planning are measured in months, and the outputs are operating plans associated with annual and quarterly operating budgets. The results of this sort of planning result in aggregate production levels and production campaigns to optimize factories and departmental output over the medium term planning horizon. This type of planning is often carried out using analytical techniques such as Linear Programming. For example Benseman [3] describes the use of a Linear Programming model to help a large New Zealand dairy company allocate available milk supplies to its various production sites. Mauderli and Rippin [4] describe a system called BATCHMAN to derive an optimal production plan of production campaigns for a batch plant to meet a particular demand pattern. A campaign is defined as the production of one or more products simultaneosly and refers to the whole plant. It uses a technique of enumerating alternative production campaigns which could be carried out in the plant coupled with a screening procedure to eliminate inefficient ones before using a Linear Programming procedure to assign campaigns to a production plan.

## Production Scheduling

Production scheduling is concerned with meeting specific production demands. This consists of a number of sub-modules according to the APICS report including finished product scheduling of packing of finished products, master production scheduling which '..disaggregates the production plan into a

production schedule for specific finished products, blend stocks, intermediate products or components to be produced in specific time periods at a particular manufacturing facility.', material plannning which is used to calculate when intermediate products should be manufactured and raw materials ordered, and material control which includes final scheduling of production at a finer level of detail to meet the requirements of material plan and the master production schedule. Short term scheduling for material control is carried out for a scheduling horizon ranging from a few shifts to a few days and involves the determination of when production activities should be carried out to meet product demands and how resources and materials should be allocated to them. Demand over these short time periods is usually fluctuating so the ability to meet demand may be constrained by plant capacity and raw material availablity fixed at a higher level based on an aggregate demand forecast. The actual production process and the relationships between production activities to be carried out for an individual product must be considered at this level. In determining how to allocate plant items to activities the suitability and availability of individual plant items must be considered in conjunction with both the static network structure of the plant and how the feasible connectivity of plant items changes as routes are set up in it. In addition the requirement to carry out cleaning activities between different processes should also be considered. This can be a severe constraint on production scheduling both in terms of the time available for scheduling and the availability of resources. Cleaning is typically carried out using a Cleaning In Place (CIP) system which is part of the plant network structure and will therefore have an effect on the availability of routing within the plant. Other factors which must be considered include operating policies related to product quality such as whether storage vessels can be filled and emptied at the same time or must be filled, emptied and then cleaned before re-use.

Batch Management

The implementation of the schedule must be managed which is the next level of the Production Planning and Control hierarchy and requires some kind of supervisory control of the plant. This is typically carried out in a completely manual fashion with a plant operator or supervisor deciding when to start production activities listed in the schedule and setting appropriate process parameters for the process control system to work to. However there has been recent work on the automation of this function. Cott and Machietto [5] describe work carried out to develop a batch management system for batch plants with a flow shop structure which is situated at a level between the production scheduling system and the process control system. In this case the batch management system monitors the status of production in the plant and determines when to start a production activity subject to rules about precedence between activities and the availability of resources. The actual rates or durations of a process when the schedule is implemented may well differ from those used at the scheduling stage. The system described by Cott and Machietto includes an on-line schedule algorithm called Projected Operation Modification Algorithm (POMA) which '..monitors the equipment allocations specified by the production plan and attempts to deal with processing time variablities by modifying the time element of the schedule.' In addition unforseen interruptions such as plant breakdown or delays in supply of raw material may occur, so that after a period of time the variance between the original schedule and the actual situation may be considerable. Therefore this stage must also include decision making about whether to simply adjust the current schedule or re-schedule the plant taking into account its new status, for example using procedures put forward by Bhattacharyya, Roy and Huang [6].

Process Control

Process control is the lowest level in the hierarchy operating over a short time horizon ranging from milliseconds to minutes. It is concerned with individual processes, for example the maintenance of the product temperature level through a heat exchanger, and also the co-ordination of several plant items together to carry out particular production activities scheduled at the scheduling level and initiated at the batch management level. For many years the process industries have accomplished this through plant automation. Automation was originally "hardwired" into banks of relays, and since the 1950s has been accomplished using mini and micro-computers, and programmable controllers, programmed with timed sequences controlling on/off activation of pumps and valves associated with particular product routes. Many sequences may be required to operate in parallel, with interlocks, either in hardware or software, between the plant items to ensure that they operate in synchronisation as described by Simmons [1].

### 1.2.1.The importance of Short Term Scheduling

It can be seen that, as described by Dempster et al. [7], carrying out the activities at each level of the hierarchy depends to an extent on the output from the previous level in the hierarchy which will impose some constraints on it. If the activities carried out at one level do not consider the practicalities of implementation at the next level then the hierarchy may not function very well. In the case of the production scheduling function it is particularly important that it fully takes into account the constraints that exist in a plant or it will not be possible to implement the schedule on-line and any effort put into the development of a batch management system will be wasted. The material control sub-module is the one addressed by the work described in this thesis because the related issues of plant representation and allocation have not been properly dealt with by other approaches documented in the

literature, and without doing this the remaining levels in the hierarchy of batch management and process control cannot function properly.

## 1.3.APPROACHES TO SHORT TERM SCHEDULING

There has been a considerable amount of research undertaken on sequencing and scheduling in both the batch process industry and manufacturing generally through the application of techniques from Operations Research (OR), AI and DES. OR approaches use analytical techniques which attempt to optimise the performance of a plant with respect to some criterion such as the minimisation of the flow time of batches through the plant. However, this type of goal is un-related to the real concern of production management which is to meet demand for products by their due dates while meeting quality requirements and minimising cost. Also, in order to make the scheduling problem fit the solution method analytical approaches use simplifying assumptions which make the schedules produced infeasible under real conditions. AI and DES based approaches are much more concerned with meeting real constraints and producing realistic feasible schedules. The strength of AI based approaches lies in the use of declarative programming languages and representation schemes which enables a system developer to concentrate on the representation of constraints, and the development of procedures to reason about the constraints and take them into account during the scheduling process. DES systems have efficient event scheduling mechanisms which can co-ordinate the activities of large numbers of entities over time. On this basis they can form part of an efficient forward scheduling tool as described by Roy [8]. However both approaches have weaknesses. The representational power of the procedural languages used for DES is not as great as that of declarative AI languages and AI based approaches tend to be less well developed than DES systems for carrying out the procedural time based part of scheduling.

## 1.4.OBJECTIVES OF THE RESEARCH

As stated earlier, unless the short term scheduling function in the Production Planning and Control hierarchy can properly take account of the constraints which are inherent in a batch process plant it will not be possible to carry out the batch management and operation of the plant properly. On this basis, the objectives of the work carried out and described in this thesis were:

1. To investigate what features are necessary in a short term scheduling tool to take account of these constraints and produce good feasible schedules.
2. The development of a system structure to implement a short term scheduling tool incorporating these features.

The remainder of the thesis is organised to review the areas of OR, DES and AI as they relate to the work which has been carried out, and then describe the representation scheme and procedures for short term scheduling which have been developed and implemented in a hybrid knowledge based system to meet these objectives.

# CHAPTER 2 ANALYTICAL APPROACHES TO BATCH PLANT SCHEDULING

## 2.0.INTRODUCTION

The purpose of this chapter is to briefly describe the research reported in Chemical Engineering literature on the use of Operations Research (OR) techniques to carry out short term scheduling and their applicability to the real requirements of batch plant operations. In a review of batch process operations scheduling Reklaitis [9] defines short term scheduling as '..the methodology according to which the order in which products are to be processed in each of the units of a plant is determined so as to optimize some suitable economic or systems performance criterion. Scheduling is always required whenever a processing system is used to produce multiple products by sharing the available production time between products.' Reklaitis [9] describes a number of factors which must be taken into account by a scheduling technique. 'The solution of the scheduling problem will be critically affected by the structure of the processing network, the processing times required for each product, the presence or absence of intermediate storage, the lost production time or cost associated with product changeovers, the cost or performance criterion used to rank schedules, as well as any due dates which are assigned to individual products.'

Because of the range of factors which affect the scheduling problem there is no general analytical method which can be applied to any particular plant configuration, product mix and performance criterion. Therefore, researchers classify problems and their solution method on the basis of the plant configurations which it can handle subject to specific assumptions made to further simplify the problem. Most of the approaches to the short term scheduling problem described in the Chemical Engineering literature originate from the area of OR on the scheduling of flow-shops and job-shops in

discrete manufacturing. A classification scheme for the representation of plant structure and problem structure, and solution procedures are typically adapted from this to the batch process industry as described by Reklaitis [9], Rippin [10], and Ku, Rajagopalan and Karimi [11]. The remainder of this chapter will describe the assumptions made in the classification scheme, the approaches which have been put forward, and their applicability to real scheduling situations.

## 2.1.CLASSIFICATION OF BATCH PROCESS PLANT STRUCTURE AND PROBLEM STRUCTURE

The classification scheme used for batch process plants is based on adapting the classifications used for job-shops and flow-shops as described by French [12]. According to French the assumptions made are often quite restrictive and inappropriate to real scheduling problems. Their purpose is to allow a mathematical theory of scheduling to be developed based on assumptions which might be made rather than to solve practically occuring problems. On this basis it is likely that a similar classification scheme will be innapropriate to real scheduling problems in batch process plants.

### 2.1.1.Plant Configuration

Under this scheme a batch plant can be described by its configuration and the classes of plant items in it. A configuration is described in terms of the stages through which a batch of product passes during its manufacture. The plant items which make up a particular stage are described as units. The configuration scheme used is as follows:

1.Single stage in which the product is manufactured in a single stage.
2.Multistage in which the product is manufactured in several stages.

The multistage classification is further subdivided as follows:

1. Multiproduct (Figure 2.1.) in which all products manufactured on the plant follow the same basic sequence of processing steps. This is typically equated to the flow shop from discrete manufacturing literature.
2. Multipurpose (Figure 2.2.) in which different products follow different paths through the plant. This is typically equated to the job-shop from discrete manufacturing literature.

A production stage can be divided into two subclasses; one with a single unit at each stage, and one with multiple parallel units at each stage. Multiple parallel unit stages can also be classified according to the properties of the units:

1. Identical units where all units at a given stage are the same in their properties.
2. Non-identical units where all units at a given stage are not the same in their properties.

This classification scheme represents a significant simplification of the situation in a real plant, because it does not attempt a proper representation of the network structure and feasible connections between plant items. It appears to have been taken primarily to allow the OR approaches for scheduling job-shops and flow-shops to be adopted based on perceived similarities with batch plants rather than to allow an independent theory of scheduling to be developed for batch plants based on their own specific configuration characteristics.

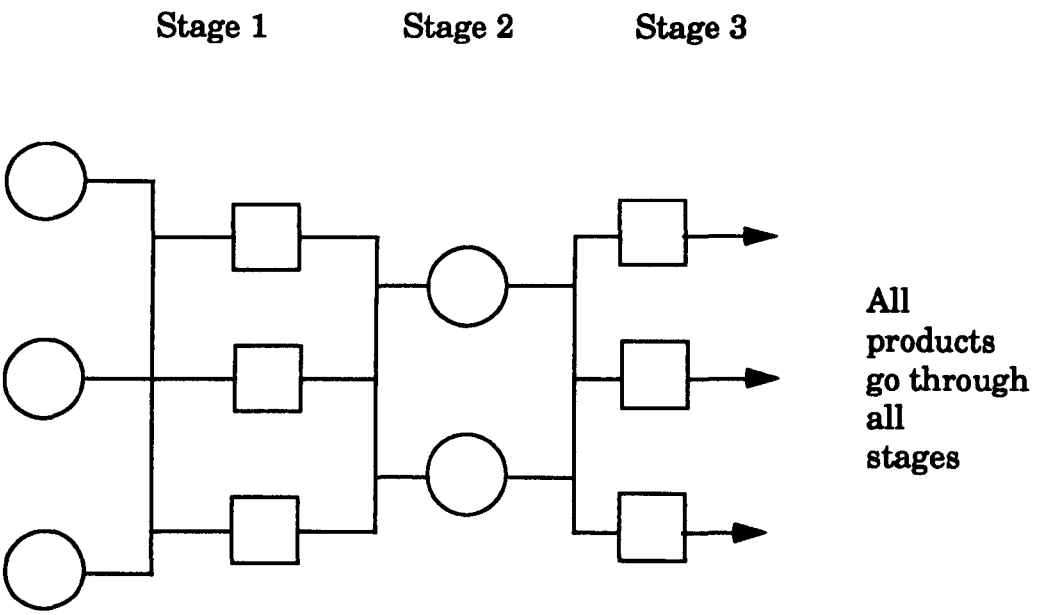



Figure 2.1. Multiproduct Configuration

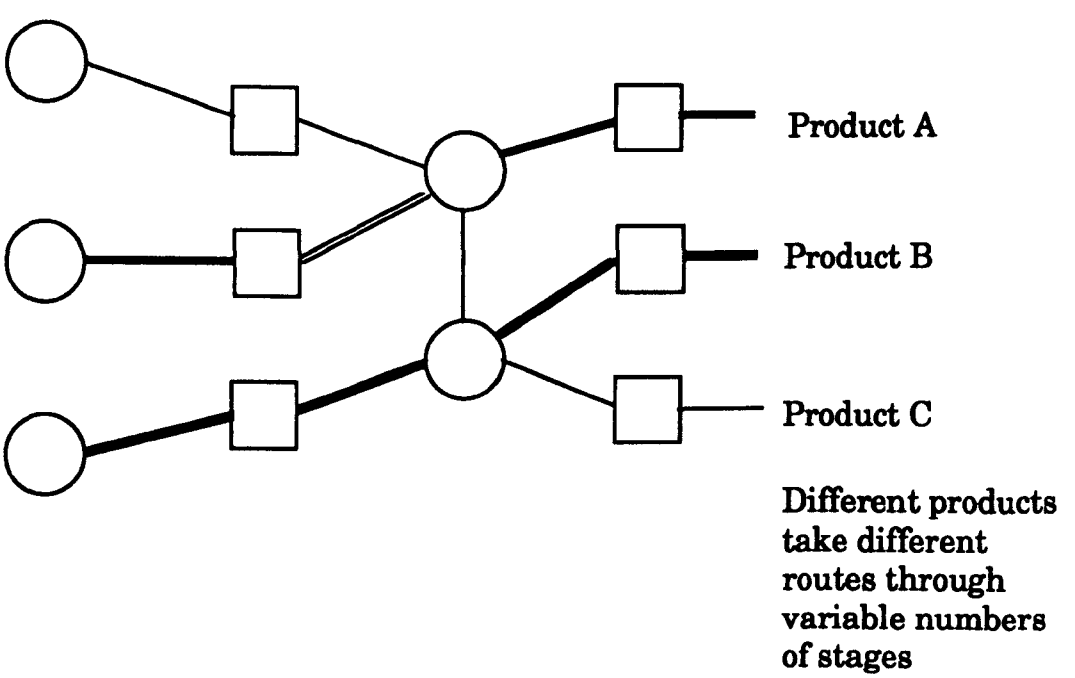



Figure 2.2. Multipurpose Configuration

The plant items which constitute the different stages are classified by their properties as:

1. Semi-continuous units which process product at a fixed rate as it is moved through them, for example a heat exchange unit, but do not run continuously throughout the production period.
2. Batch units, for example an incubation vessel, in which a static batch of product undergoes some reaction process.
3. Storage units which are used to decouple the production stages and simply provide temporary storage for the product between intermediate stages during the production process.

The availability of intermediate storage between production stages in a system is further classified as follows:

1. Unlimited Intermediate Storage (UIS), in which there is deemed to be no capacity restriction on the number of batches of product which can be stored at a particular stage.
2. Finite Intermediate Storage (FIS), in which the number of batches that can be stored at a particular stage is subject to a capacity restriction.
3. No Intermediate Storage (NIS), in which there is no intermediate storage between stages so they are directly coupled. However, a batch may be stored in the batch vessel in which it was processed or is going to be processed at the next stage.
4. Zero Wait (ZW), or No Wait (NW). In this mode there is also no storage between stages and the batch cannot be held in the batch vessel in which it was just processed, it must be moved directly to the next production stage.

Most batch plants have a combination of these interstage storage categories in them and are typically referred to as Mixed Intermediate Storage (MIS)

systems. The category of UIS is very unlikely to exist in reality although the categories of FIS, NIS, ZW and MIS are all reasonable. However, it is usually assumed that an interstage storage unit in the FIS category or a reaction vessel at an NIS production stage can hold a batch of any size which is not a reasonable assumption to make. In this case the finite nature of the storage simply represents the number of batches which can be in process at any time.

Semi-continuous processor units are also classified according to their processing rate:

1. Uniform units have a distinct processing rate which applies to all products for which they are used.
2. Unrelated units have different processing rates for different products.

### 2.1.2. Problem structure

In order to derive a particular problem structure for which analytical solutions can be formulated a number of assumptions are made which can include some or all of the following:

1. The plant is usually assumed to have a multiproduct configuration (a flow-shop) with a combination of multiple and single processing units at each stage.
2. The problem is usually viewed as non-passing, so that once a batch is in the system it cannot be bypassed by another batch introduced later, and the sequence of batches at each stage is always the same. Schedules produced on this basis are known as permutation schedules.
3. There is no pre-emption so that a batch cannot be interrupted by another batch once it has commenced processing on a unit.
4. The problem is usually assumed to be deterministic, so that all problem parameters are known in advance.

5.The problem is static, so that the production requirement remains unchanged for the duration of the scheduling period.
6.Batch transfer times between stages are negligible.
7.Unit set-up times are negligible.
8.A single interstage storage policy is used for the whole plant.

Some of these assumptions are highly unrealistic in the context of scheduling real plants. It is difficult to match the structure of a real plant against a simple flow-shop representation. Batch transfer times are a significant factor in the operation of many plants such as dairies in which a lot of semi-continuous processing is carried out. The problem is typically not static or deterministic, because additional orders may be received during the scheduling period and the plant items may be subject to variation in their processing times, and subject to random breakdowns.

The scheduling problem structure which is adapted from the literature on job-shop and flow-shop scheduling is viewed as one of producing a schedule which is optimal with respect to a particular performance criterion under some or all of the simplifying assumptions just described. A particular scheduling problem can be classified according to a set of parameters which are typically presented as follows:

1.A set of N product batches to be produced.
2.A set of M available processing units.
3.A performance criterion for which the schedule is to be optimised. In most cases the minimisation of makespan for all products is used, although other criteria, such as minimisation of mean tardiness, are also considered in some approaches.
4.A matrix of production times (Tij) associated with each product i and unit j.

5.A matrix of changeover times/costs associated with each ordered product pair.

6.A set of rules for the production process including:

a)The order in which the operations for each product must be carried out.

b)The manner in which intermediate storage between processing units is regulated.

c)The allowances which are made for subdividing product runs.

This is not a particularly useful problem structure for developing schedules for real plants. The representation of batches as discrete entities which move from stage to stage is unrealistic in many cases where batches of intermediate products are produced at one stage to act as a generic product to feed a number of later stages and do not move through the plant from start to finish. The representation of available production units is a significant simplification on the real situation where availability is severely constrained by the structure of the network and the way that product routing is controlled through the plant valve arrangement. The criteria by which a schedule is judged are not those by which real schedules are judged. In a recent survey of batch process manufacturers conducted by Musier and Evans [13] it was confirmed that the main pre-occupation of production management is in meeting production demand and customer due-date rather than optimising some performance measure of the plant.

## 2.2. APPROACHES DESCRIBED IN THE LITERATURE

The earlier approaches described in the literature present methods for scheduling based on the most simplified and restricted plant configuration classification and problem formulation as described above. Later work has built on this by attempting to overcome some simplifications and assumptions made and make the approaches more applicable to real world problems.

The most common plant configuration for which scheduling approaches are described is the multiproduct plant or flow-shop although some researchers have also developed approaches for the multipurpose or job-shop configuration. Approaches described include Mixed Integer Linear Programming (MILP), heuristic algorithms, heuristic search using branch and bound, and complete enumeration of all feasible schedules.

Birewar and Grossman [14], Rich and Prokapakis [15] and Kondili, Pantelides, and Sargent [16] describe MILP approaches in which the problem is put into a suitable formulation such that it can be solved by a MILP search procedure. The formulation involves the specification of an objective function representing the performance criterion and the variables for which values are sought such as batch start times. In addition the constraints, such as precedence between operations, on the values that can be taken by the variables in the objective function must be specified. The approach of Kondili, Pantelides and Sargent [16] attempts to address some of the simplifications and assumptions of other models and describes a structure called a State Task Network (STN) to define the process structure of the plant in the formulation. However, the STN only implicitly represents the configuration of the actual plant network in that each task in a given process has a set of units associated with it which can perform the task. In addition, as described by Machietto [17] practical use of the model requires that the resolution of time used is fairly coarse, for example discretised to half hour intervals, and a scheduling problem formulation may still involve several thousand variables.

Approaches which use heuristic algorithms are described by Kuriyan and Reklaitis [18] and [19], Rajagopalan and Karimi [20] and [21], Hasebe, Hashimoto and Takamatsu [22] and Daugherty and Felder [23]. Such approaches are generally carried out in two stages in which an initial starting sequence of batches to be processed through a plant configuration is generated by an heuristic, and then an attempt is made to improve this initial sequence

through a recursive technique such as neighbourhood search. Daugherty and Felder [23] describe an approach which uses a heuristic search for schedule improvement in conjunction with an expert system for choosing the initial scheduling heuristic based on an analysis of the problem characteristics. The definition of these characteristics appears to be restricted to a plant configuration as a flow-shop or job-shop and a performance criterion such as minimise makespan, so the approach does not appear to try and overcome the simplifying assumptions described previously. The approach described by Kuriyan and Reklaitis [19] and [20] develops sequencing heuristics and recurrence relations for scheduling the network flow-shop configuration in which there can be multiple parallel units at each stage. However, this flexibility is restricted because a series of stages with parallel units is treated as a set of parallel serial flow-shops with no crossovers of batches allowed between flow-shops "mid-stream" so the true network structure of a plant cannot be properly represented. Also the flexibility of the representation is restricted by the single storage policy which must be used, either UIS or ZW. Rajagopalan and Karimi [20] and [21] describe an approach called Idle Matrix Search (IMS) which generates an initial sequence using a modified Rapid Access procedure and then uses a neigbourhood search technique and recurrence relations for sequence improvement and evaluation. The aim in this approach is to move away from the representation of the plant as one with a UIS policy with negligible batch transfer times and setup times to one with a MIS policy with transfer times and setup times for batches. However, the plant is still restricted to being a serial flow-shop with a single batch unit or semi-continuous subtrain at each stage and the performance criterion is still minimise makespan.

Heuristic search using branch and bound is described by Knopf [24] and Ku and Karimi [25] for scheduling multiproduct plants, and by Felder, Kester and Moldin [26] for multipurpose plants. A branch and bound procedure is a carried out as a tree search of possible sequences coupled with an elimination

procedure to remove the requirement to explore all branches through a bound calculation. This calculation gives an estimate of whether it is worth exploring a branch further in comparison with the best sequence developed so far. Knopf [24] is concerned with scheduling of a two stage flow-shop representation of a dairy in which batch transfer times are significant because of the semi-continuous processing carried out and in which the effects of finite intermediate storage must also be taken into account. He comments that these factors mean that there will be both discrete and continuous changes in the values of system variables, for example the continuous rate of a processor will undergo a discrete change when a storage vessel becomes full, and that 'Because of these inherent difficulties, an analytic solution to the generalised FIS problem is not expected.' The approach uses a combined discrete/ continuous simulation model in order to take these factors into account. The simulation is first used to evaluate the completion time of an initial sequence generated using Johnson's algorithm which serves as an intial upper bound. It then uses a branch and bound procedure in conjunction with the simulation which stops the evaluation of a potential sequence once its simulation execution time exceeds that of the current upper bound. This approach uses a representation which is much closer to a realistic situation because the factors of transfer time and the effect of finite capacity on plant operation are taken into account. However, the problem formulation still assumes that the plant is structured as a flow-shop in which only permutation, non pre-emptive schedules are considered and the performance criterion is to minimise makespan.

Egli and Rippin [27] describe an approach which involves complete enumeration of all possible schedules. The approach is directed towards multipurpose plants and allows the user to specify a number of realistic constraints about the problem such as the specific assignment of process tasks to plant items, the stability of product intermediates, maximum levels of demand on utility resources such as steam and electricity, and maximum,

minimum and buffer stock levels for intermediate and final products. Each feasible schedule of batches which satisfies product demands subject to these constraints is generated and evaluated on a least cost basis, with the lowest cost schedule retained at the end of the search. However, it still relies on some unrealistic simplifying assumptions. For example, the stock levels for products are specified as simplified aggregate values, and because the approach relies on complete enumeration of feasible sequences the size of the problem must be necessarily restricted.

## 2.3.DISCUSSION

It is commonly recognised that algorithmic approaches are restricted in their applicability because the real world situation will include additional factors that make it more complex than any single one of these approaches can cope with. As well as the classification scheme for intermediate storage, the policy used for it's operation can have a marked effect on the performance of the plant, for example whether a vessel is operated according to a fill and empty policy, effectively as a balance/ surge tank, or whether a policy of fill then empty is adopted. The necessity to clean the plant in between batches or after a period of operation of the plant is also a factor which must be considered, particularly in sections of the industry such as the food industry where hygiene and quality cannot be ignored. The size of intermediate storage units, and batches does not necessarily correspond one to one as is usually assumed if finite storage is taken into account. The choice of which plant item to use in a given process route is often situation specific and cannot be generalised in an algorithm.

Wiede [28] discusses problems with the use of algorithmic approaches for scheduling as follows; '..the solution algorithms available for various forms of the scheduling problem are quite diverse and specific to problem structure. The scheduling problem under consideration must be very well defined so that

it is clear which algorithm is appropriate.... Moreover, specialised constraints, schedule interruptions and other non-ideal problem constraints may make it difficult to force the resulting scheduling problem to conform to any of the standard forms.'

The approach put forward in by Wiede [28], and Musier and Evans [29] among others to overcome these limitations is to integrate the algorithmic approaches into an interactive system, so that the scheduler can either manipulate the way that the schedule is generated by the algorithm, or adjust the schedule after generation by the algorithm. Ventker, Baker and Neville [30] describe the use of PROSIT (PROcess Scheduling with Interactive Technology) which was introduced for use in the process industry with these types of facilities. They comment that the manual scheduling facilities were well used, but there was a resistance to the use of the algorithms through lack of understanding and a lack of control over schedule development when they were used. In addition, the representations of the plant and the problem assumptions used in these approaches are still limited by the algorithms used, which means that the schedule feasibility must still be questionable.

For any short term scheduling technique to be of practical use in a batch process plant it must take all relevant factors into account and produce a schedule which is feasible as well as meeting the required performance criteria. It is no use having a sophisticated process control and batch management system if the schedule driving it is poorly formulated and cannot be implemented. For example, some recent research which has been described by Cott and Machietto [5] on the Computer Aided Operation (CAO) of batch plants includes on-line modification of a schedule to account for process variations. However, this type of system requires a scheduler which can correctly take the real plant constraints into account in order for it to function well. Therefore, the pre-requisite for application of scheduling and control in a batch process plant is a model of the plant for scheduling and control that will

accurately represent the constraints within the plant and allow the generation of feasible schedules for the plant which can then be implemented.

It seems unlikely that general algorithmic approaches to the modelling of batch process plants will be able to incorporate the complexities in a typical plant sufficiently well to produce schedules which can actually be implemented. The use of Discrete Event Simulation (DES) and Artificial Intelligence (AI) techniques enables these complexities to be represented. Knopf [24] recognises this and uses a simulation model to more accurately represent a batch plant as part of his procedure. The remainder of this thesis will concentrate on the application of appropriate techniques from these areas in a batch process scheduling and control system.

# CHAPTER 3: SIMULATION BASED SCHEDULING AND CONTROL

## 3.0.INTRODUCTION

There has been a considerable amount of research into the use of Discrete Event Simulation (DES) based systems for scheduling and on-line control in manufacturing systems generally, because it can co-ordinate the activities of a large number of entities in a complex finite capacity system over time. This has resulted in a number of products such as PROVISA [31] which are now available commercially and in use for this purpose. However, there is a certain amount of debate about the validity of simulation for scheduling because it uses deterministic data. Its suitability is also questioned considering it's traditional role as a descriptive rather than a prescriptive technique. In this traditional role simulation has been used for the analysis of the operating characteristics of existing manufacturing systems and the design of new ones. Pegden [32] comments on the two uses of simulation 'Although the basic manufacturing modelling requirements are the same in both design and control applications, there are some basic differences in the requirements on the simulation tools needed in these applications.' In addition to these differences in the requirements, there are also marked differences between the way simulation is used in these applications. This is because the objectives of the two uses are different, so the characteristics of a particular system being modelled are viewed differently, and given different emphasis in their relative importance in terms of modelling and experimentation.

This chapter will briefly describe the objectives, features and use of simulation for analysis and design purposes. The use of simulation for scheduling and control will then be described to indicate differences from its analysis and design role and the requirements for a simulation tool developed for this purpose. The validity and suitability of using simulation for scheduling will be discussed in the light of the objections which have been raised.

## 3.1.SIMULATION FOR ANALYSIS AND DESIGN

### 3.1.1.Objectives

The objectives of simulation in this role are descriptive, to illustrate the behaviour of a system, rather than to indicate how to arrive at a particular solution (prescription). Systems are characterised and classified by their time dependent behaviour and the behaviour of the entities in them. Most systems have at least some entities which are subject to stochastic variation and random occurrences such as breakdowns. In this role it is concerned with estimating values for response variables (outputs) of interest, such as the makespan of batches/ jobs in a system, for a given set of inputs. Each input which can be controlled by the user is called a factor and can be set at a particular level for a specific run of the simulation. It has three main modes of use with respect to the type of objectives which could be defined as described by Davies and O'Keefe [33]:

1. Prediction of output variable values from a single system configuration, to determine '..average results and confidence limits of a simulation run which has specific factor levels.'
2. Comparison of system configurations '..to determine whether one option is better than another..'
3. Investigation of one or more system configurations, typically using VIS, to '..indicate the major factors which affect the flow of entities in a system, but is not required to provide precise answers.'

Determining the objectives according to one of the above criteria will indicate which output response variables are of interest, how accurately they need to be estimated, and which input factors/ levels are thought to affect them. The focus of all these objectives is on "typical" system behaviour. Law and Kelton

[34] sum up the traditional use of simulation and say that '..a simulation is a computer based statistical sampling experiment.'

### 3.1.2.Building a Simulation for Design and Analysis

A simulation model which is used in this role is typically built for a "one-off" study of system behaviour. It must be built so that the input factors and levels can be varied as required, but the ability to easily update the model structure apart from this is not necessarily a high priority, so the data structures, behaviour, and control rules may all be coded in the simulation program itself. A typical approach to this type of requirement is the Simulation Programming Language (SPL) SIMAN [34], where the behavioural model is built separately from the experimental "frame", so that the changes to input factors and levels can be made easily without altering the model code. In such a simulation the control logic will typically be based on simple probabilities that an entity will take one path or another as described by O'Keefe and Roach [35], or the use of simple dispatching rules which choose entities from queues based on the value of an attribute associated with that entity.

Because of the emphasis on modelling stochastic behaviour and the occurrence of random events an important part of building a simulation model is choosing the correct discrete and continuous probability distributions and defining their parameters for the entities in the system. Discrete distributions are used to model things such as random branching in the possible behaviour choices of entities, and the setting of values for entity characteristics when they enter the model. Continuous distributions are used to model such things as the duration of an activity, and the time between random occurences such as plant breakdowns. Simulation tools usually contain comprehensive features for defining these distributions and sampling from them as a simulation run progresses.

### 3.1.3.Experimentation with a Model

Assuming that a model has been built incorporating the correct control logic, distributions, and other data, and has been properly validated using an appropriate technique it can be used for experimentation. When using a simulation for analysis and design purposes measuring response variable values from the typical behaviour of the system in relation to the defined objectives is usually the reason for running the simulation. In some cases accurate estimation of response variable values is required because of the stochastic nature of the system and this requires careful experimental design and analysis. In some other cases the observation of trends in results or comparison of difference in magnitude between results are more important than accuracy although this still requires careful experimental design and analysis.

In designing the experiments to be carried out and analysing the results the simulation user/ analyst must take into account the purpose of the study, and the number of input factors and levels involved. Techniques which can be applied so that statistical analysis of the results can be carried out are described by authors such as Law and Kelton [34] and Pidd [36]. It is often important to get rid of any bias in the results due to initial starting conditions. This can be achieved by starting the model empty, and looking for the point at which a steady state is reached by using the cumulative moving average approach. Recording of the output for steady-state analysis is started from this point in all replications of the experiment that are made so that the results which are used for analysis do not stem directly from a real system initial state.

In order to predict the mean and confidence limits of particular response variables of interest for a single system configuration a set of Independent Identically Distributed (IID) sample values for each variable must be

generated, so that the sample mean and variance can be calculated, and a confidence interval constructed for the precision of the sample mean as an estimate of the population mean. However, simulation output data is generally autocorrelated, in that one value depends to an extent on the previous value, so the output from a single run of the simulation cannot be used to calculate the sample variance. Therefore, a number of independent replications must be made so that an unbiased sample variance can be calculated from the different replication response variable means, and thus a confidence interval can be constructed.

In a large number of cases the simulation will be being used for comparison purposes. In the most basic case, where two different factors at one level or one factor at two levels are being compared, then the experimental process involves running enough replications of each system configuration to get good response variable estimates and then carrying out a parametric test to obtain a confidence interval of whether any difference between the response variables is significant.

It can be seen from this brief description that the use of simulation in this way is usually based on a relatively inflexible model, containing simple control logic, and using data sampled from distributions rather than real system data. The experimental process may require a large number of long runs of the simulation to be made, and the results obtained are of a statistical nature referring to a model of the system run under "typical" conditions.

## 3.2.THE OBJECTIVES AND USE OF SIMULATION FOR SCHEDULING

### 3.2.1.Objectives

Scheduling is a prescriptive task in which the aim is to determine the best sequence for the production of a set of batches or jobs and their assignment to plant items over time to meet the scheduling objectives subject to constraints

imposed by the finite capacity of the system, the structure of the system, and operating restrictions such as the requirement to clean plant items in a batch plant. The main reasons for using simulation in this context are based on its ability to accurately co-ordinate the activities of a large number of entities over time in a complex system taking into account the actual constraints that exist. This means that a schedule produced with a suitably developed simulation will enable a user to predict realistic completion times for the production which is to be carried out, and predict whether the scheduling objectives can be met. In addition the output from a suitably formatted trace of the simulation can act as a detailed timing plan of the allocation of resources to production activities which can be used for control within a manufacturing plant.

### 3.2.2.Building a Simulation for Scheduling

In order to accomodate these objectives requires a suitable modelling tool to be used. It is important that the complexities of system structure and specific control logic can be modelled if the schedule is going to be feasible and acceptable for implementation. In a scheduling simulation, which is used on a very regular basis, it is important that the static model structure, control logic, and the behavioural logic are kept separate to facilitate changes to the structure, and control logic if necessary. For example, the scheduling objectives of the plant may change over time, and the model of the plant configuration must keep up with any changes made to the real plant. It is also important that these changes can be facilitated through changes to the data of the model rather than the code of the program. The range of possible approaches to building a simulation model is quite diverse, especially when knowledge-based approaches are taken into account. It goes from using a general purpose procedural or declarative programming language (FORTRAN, PROLOG), to a general purpose procedural language simulation library (SEE-WHY), or an AI simulation toolkit (Simulation Craft), to a SPL (SIMAN), to a

simulation program generator (CAPS/ ECSL), to a generic modelling system (WITNESS), (Figure 3.1. (after O'Keefe and Haddock [37])). The scope of these approaches, in terms of the classes of system which they can model, ranges from any class using a general purpose language, to a specific class or sub-class using a generic tool. The modelling effort and development time required for modelling a specific system decreases as the scope of applicability is narrowed, because more of the work has already been done by the developer of the tool used, so more of the modelling can be done through data input rather than through writing code. In the case of a generic tool which is aimed at a specific system or class of systems it should be possible to build up a detailed model through data input only. Therefore, a suitable generic simulation tool should enable the requirements for building a scheduling simulation to be met and most simulation systems which have been developed for scheduling are based on the generic approach. The development of an appropriate generic tool for scheduling will be discussed in detail in Section 3.3. on generic simulation.

Control Logic

Some scheduling simulation approaches, such as that described by Bollino [38], simply allow the user to specify which of a number of standard queuing priority rules should be used to control production through a plant. However, In a scheduling system a true representation of decision making based on the global system state is desirable as described by Larsen and Alting [39] because queueing priority rules only use local information based on the operation for which the batches/ jobs are queueing, which implies limited decision scope. Bhattacharrya, Roy and Huang [6] say that the effects of this are that '..requirements of customer demand are not specifically considered, nor is any consideration given to potential capacity shortages'. This is particularly true of a batch process plant in which the activities carried out in different parts of the plant are very interdependent. For example Barnes and Gardner [41] describe a simulation for scheduling a batch process plant where

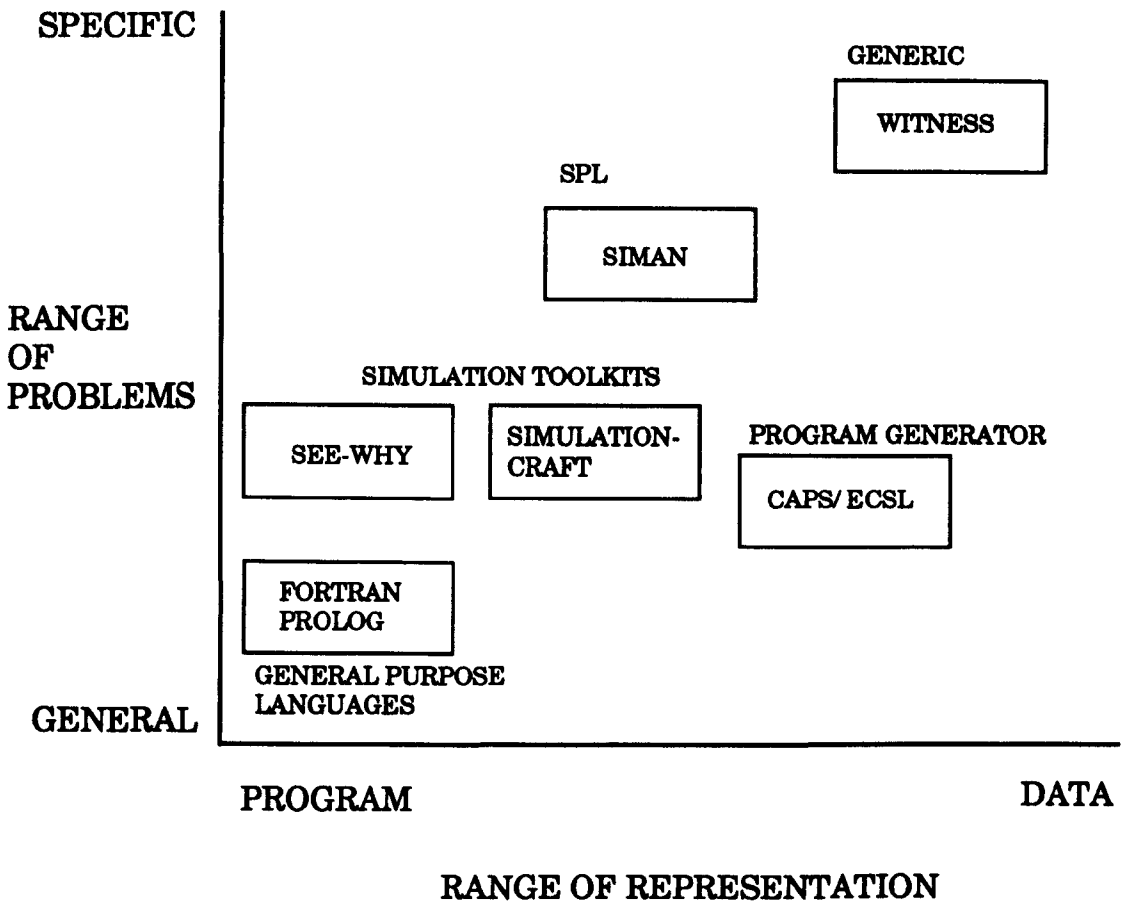



Figure 3.1. Model Building Approaches
(After O'Keefe and Haddock)

the ability to produce good schedules depends on assessing upstream storage availability when sequencing an earlier stage. Therefore the ability to include complex control logic in the model is important. However, there are problems with the use of procedural simulation languages to develop appropriate tools in this respect. Moriera da Silva and Bastos [41] and Bhattacharyya, Roy and Huang [42] who are concerned with the development of generic simulation based control systems for Flexible Manufacturing Systems (FMS) comment on the restrictions of simple priority rules and describe the need for more comprehensive control rules based on total system state. They comment on the lack of flexibility in traditional procedural simulation languages for representing more complex decision rules in a generic scheduling system. Moriera da Silva and Bastos [41] discuss the development of their system; 'During the initial stages of development of VISUALPLAN the decision mechanisms were materialised by a FORTRAN 77 library of subroutines with classic (e.g. FIFO, MOR, LPT, etc.) priority rules which could be chosen interactively by the user at any stage of the run. Soon it became apparent that more sophisticated rules that could take into consideration the total state of the system needed to be developed...The modelling of such rules was then tried in FORTRAN but, even for incipient ones, the approach was soon dismissed because of the difficulty of dealing with recursivity in that environment.' To overcome these problems both sets of authors have proposed the use of a knowledge-based hybrid approach to combine the procedural generic behavioural simulation model with a declarative knowledge base to provide the flexibility for representing the complex control logic in the simulation.

Structural and Control Data Representation

In a simulation which is used for scheduling the static data such as activity durations and process rates used to drive the simulation is deterministic. The stochastic nature of activities is not modelled, and random events such as

breakdowns are not incorporated explicitly because this is inappropriate for the short time-scale for which the simulation is run. Much more emphasis is put on the ability to represent the structure of the system and to handle large amounts of real control data about the manufacturing system required for scheduling purposes. Wyman [43] comments on the features of traditional simulation tools and their applicability to scheduling applications saying that '..manufacturing companies need a far richer set of modelling features to express realistic details that can be followed on the shop floor.' There should be a number of features in a scheduling simulation tool for this purpose to enable the output from the model to be used for control of a batch plant. It should be possible to represent specific orders and their status. It should be possible to correctly represent the network structure of a batch plant and the connectivity of plant items. It should be possible to represent control information such as process plans or production recipes which describe how to manufacture a particular product, what possible routings could be taken through the plant and which plant items could be used for each stage including preferences. In a batch plant scheduling tool Bill of Process information is needed which describes how much material at each stage of a process is required for the manufacture of a particular product and how the material balance of the system is affected by a process. The degree to which these features are provided by current tools documented in the literature, and importance of them to scheduling in a batch process plant will be described in detail in Chapter 6 on the specific modelling of batch plants.

### 3.2.3.Use of Simulation for Scheduling

Validation of a model of a particular plant should be carried out to test whether it will produce schedules which are feasible and reasonable. A simulation model will never be able to reproduce a schedule produced in real life because human schedulers are inconsistent. Therefore this validation should look at things such as the levels of plant utilisation and whether

resources are correctly assigned to production activities taking constraints into account and the control rules incorporated in the model. This level of validation should be distinguished from assessment of the schedules that are produced and this is discussed further in Chapter 9 on model implementation.

In order to develop a schedule with a simulation model it must be initialised with the current plant status, and then run using an appropriate strategy.

Simulation Initialisation

In a scheduling simulation it is important that the conditions at the start of the simulation run match those of the real plant at the start of the scheduling period, rather than using "typical" conditions as in traditional simulation. The aim is not to get rid of the effect of these initial conditions, but to predict whether it can meet the scheduling objectives starting from them. Two types of data are required to initialise the simulation for scheduling: data on the production requirements to be met, and data on the state of the plant at the start of the scheduling period. The simulation may be initialised through data input by the user of the system, for example Sturrock and Higley [44] describe the use of data files for configuration and initialisation of a model. The parameters of the model are set through one file, and the plant can be preloaded with work through another, together with a list of items to be processed in a desired order and any planned downtimes such as maintenance work. However, automation of the initialisation process is desirable, because of the time taken to enter large amounts of data manually, and the potential for errors in a manual approach. Grant [45] describes the features of SCHED/ SIM, (now renamed FACTOR), and comments on the integration of the simulation into the other components of the manufacturing environment, saying that an important feature is '..an interface to the MRP system to extract master schedule information as well as shop floor data...This interface collects the current list of orders to be processed at the production department

and automatically loads those into the SCHED/ SIM system. A similar interface is provided to collect information regarding the current location and status of released orders from the shop floor data collection system.' The information is stored in a local database which can be accessed and edited by the scheduler. He further comments that this interface has to be custom developed for the particular system that the scheduler is interfaced to. The issue of initialisation is an important one in developing a scheduling system that can be successfully implemented in a real environment. However, there are some aspects which have not yet been fully addressed. For example, how to initialise the model if it is desired to carry out scheduling in advance of the current time when the actual plant status at the start of the scheduling period will not be known, or the system is operating in an environment where there is no cut off point in production so that the status of the real system can be downloaded to the simulation database. These issues will be discussed in Chapter 9 on system implementation.

Simulation Execution

In a scheduling system, the period for which the simulation is run depends on the scheduling horizon which may be based on the timescale of the objectives, or the potential for the schedule to be corrupted as described by Sturrock and Higley [44]. In a batch process plant such as a dairy producing Fast Moving Consumer Goods (FMCG) short term scheduling is typically carried out daily. In a discrete parts manufacturing environment the requirement might only be to carry out scheduling weekly because the potential for the schedule to become corrupted is less. In addition the simulation might be used for more than one level of scheduling. Spooner [46] describes the use of a simulation model over two time horizons in a discrete parts batch manufacturing environment with long lead times. It is run on a long term basis, for example a few months, to determine the likely overloads in the plant and any lateness of

batches, and on a day to day basis to control the flow of batches through the various manufacturing stages.

Simulation will not search directly for an optimal solution to the scheduling problem, although it has been used by Knopf [24] as part of a branch and bound procedure for scheduling in a dairy. It always moves forward through the search space of system states, with no potential for backtracking once a scheduling decision has been made. It is basically up to the user of the model to consider whether the results from a particular simulation run, with a given set of input parameters and control rules, are satisfactory with respect to the scheduling goals or not. A number of different approaches have been put forward for the use of simulation in this respect.

Iterative Approach

Some researchers such as Grant and Lagoni [47] recommend an iterative approach is taken, so that after analysis of the results of a particular simulation run changes may be made to the parameters of the model, for example altering batch priorities, which is then run again, and re- analysed to see if any improvement to the schedule can be obtained. A number of researchers have tried to automate this approach through the use of an expert system linked to the simulation model. Sun [48] describes an Intelligent Front End (IFE) combination of an expert system and simulation model to develop the control policy for a production system over a short term scheduling horizon. The expert system contains rules to analyse the results of the simulation output to assess reasons why a specifed production objective has not been met. This is in order to make control decisions to improve the schedule for the system in conjunction with queries to the user. Modifications are made to the model inputs as a result of this dialogue, which is re- run to produce a new set of results. The procedure is repeated until the specified objective for the system, typically to meet due dates, is met as far as possible.

Control Strategy Approach

Other researchers have put forward the view that a control strategy approach should be adopted, with only limited iterative experimentation being appropriate or necessary. Roy, McCarthy and Klapatyj [49] base this approach on Optimised Production Technology (OPT), with initial use of the simulation to analyse the behaviour of the system to determine a good input control policy for release of work into the system. This should maintain lead times at predetermined levels, and keep critical resources loaded with just enough work to keep them working at full capacity. The input policy is then used to run the model for the scheduling period, with any experimentation to eliminate job lateness restricted to the resolution of capacity constraints through available options such as overtime, sub- contracting, or reducing the workload on the system through changes to the Master Production Schedule or re- negotiation of due date with the customer. The aim of this approach is to make the task of the scheduler more manageable because the system already has the rules for developing efficient schedules built in. In a real life situation there will only be a few options for changing how a short term schedule is derived so there are only a limited number of experimental options to concentrate on. This is particularly true in a batch plant environment such as a dairy, where the scheduling time-scale is very short so both the options for change to a schedule and the time to carry out experimentation will be limited.

Interactive Approaches

Visual Interactive Simulation (VIS), in which the course of the simulation over time is animated and can be halted either by the user or the model for interaction in which the user may change some of the model parameters or entity data, was developed by Hurrion [50], and demonstrated through a number of models including a job- shop and a coal merchants yard. These

models basically relied on the user to judge how to control the operations of the system, using the status of the simulation to guide them. The main aim was to demonstrate that the ability of the user to interact with the model gave results which benefitted from the incorporation of complex decision making logic rather than using a simple priority rule scheme for controlling the simulation. Early applications of the approach were developed for the batch and continuous process industries. Secker [51] developed a VIS for scheduling a batch process plant, and Fisher [52] developed a model for the control of a continuous process plant. Both these systems were intended to be used in real environments for scheduling/ decision making purposes. Fisher describes the use of the model in this respect as an exploratory one. The evaluation of alternative strategies initiated and controlled through user interaction is carried out by watching how a final outcome is reached through the iconic graphic display.

The use of visual interactive techniques for scheduling has moved towards the use of interactive planning board approaches, rather than the use of interactive simulation, because this gives the user a broader cumulative picture of the status of the system over time. O'Keefe and Haddock [37] comment on the problems with using the iconic display of a VIS for scheduling saying 'An iconic display is very useful for micro- level simulation... When we want to look at scheduling and flow issues, however, the iconic display is too detailed and provides no cumulative information.' In the case of a planning board based system, an initial schedule is generated by the system and output in the form of a Gantt chart. This chart may be the output from a rule-based simulation as in the PROVISA system described by Beadle [31], or a batch sequencing algorithm as in the approach for batch process plants described by Musier and Evans [29]. Generally speaking, after the initial schedule has been generated, the user can manipulate the jobs or batches on the planning board, with warnings from the system if constraints are breached, to attempt to improve the schedule if possible. However, planning boards are difficult to use

for large and complex systems when there may be a considerable number of "live" jobs or batches and it will be difficult to see all the consequences of making any manual changes to a schedule.

Simulation Output

The required output from the simulation is not just summary statistics for the values taken by the output measures of interest, but a detailed schedule of the start and end times of the activities of entities, the progress of batches/ jobs through the plant, and the allocation of resources during the simulation. Thus the output from the system is based on a trace of the events which occured, formatted appropriately into reports which are useful to the user. Sturrock and Higley [53] describe how the information in a sequential event trace can be formatted to view the operations of the plant from different perspectives such as a machine loading schedule, and a product tracking schedule. The output can also be formatted graphically, typically in a Gantt chart or planning board format, for example as in the PROVISA system [31].

It can be seen from this description that in contrast to models developed for analysis and design purposes scheduling simulation models are deterministic, require the incorporation of complex control logic and a detailed structural description of the system being modelled, are generally run once only for a particular scenario representing a short time-scale, and the behaviour of interest is the specific behaviour based on the real initial conditions of the system.

## 3.3.GENERIC MODELLING FOR SCHEDULING

A generic simulation tool is most specific in the range of systems that it can be used to model (Figure 3.1.), because most of the modelling requirement to build a representation of a specific system has been moved from code to data.

The "core" of the system class behaviour is left as code, which can be detailed through data input to produce a specific model from the system class. This means that as long as the specific system to be modelled is within the class of systems that the generic tool is aimed at then it can be modelled through data input only without compromise in the detail of the model produced. This is unlike a program generator approach which also allows the user to build up a model through data input, but aims to have more general applicability and only produces a model which is perhaps 70% complete as described by Crookes [54] and relies on the user to customise the code to produce the final model.

The generic approach has been widely adopted for the simulation of manufacturing systems for all types of analysis. O'Keefe and Haddock [37] describe two generic systems for analysis of FMS behaviour; one traditional batch simulator for developing statistics on traditional performance measures such as part flowtime, and one with VIS facilities for investigative purposes. Hills and Rogers [55] describe the use of a generic system for modelling batch process plants to improve the understanding of the dynamics of an existing plant in an effort to improve efficiency.

A generic modelling tool can vary considerably in its coverage of real systems and the detail with which they can be modelled. For example Crookes [54] describes the building of a simulation model which is generic with respect to a particular plant. Bhattacharyya, Roy and Huang [42] describe the development and features of a generic simulation system for the FMS class of manufacturing systems and comment that the class of systems for which the tool is developed must be well defined and restricted so that '..the general behaviour pattern is sufficiently similar in nature for this to be embedded in such a model..'. However, O'Keefe and Haddock [37] comment on system complexity and generic models; saying that they would 'provide sufficient detail for sizing and looking at the effect of various constraints, but may not be appropriate for detailed scheduling. Control issues, such as complex

interaction between AGVs, where timing is liable to be crucial, will almost certainly will require a SPL of some sort.' Moreira da Silva and Bastos [41] and Bhattacharyya, Roy and Huang [42] have put forward the use of a hybrid approach which allows the control logic and system structure to be represented in a declarative format so that it can be tailored to the specific plant being modelled. This overcomes these reservations about the use of generic models for scheduling because a declarative language makes it easier to increase functionality incrementally.

## 3.4.DISCUSSION

### 3.4.1.The Validity of Simulation as a tool for Scheduling

From the preceding descriptions of the use and development of traditional and scheduling/ control simulation models it can be seen that there are a number of differences between the two but the main reason some researchers question the validity of simulation for scheduling is the use of deterministic data. This stems primarily from the original role of simulation as an experimental tool for analysing the behaviour of systems containing stochastic elements. For example, Harmonosky [56] discusses the decision making mode of an on-line simulation used to evaluate alternative control strategies over an 8 - 24 hour period and says 'The question must be addressed of what types of random events are reasonable to include during these types of run- lengths and/ or what would be the innaccuracies introduced with assuming deterministic behaviour'. In fact the use of randomness in such short periods of time is statistically invalid so the use of deterministic data for scheduling simulation is the correct approach to take. Therefore the validity of the technique is not questionable in this respect. However, this does not mean that randomness should not be taken into account at all. Pegden [32] comments on the assumption that random events should simply be left out of the scheduling model with no consideration of their effects saying that '..any performance

values... generated by the simulation run using this approach are optimistic because they are based on an unrealistic facility corresponding to no breakdowns. This approach to scheduling creates a situation where the actual performance is worse than the performance predicted by the simulation.' Roy, McCarthy and Klapatyj [49] describe the use of conservative activity durations and bringing forward due dates to give some protection against the stochastic nature of the system and possibility of random events, which will overcome the reservations made above. They say that 'The use of such protective measures could be reduced as managements confidence in the control system increases and the reliability of the physical systems is understood and improved where necessary.' so the performance of the system can be increased without raising expectations that cannot be met.

### 3.4.2.The Suitability of Simulation for Scheduling

As discussed by Roy [8] simulation is able to accurately represent the finite capacity of a system, the logic of interacting entity operations, complex control logic can be incorporated, and it's internal clock means that it will predict realistic start and finish times for activities within the system. Thus it is very suitable for scheduling because it can predict a realistic completion time for a job within a system and '..produce a detailed timing plan for the start and end times of each of it's operations, which can be used for effective day- to- day control.' This is what the main management requirement from a scheduling system is because their prime concern is to meet production obligations through an acceptable feasible plan. If a simulation model is carefully developed and validated and contains the appropriate decision logic then the user can have confidence that the standard of schedules produced will be within an acceptable performance range for a prescribed range of inputs. The issues which have been raised against the suitability of simulation for scheduling are concerned with the efficiency of the schedule produced and are secondary concerns in the light of this main requirement. In a general review

of scheduling approaches Rodammer and White [57] said that the experimental nature of simulation is its principal disadvantage because '..even highly accurate modelling does not guarantee that optimal or even good schedules will be found experimentally.' As described earlier scheduling in a real situation is based on a number of conflicting goals so trying to find an optimal schedule with respect to single goal such as minimisation of makespan will probably bring no benefits and may be detrimental to the business through missed due dates for example. In addition an optimal or "good" schedule produced using an analytical technique which relies on a model of the plant that contains so many simplifying assumptions that it cannot actually be implemented is of little use in a practical situation.

### 3.4.3.Features Required in a Simulation Model for Scheduling of Batch Plants

The system must be generic and should include the ability to represent complex control logic, structural and control data at a sufficient level of detail through data input only to properly represent the control strategy and constraints on schedule development. Appropriate methods of representation for these features are therefore required. AI research is concerned with knowledge representation issues and ways of reasoning about constraints, and approaches from this area are reviewed in the next chapter. Some of the required features cannot easily be represented in a generic model with sufficient flexibility and detail using a procedural language. Therefore, the hybrid approach has been put forward to overcome reservations on the use of simulation for scheduling, and this will be discussed in Chapter 5.

# CHAPTER 4 THE APPLICATION OF ARTIFICIAL INTELLIGENCE RESEARCH TO SCHEDULING AND MODELLING

## 4.0.INTRODUCTION

The reasons put forward for applying AI based approaches to modelling and scheduling are essentially the same as the reasons for applying simulation based approaches. That is to overcome the simplifying assumptions of analytical approaches in order to determine good feasible schedules which meet scheduling goals. The issues adressed are knowledge representation, and control of the search for a schedule. In a system for scheduling batch process plants these are both important. This chapter will review approaches which are applicable to the representation of the plant network and the constraints on the connectivity of plant items, the representation of the production activities and how they are related in production "recipes", the representation of the control strategy, and how the plant items should be allocated to resources as scheduling proceeds.

## 4.1.APPROACHES TO SCHEDULING

Fox [58] discusses techniques used in AI based scheduling saying that a core AI concept is search and the aim of AI techniques used in scheduling is to reduce the search space of alternative schedules to manageable proportions. Techniques used to do this include the use of situational knowledge in rule-based systems, the reformulation of the problem as a simpler task whose solution can be used to guide the original problems solution (heuristically guided search), the use of opportunism, and the development of powerful knowledge representation techniques which allow pattern representation and abstraction.

### 4.1.1.Rule-based Systems

Rule-based or expert systems have been implemented successfully in a number of fields concerned with manufacturing, such as process diagnostics described by Sachs et al.[59] and process control described by Efstathiou [60]. The success of expert systems generally has prompted researchers to apply the technique to scheduling as reviewed in a number of recent papers, for example Kanet and Adelsburger [61], and Kusiak and Chen [62]. Rule-based systems are typically dynamic in their approach to scheduling, containing a set of domain specific scheduling rules, and making decisions based on the current system state. This state may be derived directly from the plant, or from a model of the plant which has an appropriate time advancement mechanism. Well known examples which operate in this way include those described by Ben- Ariah [63], and Bruno et al. [64]. There are also a number of systems reported where the purpose of the expert system is not to carry out the scheduling directly but to use the rule-base to determine how to schedule the system. For example Daugherty and Felder [23] describe the use of an expert system to choose a heuristic for the initial scheduling of a multipurpose batch process plant. Copas and Browne [65] describe the use of a rule- based system which analyses a manufacturing system to determine potential bottlenecks and aids the user in the selection of a heuristic for scheduling.

### 4.1.2.Heuristically Guided Search

The use of rule-based systems for scheduling has been questioned by some researchers such as Fox [58], and Kempf et al. [66]. They cite a number of reasons which they argue make purely rule-based systems unsuitable:

1. There are difficulties with building a rule- base both in terms of correctly understanding the methods used by schedulers, and translating these methods into rules to be used by the system.

2.The rule- base must be sufficiently comprehensive to adapt to a wide variety of situations. The production environment is often subject to change which quickly makes a system rule-base obsolete.

3.Expert systems which only consider the current state of the plant tend to be myopic in their scheduling strategy.

4.Factory scheduling is so complex that the schedules produced by human schedulers are not necessarily particularly good, so simply implementing their scheduling rules in an expert system will not result in a better schedule.

5.Human schedulers use manual methods, which tend to focus on individual decisions and their immediate effects rather than the global effects.

Fox [58] and Kempf et al. [66] say that computerised scheduling systems which maintain a "map" of resource availability over time, and use extensive search, can focus on global system performance much better. Scheduling is viewed as a heuristically guided search in these approaches. In order to reduce the complexity of the search space it may be decomposed into a hierarchy of levels of problem aggregation.

The most well known examples of the heuristically guided search approach are the Intelligent Scheduling and Information System (ISIS) developed by Fox [67], and its more recent evolution the Opportunistic Intelligent Scheduler (OPIS) [68] which were both developed for job-shop scheduling in particular. There are also examples of similar systems for job-shop scheduling from other researchers, such as SOJA [69], CHRONOS II [70], and OPAL/ OSCAR [71], and a system from Steffen [72] for scheduling a batch/ continuous process plant. These systems are known as constraint directed systems. More recently the aim has been to reformulate parts of the problem topology as a Constraint Satisfaction Problem (CSP) in order to use applicable techniques from this area of research. Among examples of systems which include the use of these

techniques is the Distributed Asynchronous Scheduler (DAS) described by Burke and Prosser [73] which distributes the scheduling process across a number of autonomous agents each of which attempts to solve its own particular dynamic CSP. Elleby et al. [74] describe a dynamic adaptive scheduling system in which sets of schedules are implicitly maintained as the set of solutions to the current CSP. Fox et al. [75] call this type of approach Constrained Heuristic Search (CHS) as opposed to constraint directed search, and also describe a factory scheduling system in which parts of the problem are restructured as a constraint graph. The scheduling strategy embodied in both constraint directed and constrained heuristic approaches varies from essentially a "fixed" predictive strategy, for example as in ISIS, to an opportunistic reactive strategy, for example as in OPIS and DAS.

## 4.2.KNOWLEDGE REPRESENTATION

To produce a good feasible schedule requires appropriately representing real system constraints and the knowledge to reason about them. Rich and Knight [76] comment on the role of knowledge representation in problem solving. Real knowledge must be "mapped" into a suitable internal format so that it can be manipulated by the problem solving program. The solution can then be "mapped" back to the format of real knowledge, and output so that it appears to have come from the initial real knowledge. 'If no good mapping can be defined for a problem, then no matter how good the program to solve the problem is, it will not be able to produce answers that correspond to real answers to the problem.' In both dynamic rule-based and heuristic search approaches, one of the key issues addressed is knowledge representation.

In rule-based systems both the knowledge about the constraints and the scheduling rules which take the constraints into account are represented in production rules of the form IF <conditions> THEN <actions>. In a batch process plant the knowledge about the constraints in the sytem must include a

representation of plant item connectivity and how the connections set up between plant items affect their availability for allocation to production activities. This could be represented in a rule-based format. As an example some of the connectivity constraints between two storage vessels (vessel 1 and 2) and two semi-continuous process items (separator 1 and separator 2) might be represented as follows:

IF vessel 1 has an input from separator 1
THEN no input is possible from separator2

IF vessel 1 has an input from separator 2
THEN no input is possible from separator 1

IF separator 1 has an output to vessel 1
THEN no output is possible to vessel 2

A rule-based system would require a large set of complex domain specific rules to represent all the connectivity constraints in a typical batch plant with a large number of plant items each of whose connectivity constraints had to be represented in this way. In this sort of representation there is a danger that gaps could exist, or contradictions could be expressed on the connectivity of plant items. Also any changes to the real plant structure would require careful updating of the rule-base. Therefore, although this would be a feasible approach to take to representing these connectivity constraints it is not a very satisfactory one.

In heuristic approaches the representation of constraints is more structured with the aim of guiding how the system operators which carry out the scheduling function generate new states in the search space. Constraints are represented explicitly through data structures rather than implicitly through rules. Smith et al. [77] describe the partioning of the factors that

influence factory scheduling into two classes of constraints; scheduling restrictions, and scheduling preferences. Scheduling restrictions are further classified into causal restrictions such as precedence, or resource requirements; physical restrictions such as plant unit capabilites; and resource availability restrictions for example machine breakdowns that are outside the schedulers control. Scheduling preferences must be considered to focus the scheduling on good solutions. These preferences are also further classified. There are organisational preferences for goals such as meeting due date or maximising resource utilisation and there are operational preferences which '..express preferred choices at the level of individual scheduling decisions (that is, the selection of specific operations, resources, and time intervals) and reflect the heuristic knowledge present in a given scheduling environment.' [77]. The classification of constraints used varies slightly from system to system. For example Burke and Prosser [73] describe four classes of constraints in their system; precedence, technological, temporal, and preference. They further classify these constraints into non- negotiable (technological and precedence) and negotiable (temporal and preference). Restrictions or non- negotiable constraints cannot be breached, whereas preferential or negotiable constraints can be breached or relaxed in order to generate a feasible schedule.

In a representation of a batch plant if the constraints related to the connectivity of plant items could be represented through data structures then the availability of a plant item to make a connection to another plant item at any time could be inferred through a general procedure. The model of a plant and its constraints could be developed in a more structured way than through using a purely rule-based representation and would not suffer from the dangers and problems described earlier. In order to achieve this representation a suitable data structure is required on which to build the representation of a plant item and the constraints on its use.

## 4.3.FRAMES AND RELATED STRUCTURES

Minsky [78] developed the concept of frames originally to represent stereotyped situations, in order to match against particular situations or scenarios encountered by a system to help it in reasoning about them. In visual scene analysis for example, where there might be a collection of frames which represented the scene from different viewpoints. The key thing about a frame representation is that it provides a compact way to collect and classify data about something whether it is a production activity or a batch plant item and provides a structure that can be manipulated easily. One of the major themes of the work done on ISIS and related systems was knowledge representation using frames. A frame-based language for building complex domain models was developed called Schema Representation Language (SRL) in which frames are called "schema". This has been used in a number of related systems such as OPIS, CALLISTO for project management [79], and the Knowledge-based Simulation system (KBS) [80], and now forms the basis of the knowledge representation language CRL in the knowledge-based system development tool Knowledge Craft.

Fox [67] describes the key features of SRL for building a factory model; 'It provides the structural primitives in which the domain's conceptual entities are defined.' 'On top of this, concepts germane to scheduling are defined, and these are instantiated as a model for a particular plant. The key feature of SRL which allows this layered approach is the user- definable relations. High-level, domain- dependent relations may be constructed from low level, domain independent primitives.' The basic syntax of a schema in SRL is typical of a frame- based system. It is a data structure composed of a collection of slots, which define the attributes of the structure. The attributes defined for a schema perform two functions. They define the characteristics of the specific object itself, and allow it to be related to other schemata in the knowledge base. The ability to use attributes to define relations is one of the most

important and basic features of frame-based systems, because it allows the concept of classification and inheritance to be used through IS- A and INSTANCE- OF relations. SRL also allows domain specific relations to be defined by the user. The ability to define and represent domain specific relations between entities in a scheduling system is an important part of knowledge representation, which enables the representation of some classes of constraints, particularly temporal and precedence constraints, and technological constraints resulting from composite requirements or the structure of the domain. Therefore, a frame-based system with the types of features described for SRL would be a suitable one with which to develop a representation scheme for batch plants in which some of the constraints need to expressed as relations about the ability of plant items to connect to each other.

## 4.4.REPRESENTING RELATIONS, ENTITY CHARACTERISTICS AND PREFERENCES

Relations may specify composite or alternative conditions. For example, in SRL, there are a number of schema, such as AND, OR, and XOR representing composite or alternative states required for some activity to take place. In other systems similar AND/OR structures are used to represent relations between activities or tasks. For example, Canzi et al. [70] represent process plans as an AND/OR graph which shows tasks which must be carried out together and alternative ways of carrying out tasks. AND/ OR graph structures can also be used to represent aspects of the structural relationship between entities. For example, Homem de Mello and Sanderson [81] describe an AND/ OR representation of the feasible sequences that can be used to put a particular assembly together, where the structural relationships of the components enable the assembly to be put together in a number of ways. They say that the AND/ OR graph representation of assembly plans combines all possible assembly sequences in a compact way.

A number of researchers have looked at ways to define temporal and causal relations between actions and states, using point and interval notations. For example Allen [82] has defined a representation of time based on intervals and relations between them such as BEFORE, MEETS, and DURING. Allen commented that this scheme could be used to represent process plans and infer relationships between intervals not explicitly represented in the plan. This interval based representation scheme has been used by a number of researchers for representing process plans in scheduling and related systems. For example, Parthasarathy and Kim [83] have used a hybrid point and interval derivative of the representation in developing a temporal reasoning system for use in a real-time dynamic decision making in an automated manufacturing system. Sathi et al. [79] use it to represent and reason about activities in project planning systems. In ISIS Fox also introduces a number of causality relations as well as the temporal relation to enforce precedence relations between activities. This is because the temporal relations do not explicitly specify that one activity must be performed to some degree before another.

In addition to the use of attributes to specify relations, they are used to specify characteristics of the entity itself. The value or values taken by an attribute at any particular time may be variable but may well be restricted to come from a continuous or discrete range of choices. The restrictions may represent technological or temporal constraints, for example, a set of plant units which could be used to perform a particular task or operation, or a due date on a particular job. SRL approaches this through the use of range-constraint schema which can be attatched to slots and constrain the values that the slot can take. Other systems such as DAS [73] typically represent choices as a list or range of values in a "possible choices" slot, and then have a "mirror" slot which holds the actual current value taken. In general, if an attribute may take a range of values, then depending on circumstances some of these values

may be currently inadmissable due to constraints already present, or if they are admissable, then some values may be more acceptable or preferential than others.

The preference for a particular value or values, may vary depending on circumstances. Preferences are represented in various ways by different researchers. In SRL and ISIS, preferences are determined by a relaxation specification, which may be a set of discrete alternative choices which each have a utility value associated with them, or a rating of how preferrable a continuous value is. In SOJA [69] where choices exist for satisfying constraints in the scheduling phase of its operation, then preferences about these choices are represented as production rules. In DAS [73] operations to be scheduled on a resource are selected on the basis of a predefined strategy such as "most constrained".

These representation schemes are applicable to the batch plant scheduling model for a number of purposes. The connectivity constraints between plant items in a batch plant are based on the structure of the plant and affect how they can be feasibly connected together. A representation scheme to show this must be able to show the alternative ways in which plant items can be put together without breaking the constraints. The AND/ OR structure is appropriate for this purpose and would enable it to be done in a compact way. It is also an appropriate structure for production "recipes" to show alternatives between production activities or the requirement to carry them out together as described above. The scheme for representing temporal relations in process plans as described by Allen [82] is an appropriate one for the representation of precedence relations between production activities in a production recipe. The use of production rules is the most flexible way to represent preferences both at the level of production recipes and activity scheduling, and the allocation of resources to activities.

a consequence of a rule firing and typically form the first condition of one or more other rules. Therefore they can be used to structure the search by restricting the number of rules which could fire on any recognise-act cycle to a set which have the context in their conditions. For example, the system developed by Bruno et al [64] is written in OPS5 which supports the use of contexts and they are used to structure its execution as an activity scanning simulation through a number of tasks, and enforce a partial ordering on the firing of rules concerned with scheduling. In the system described by Ben-Ariah [63] written in PROLOG, rules are matched and fire according to their order in the rule-base, because the basic PROLOG search mechanism operates in a depth first fashion. Therefore, in this case the order of the rules in the rule-base will have a considerable effect on the way that the system operates and will require careful consideration by the system developer. The use of rules can allow a great deal of flexibility in the strategy adopted for scheduling which is determined by the current state of the system. In the approach described by Sackett and Fan [85], determination of which job/ task to schedule next is based on a series of cases which define what to do in different circumstances, and may specify the use of a particular rule-base. The system is also written in PROLOG, and the importance of cases is reflected by their order in the database. Because the scheduling strategy is embodied in the rules of the system it can be altered simply by adding or deleting rules. This is easier if the system is based on a full expert system structure, because the order of the rules would not then matter.

One of the reservations about the use of rule-based systems for scheduling is the inability to look at the entire time horizon at once in order to assess the global effect of a local decision based on the current system state. To overcome this, and also to allow reasoning at a higher level of abstraction algorithmic procedures may be incorporated into the structure of the expert system. Wu and Wysk [86] describe a number of features which they say should be incorporated into an expert system for factory control and scheduling

including 'An interface for algorithmic procedures, which allows the integration of analytical models to an ES.' and 'A look- ahead mechanism (for example a simulation model) to conduct "what- if" analysis. The system of Ben- Ariah [63] includes a dynamic routing algorithm written in PASCAL. The system described by Bruno et al [64] is for scheduling an FMS and uses two analytical procedures in the decision making process to determine lot priority (defined as time/ (due date - release date)) and maintain the system loading within predefined capacity limits. In order to ensure that the introduction of a lot to the system as determined by the application of the scheduling rules will not break predefined capacity constraints, it uses a queueing network algorithm to estimate the performance of the FMS if the lot were introduced.

In heuristic search approaches, the schedule is developed by generating states which represent feasible allocations of jobs/ tasks to resources over the whole scheduling time horizon, and rating these states against the scheduling objectives for the system. In a number of systems a hierachical decomposition of the scheduling task is done to direct the search, and limit the number of states in the search space that have to be generated. For example, the ISIS architecture has a four level hierarchy as described by Smith [77]. It takes an order based scheduling perspective, and the first level prioritises all orders for the scheduling horizon on the basis of a priority class and the closeness of the order due date. Scheduling then proceeds on an order by order basis. At the second level, a critical resource capacity analysis is carried out to propagate the temporal consequences of the requested start and due dates assigned to a selected order and results in a coarse schedule. At the third level, resource analysis, the full range of restrictive and preferential constraints that surround the production of the current order are considered. Operating over the set of possible routings for the order '..a heuristic search is performed that proceeds either forwards from the order's requested start date or backward from it's requested due date. Alternative schedules for the order are explored

incrementally: On each iteration of the search, the current set of candidate partial schedules is expanded by considering one additional scheduling decision..' 'Using a beam search, ISIS retains and extends only the n best partial schedules.' Constraints are collected and applied '..to assess how well each candidate satisfies relevant preferences; this method provides the basis for pruning.' At the completion of the search, the highest rated schedule for an order is passed down to the final level together with corresponding constraints. The final level, resource assignment, makes the final allocation decisions for each resource in the schedule of the order within the time-bounds derived from the previous level. The decisions resulting from the scheduling of this order are added to the overall shop schedule and will constrain the scheduling of subsequent orders.

In contrast to rule- based approaches the scheduling strategy in heuristic search approaches is "embedded" in the system and may or may not be adjustable by the user. For example the strategy in ISIS was essentially fixed on a single order based scheduling perspective, so OPIS [68] extended this approach to consider two scheduling perspectives simultaneously, resources and orders, because it was found that ISIS could not handle conflict for resources very well. In OPIS, the order in which scheduling decisions are made is not fixed in advance, but is determined dynamically according to '..the structure of the constraints implied by the current solution state.'. This is known as an opportunistic approach to problem solving and is the strategy adopted in other systems such as DAS. OPIS uses a blackboard architecture, and a number of knowledge sources specific to particular sub- problems identified by the system manager. In the context of generating a schedule, a capacity analysis knowledge source is used to identify bottleneck resources. Schedules of operations are first generated for these resources, using an iterative dispatch based approach. A shift to an order based perspective is then made, to determine the remainder of the schedule around the already scheduled resources. In DAS as described by Burke and Prosser [73] the user

has some more control over the scheduling strategy. By assigning priorities to the objects in the system, the scheduling effort can be focussed, for example, onto critical resources and critical operations.

## 4.7.KNOWLEDGE MAINTENANCE

If the system is scheduling on the basis of a model, then the consequences of any local decision or event must be propagated through the model. The consequences could either result in a change in system state, or change the bounds of constraints on entities affecting the way that scheduling requirements could be satisfied. This is discussed by Burke and Prosser, [73]; 'It is not uncommon for constraints (scheduling decisions and significant events) added in this way to have implications for the global hypothesis which are not fully represented within the constraints themselves.' Propagation ensures that local decisions can be focussed on with the assurance that global consistency will be maintained. In a batch plant, a local decision to use a plant item for a particular activity could affect the availability of other plant items because of the structure of the plant network. Therefore, for a scheduling application using a model of a plant network a method of propagating these effects is necessary.

Researchers in the area of Constraint Satisfaction Problems (CSP) have developed structures for representing groups of variables in a constraint graph where the nodes represent variables to which a value is to be assigned from some applicable range. Each individual node is subject to unary constraints on the value it can be assigned. In addition the nodes are connected together by arcs which represent relations between them and further constrain the values that they can take. Whenever a node is assigned a value the effects must be propagated to some or all of the other nodes in the network. Researchers have developed constraint propagation algorithms for updating the status of the system as a result of a particular decision. For

example Mackworth [87] developed algorithms for maintaining consistency of a network subject to unary and binary constraints, and Allen [82] developed an algorithm for propagating the effects of changes in the status of relations in a network of temporal intervals. The problem with constraint propagation algorithms is that they can do a lot of redundant work, which can lead to inefficiency in large networks. In the most basic case, as described by Mackworth, every time a change is made to a network, then propagation must be repeated from the start of the network because of any possible consequences of this new change on other parts of the network. However, in knowledge-based scheduling applications where the parts of problem representation can be formulated as a network of nodes linked by relations, constraint propagation algorithms have been found very useful to update the status of the model. This approach is usually applied to update the temporal relations among activities in production processes. Parthasarthy and Kim [83] describe the use of constraint propagation algorithms for updating a network of temporal intervals used for real- time control. LePape and Smith [84] describe the propagation of changes up, down and across hierarchies of knowledge at different levels of abstraction in the OPIS system. In a purely rule- based system, the consequences of a rule firing may be propagated through other rules. In a mixed rule/ frame or purely frame-based system, then daemons attached to a particular entity slot which is updated may be set up to propagate the effects to other entities in the system. This mechanism is used in CHRONOS II [70] and the Constraint Maintenance System (CMS) in DAS [73] in which the frames in the system represent the nodes in the network.

## 4.8.KNOWLEDGE-BASED CONFIGURATION OF INTERCONNECTED SYSTEMS

The rule- based and heuristic search approaches described in this chapter tackle a number of issues such as the classification and representation of the

real constraints in a manufacturing system, and reasoning about the temporal relations between process actvities. Scheduling in a batch process plant is subject to an additional constraint that is not found in discrete part manufacturing environments addressed by these approaches. A typical batch process plant is an interconnected network of plant items and finite capacity storage vessels. The valve arrangements in the network enable different configurations of plant items to be linked together to route product through the plant and carry out particular processes. In this respect routing within a batch process plant has characteristics which make it very similar to the generic configuration task as described by Mittal and Freyman [88]:

> 'Given: (A) a fixed pre- defined set of components, where a component is described by a set of properties, ports for connecting it to other components, constraints at each port that describe the components that can be connected at that port, and other structural constraints (B) some description of the desired configuration; and (C) possibly some criteria for making optimal selections.
> Build: One or more configurations that satisfy all the requirements, where a configuration is a set of components and a description of the connections between the components in the set,..'

In a batch process plant the situation is also further complicated by the fact that resources available for configuring routes are strictly limited, and the level of availability changes dynamically, because of changes in the set of routes which are running at any particular time. Also, because of the valve arrangements, even if a plant item is not in use it may still be unavailable to be used in a particular route configuration. This is one of the key issues which must be addressed in developing a short term scheduler for batch process plants.

Configuration is a search problem, and a number of approaches to the problem in general have been described, although they normally ignore the problem of restricted levels of resources. The most well known example is the R1/XCON system for configuring Vax computer systems, [89]. This is a rules- based approach, which works on the basis that it is possible to dynamically define a decision order for a given configuration problem such that each decision brings the system closer to a successful solution with no requirement to backtrack. Some other researchers define the problem as a CSP. Havens and Rehfuss, [90] describe a constraint based reasoning system for configuration applications which comprises four components in its architecture: a model based reasoning representation, a rule processor engine, a constraint propagator, and a truth maintenance system. They describe the configuration process as a search which continually constrains the search space. 'Search and constraint propagation form a positive feedback system... The choice of an hypothesis provides constraints on the search space which help choose another hypothesis and so on.' Crone and Julich, [91] describe a system for dynamically reconfiguring communications networks under varying conditions. Each node in the constraint graph represents a particular object class instance with associated variables, and as the search proceeds and variables become instantiated, the effects are propagated through the network. A recent paper by Sathi et al [92] tackles the problem of configuring a number of artifacts when resources are limited and can be used across more than one artifact configuration. This causes problems because '..supplying one configuration can affect an inventory managers ability to supply other configurations.'. They use a constrained heuristic search approach to tackle the configuration and resource allocation problem at the same time. The ways in which artifacts can be configured are decomposed into a number of levels. In general, each level represents a number of choices, so the decompostion is represented as an AND/ OR tree. The topology is traversed from top to bottom, and for a given artifact generates a constraint graph representing the feasible ways that it can be constructed. After all feasible configurations of a product

have been determined, a resource allocator is used to assign quantities of components to each feasible configuration based on component inventory and predefined quantity and composition constraints, where alternative components exist.

## 4.9.DISCUSSION

There are two main approaches to scheduling from the area of AI research, the use of rule-based systems which effectively carry out dynamic scheduling based on the current system state, and the use of search based techniques which attempt to look at the whole time horizon during the scheduling process. In both cases, the aim is meet scheduling goals while satisfying a set of constraints which are either fixed, or can be relaxed to some degree. The rule-based approach effectively adopts the same strategy as simulation based approaches, and may be structured as a simulation model as done by Bruno et al. [64]. However, the use of an expert system development tool such as OPS5, or a declarative language offers considerably more flexibility in representing control rules than a traditional simulation language. In addition, there are several aspects of the research into knowledge representation, reasoning about constraints, constraint satisfaction, and configuration systems which are directly applicable to the modelling and scheduling of batch process plants.

Batch process plants are typically complex highly interconnected structures, so for scheduling and control purposes it is important that they are represented at the correct level of detail. The importance of representing the problem at the right level of detail is stressed by researchers using AI techniques for scheduling, and powerful representational languages have been developed for this purpose, for example SRL [67]. The use of a frame-based system in which slot values can represent relations between objects, enables semantic networks of complex data structures to be built to accurately model

the physical and control structures of a manufacturing system. Representing the constraints which exist on possible configurations of batch plant items is particularly important. Homem de Mello and Sanderson [81] discussed the use of AND/ OR structures for representing assembly plans, saying that they represented all possible assembly sequences, but in a compact way. This comment also holds for configuration purposes where alternatives exist for the component parts of the final "artifact", for example a process route in a batch plant, where not all components are compatible with all other components but can all form part of a configuration. It has been demonstrated that alternative configurations of artifacts can effectively be represented through the use of AND/ OR structures by Sathi et al [92], so this structure has the potential to be a suitable for the representation of plant item connectivity in a batch plant.

Production in batch process plants is controlled by "recipes" of activities which may have to be carried out in sequence, or may be able to proceed in parallel. Therefore, a scheduling system must be able to represent these "recipes" correctly, and be able to reason about how to proceed with the activities. AI researchers have developed representation schemes for temporal relations between time intervals, which can be used to build models of control structures, and reason about precedence and causal relations in process plans.

Batch plants are highly interconnected, so the results of a scheduling decision in one part of the network tend to affect other parts of the network as well, for example making resources unavailable even though they are not currently in use in an activity. The research into the area of constraint satisfaction has resulted in the development of constraint propagation algorithms which enable the effects of decisions to reach all parts of the system that are affected by them, ensuring that the model remains globally consistent.

Correctly configuring the plant to route and process product through it is an important aspect of the scheduling and control process. The configuration

problem in batch plants appears to match the generic configuration task as described by Mittal and Freyman [88]. It has been demonstrated that a rule-based approach can be used successfully for configuration purposes because it enables a situation dependent decision order to be imposed which moves the system forward to a solution as described by McDermott [89]. In batch process plants, the decisions concerning configuration of routes are typically situation dependent and, because resources are limited and interconnected, each decision has a constraining effect on the remaining choices. Therefore, in a batch plant both the decision order and constraints imposed by the decisions should be taken into account during the configuration process. Havens and Rehfuss [90] have demonstrated the use of constraint propagation to account for the constraining effects of decisions made during a configuration process.

# CHAPTER 5 HYBRID KNOWLEDGE-BASED SIMULATION

## 5.0.INTRODUCTION

In Chapter 3 it was argued that a flexible generic modelling architecture was required for the development of simulation models which could address the scheduling of real batch process plants. Appropriate knowledge representation schemes and reasoning schemes from AI based approaches to modelling, short term scheduling and configuration were discussed in Chapter 4. This chapter will discuss how simulation can benefit from being combined with AI modelling approaches to produce a suitable hybrid environment for scheduling and control.

## 5.1.A COMPARISON OF SIMULATION AND AI BASED MODELS

The comparison between AI and simulation modelling has been made by a number of researchers including Vaucher [93], Shannon et al.[94], O'Keefe [95], Doukidis [96], and Hurrion [97]. The comparison made generally refers to the area of AI concerned with rule- based or expert systems. However, there are other areas where a comparison can also be made, for example with AI planning systems as described by Rich [76] which use operators very similar to the production rules in knowledge-based systems. Simulation and AI can be compared from the point of view of their use, their knowledge representation, and the structure and execution mechanism of simulation and AI programs.

### 5.1.1.The Use of Simulation and AI

Vaucher [93] discusses differences in the use and operation of simulation and rule- based programs and comments that although they are not trying to achieve the same thing the two approaches have much in common in that

'..progress is achieved via a series of transitions subject to preconditions, events in one case and deductions in the other.'

The purpose of running a simulation experiment is to predict what the future will look like, or what levels of output can be expected from a system, or what trends are apparent in the behaviour of the system. A simulation starts from a set of initial conditions, and using a particular set of operating rules moves forward along a single path through it's state space until it reaches a point in time or an event where it is terminated. The results collected from the experiment can be analysed to make inferences about the behaviour of the system. The purpose of a running a rule-based AI program is to solve a particular problem with a specified goal, for example to determine the cause of a set of observed conditions in a diagnostic system. It attempts to solve the problem by a search process through it's state space, either forwards from initial conditions to the goal, or backwards from the goal to the initial conditions. During the search it may well backtrack if it can no longer progress along a particular path, whereas (except in a few limited cases) simulations do not backtrack.

Although the use of simulation and rule- based programs appears quite different from this description their knowledge representation and structure and execution mechanisms cover a lot of common ground. They are both concerned with building models of complex systems, specifying relationships between entities which interact in complex ways, and whose behaviour is difficult to analyse. Vaucher [93] discusses knowledge representation saying that only relatively recently AI researchers have concluded that to exhibit intelligent behaviour systems must '..have some knowledge of the properties of the objects they manipulate as well as general common sense world knowledge. This is more than just data; knowledge also involves awareness of attributes, relations, rules of deduction, rules of evolution, etc... Exactly the

sort of things that simulation workers have been integrating into their models, AI needs to put into its reasoning systems.'

### 5.1.2.Knowledge Representation

In a procedural simulation model, the structure and state of the system is represented by the permanent and temporary entities and their attributes. The entities are typically classified according to their general characteristics, and the data about the state of each specific entity in a simulation is represented by a record structure, which lists the values of its class specific static and dynamic attributes. In a knowledge-based system information about the objects within it may be represented by simple facts or predicates, semantic networks which indicate how objects are interrelated, or more complex object, frame, or schema structures. Simulation entity records are very similar to frames in that they list the static and dynamic attributes of specific instances of object classes in a set of data fields or slots. However, frames may also include more complex information than simulation entity records, for example lists of values rather than a single attribute value and they may define relations with other frames.

In a simulation the behaviour and interaction of entities is specified by activity cycles or similar structures. These specifications are translated into procedures in the simulation language to control the progress of an entity through the simulation. Some procedures specify the conditions under which an entity may start an activity, known as a conditional event, and determine the time when that activity will end, known as a scheduled event. At a scheduled event, a procedure specific to that event determines how the state of the entity and the system is changed. In a knowledge-based system, the behaviour and interaction of objects is typically represented through production rules of the form IF <conditions> THEN <actions> or IF <conditions> THEN <goal>, which may alter the current state of the system,

or direct the way that the system is moving. The conditional event and the production rule are very similar, in that both specify a set of preconditions which must be present for the event to occur or the rule to be applied, and both specify what actions should be taken in this case.

It can be seen from this comparison that the concepts such as frames and rules which are being used in AI have existed in simulation for a long time although the terminology is different. However, the declarative language based tools available in AI give more powerful and flexible ways of representing these structures.

5.1.3.Structure and Execution:

A number of researchers such as O'Keefe [95] and Doukidis [96] have commented on the similarity between the structure of an expert system and a simulation, in that both contain a model of the system, rules about the behaviour of the system, and a mechanism for applying the rules to change the state of the system or infer something about it. However, Shannon et al. [94] commented that although the structure is similar a rule-based system is generally more separated and modular, so that the model database, knowledge-base and control structure are separate and each can easily be modified without affecting the others. Many procedural simulation models tend to have integrated information and control logic which makes them difficult to maintain and modify, although this is addressed to an extent by generic models where the model data is separated out from the procedural code.

As discussed by O'Keefe [95] one of the key things that makes simulations and rule-based systems similar is that the control mechanism is independent of the other modules of the system such as rules or events, and can theoretically work with any number of these modules in any order. In the case of a

simulation this mechanism is the executive, and in the case of the expert system this mechanism is the inference engine. In particular, O'Keefe [95] and Doukidis [96] point out the similarity between the way in which the executive of a three phase event simulation and the inference mechanism of an expert system work. The 'C' phase of a three phase simulation and the recognise- act cycle of an expert system both involve scanning a collection of possible rules which could be applied depending on the current state of the model. However, the mechanism for applying the rules in the three phase event simulation is a simple and restricted form of the recognise- act cycle of an expert system as discussed by Doukidis [96]. In an expert system, before a rule is chosen and applied, a conflict set of all possible rules which could be applied is assembled, whereas in the 'C' phase of a simulation, the rules are applied in their order in the rule-base. Therefore, as described by Pidd [36] a simulation developer must be careful to order the 'C' phase events carefully so that conditions in the simulation are tested in the right order. Both Vaucher [93] and O'Keefe [95] comment that although the recognise- act cycle of an expert system is more flexible than a simulation executive it is less efficient. Also, as described by Vaucher [93], the simulation executive, with it's event list, is better suited to synchronising parallel activities to produce global behaviour over time than the inference engine in a rule-based system which cannot handle the complex dynamics of changes in system state over time.

## 5.2.RESTRICTIONS OF PROCEDURAL SIMULATION LANGUAGES

O'Keefe and Roach [35] describe one of the major limitations of procedural simulation languages as '..the inability to model intelligent behaviour.' Simple approximations have to be used instead, such as determining the path of entities through the use of probabilities, or simple decision rules, such as "always join the shortest queue". Marsh and Williams [98] describe other disadvantages of using procedural languages, in particular FORTRAN, for modelling simulations of battles which involve complex tactical decisions. The

properties of individual entities have to be held in separate arrays, depending on their type, so they lack clarity. 'The representation of interactions between entities during events is obscure. It is often difficult to determine and model all the possible outcomes of an event, or to isolate the effects of one type of event upon the movement of entities into successive events...Representation of tactics is difficult. This is because the data structures are unsuited to representing preconditions of tactical conditions, and because FORTRAN code becomes cumbersome when representing complex decision rules.'.

Another limitation of FORTRAN and other procedural languages particularly when building a generic modelling tool is the requirement for the system designer to determine all model class data structures in advance so that they are flexible enough to build models to the right level of detail. A possible consequence from this is that there may be a lot of redundancy in models built with the system. There may be cases where the structures are not flexible enough to incorporate the desired level of detail. For example, difficulty in modelling the number of connections that can be made from a plant item may limit the size and complexity of batch plants which can be modelled. O'Keefe and Haddock [37] cite limitations in the level of detail which can be modelled as a reason why procedural generic simulation systems are not suitable for scheduling purposes.

## 5.3.KNOWLEDGE-BASED SIMULATION

The recognition that simulation modelling and work in some areas of Artificial Intelligence (AI) share the common goal of seeking to model some aspect of the real world has led to researchers from both fields combining techniques which will benefit them in their aims. Areas covered by researchers in simulation include intelligent aids to parts of the simulation study cycle, such as model building using a natural language interface described by Paul [99] and Ford [100], statistical analysis of simulation results described by Haddock [101],

and the development of goal- directed simulation systems described by Prakash and Shannon [102]. Reddy et al. [80] describe a large number of facilities which should be incorporated in an "ideal simulation environment" such as '..declarative model representation, behavioural representation of system entities through an object- oriented (frame- based) knowledge representation that lets entities be altered without altering the simulation model interpreter', and '..expression of events as rules to make models more readable'.

The desire to incorporate complex decision making into the simulation control process led to the development of VIS systems. Research into knowledge-based simulation systems is a step forward from the use of VIS to achieve this by incorporating this decision making directly in the model and overcoming the limitations of procedural simulation languages in this respect. The requirement to accurately represent system structure at the right level of abstraction is an equally important factor in the use of simulation for real world control applications which is difficult in procedural simulation languages, and can be more easily accomplished using knowledge-based techniques.

### 5.3.1.Approaches to Knowledge-based Simulation

The use of applicative languages such as Lisp or declarative languages such as PROLOG for the construction of simulation models to give more flexibility for knowledge representation, and the execution of the model, has been the main way put forward to overcome the restrictions of procedural simulation languages. There are a number of examples of simulation systems developed commercially using Lisp including the Rule Oriented Simulation System (ROSS) [103], the Knowledge-based Simulation (KBS) (which is part of the Knowledge Craft environment) [80], and SIMKIT [104]. They are described as object-oriented modelling systems, and the object-oriented approach is

probably the most common one taken in developing knowledge-based simulation models. The language Smalltalk- 80 [104], which was developed specifically for object- oriented programming has also been used for building simulation models including a generic visual interactive batch process modelling system developed by Vaessen [105], a model for scheduling dairy packing lines described by Alasuvanto et al. [106], and a graphical simulation program generator for material handling systems developed by Thomasma et al. [107]. In a completely object-oriented system modelling system such as Smalltalk-80 all knowledge about the objects in the simulation is encapsulated in the objects themselves in the form of properties and methods or behaviours. Properties represent the static and dynamic data detailing the object. Methods are expressed as rules or procedures which represent the behaviour of the object. All the behaviour of the objects in the system is governed by message passing from one object to one or more other objects to either update data slots, or activate methods. The Smalltalk- 80 environment includes a predefined simulation object to control the execution of a simulation model using a process interaction approach.

The other main approach to developing knowledge-based simulations stems from the comparison of expert system and simulation structure, so that the expert system structure is integrated with the simulation structure to produce an expert simulation system. Some researchers have taken the basic inference engines of expert system tools and added a time advancement mechanism in order to achieve an expert simulation structure. Probably the most well known example of this is the system of Bruno et al [64] described in Chapter 4. Other rule-based simulation systems described by Robertson [108] and Shivnan and Browne [109] also use an OPS based inference engine. In this approach the static and dynamic data about the system entities is separate from the rules concerned with modelling their behaviour, and the data representation format can range from simple facts or predicates to frames or schema. Shivnan and Browne [109] comment that the best knowledge

representation scheme in a system of this type for production activity control would be a hybrid one of frames for domain knowledge and rules for heuristic control knowledge.

Some researchers have attempted to alter the way that simulation models execute. Futo and Gergely [110] have developed a system called TS- PROLOG, which allows the simulation to backtrack so that it can be given goals to work towards and try different ways of achieving them. In the system described by Robertson [108] the objects in the simulation have an agenda of goals rather than a process description, and the rules in the system are applied to determine how the objects should behave to achieve these goals in relation to the current system state. Cleary, Goh and Unger [111] have developed a simulation system in Concurrent PROLOG (CP) called T- CP which runs by attempting to move all entities through their processes defined as procedures concurrently, rather than sequentially as in standard PROLOG.

### 5.3.2.Advantages of Knowledge-based Simulation Approaches

The main advantage of the knowledge-based approach to simulation comes from the flexibility for representing data structures and rules through using a declarative language. Fan and Sackett [112] say that a language such as PROLOG frees the system developer to a large extent from the constraints on the size and complexity of data structures and behaviour rules which can be developed. PROLOG enables complex interconnected data stuctures to be represented flexibly and easily, for example Vaucher [113] demonstrates how a network structure can be easily represented as a set of simple PROLOG facts representing the arcs and their node connections. Asfahl and Balagamwala [114] describe the use of PROLOG for modelling and simulation of complex logic circuits saying 'PROLOG has been found to be capable of modelling all structural elements (gates)... interconnected in any fashion in any logic circuit.' Haddock and O'Keefe [115] describe the ability to include

complex decision making rules in the PROSS simulation system, a process description network simulation system developed in a PROLOG dialect. 'The scheduler is modelled as a process description with a number of associated rules that model the scheduling heuristics' 'If the scheduling heuristics changed, or the scheduling constraints changed... then only the rules need changing. The process descriptions are still valid models of material behavior'. Ruiz-Meyer and Talavage [116] describe the inclusion of complex decision rules in simulation objects as the main benefit in the SIMYON simulation system, an object- oriented modelling system developed in a PROLOG like language. The use of rules and properties in an object which allow it to keep track of its own history enables the modelling of adaptive behaviour. Ruiz-Meyer and Talavage [116] give an example of this approach for routing of an AGV through a manufacturing system given a message to pick up a part from a particular machine.

The object-oriented approach is the one most taken in developing a knowledge-based simulation and Rothenburg [117] describes it's advantages. It 'provides a rich, lucid modelling paradigm whose strength lies in it's ability to represent objects and their behaviours and interactions in a cogent form that can be designed, compehended, and modified by domain experts and analysts far more effectively than than with previous approaches'. Irrelevant details of the modelling of objects are hidden from the user, and the concept of encapsulation associates the behaviour of an entity with its state definition. It allows the modelling of certain real world objects, such as vehicles, in a natural way. 'Similarly, it provides a natural way of modelling static, taxonomic relationships among objects by the use of sub- class hierachies, while minimising the redundancy (and possible inconsistency) of their definitions through the inheritance of attributes and behaviours over these hierarchies.' He says that message passing also provides a natural way of modelling the dynamic interactions between some kinds of real world entities. Objects provide comparable features to frames when looked at as a way of

data representation, and the two terms can be used interchangeablely in this respect. However, the message passing approach of object-oriented programming is not ideal for a scheduling simulation as will be described in the next section.

### 5.3.3.Problems with Declarative Languages and Knowledge-based Approaches

For the most part, except for the simple backtracking and parallel simulation systems developed in PROLOG, the development of simulation systems in declarative languages has been driven by the desire to increase the flexibility of knowledge representation and control rules, rather than to significantly change the way that the simulation executes. Therefore, the procedural simulation executive is simply duplicated in the declarative language of the knowledge-based system. For example, there are many examples of simulation systems developed for research purposes in PROLOG reproducing one of the "world view" executives, typically the process interaction approach as described by Haddock and O'Keefe [115] and Vaucher [113]. Ahmad [118] has developed a system for automatic generation of models using the process view or the three phase view which provides the same basic simulation procedures as the FORTRAN based Micro- Vision system. However, there are problems with using a declarative language to develop a simulation. Fan and Sackett [112] comment on some disadvantages of using PROLOG including the requirement to use awkward constructs for some procedural tasks, and the slowness of long numeric computations. They say that 'PROLOG is definitely not the language for continuous simulation where differential equations are involved.' Shannon [119] indicates speed of execution as the primary disadvantage of knowledge-based simulation, and Reddy et al. [80] commented that KBS ran very slowly when used to model a complex manufacturing system. Therefore the use of a purely declarative approach is unlikely to be practical for an application such as scheduling where the model is likely to be complex and speed of execution is important.

Restrictions with pure object-oriented approaches to simulation are discussed by Rothenburg [117] and Radiya and Sargent [120]. The main drawback for a scheduling application is the essentially local and reactive behaviour of the simulation entities, which is governed by receiving stimuli in the form of messages from other objects, rather than taking into account the global system state. As described in Chapter 3 decision making based on global system state is a desirable feature of a scheduling simulation, so a pure object-oriented approach would not be the best approach to take in this respect. Radiya and Sargent [120] describe the integration of a rules-based approach with an object-oriented approach to overcome this.

There are also difficulties with the use of expert system tools to develop simulations. Walker et al. [121] comment on difficulties with simply augmenting a expert system development tool such as OPS5 with a time advancement mechanism. '..the production rules used within many KBS applications are not ideally suited to the representation of complex domain models and the associated model manipulation operators. This is a particular problem in situations where the application of individual operators has effects which must be propagated throughout different parts of the domain model. The number of individual production rules required to implement procedures for applying operators and propagating their effects throughout the domain model can lead to large and unmanageable domain models.' Nadoli et al. [122] comment that the inference mechanism of a rule- based system may result in asynchronous state changes in a simulation. The event scheduling mechanism of a procedural simulation language is better at handling the co-ordination of interacting entities than the inference engine of an expert system.

## 5.4.MIXED ARCHITECTURE HYBRID SIMULATION

Haddock and O'Keefe [115] and Ruiz-Meyer and Tavalage [116] describe their systems as hybrid because the control rules and data structures have been separated out from the procedural simulation logic. However, the procedural logic is still implemented in a declarative language, with the potential associated problems as described above. Some of the main procedural tasks in a simulation are carried out by the executive. The executive has to maintain a time ordered list of future events in the simulation which is subject to constant updating through adding new events and deleting events which have occurred. The procedural simulation languages are well proven in this respect, using linked- list, and binary tree techniques as described by Marsh and Williams [98]. They also provide good facilities for other procedural tasks such as numerical computation, and for updating graphic displays of the simulation progress. Therefore, rather than recode the whole simulation in a declarative language, the main benefits of the declarative approach can be obtained through a mixed language hybrid architecture, where the advantages of the declarative language are used for knowledge representation and complex control logic, and the simulation executive and basic system behaviour is retained in the procedural language.

O'Keefe [95] discusses how expert systems could be integrated with a simulation model and lists four basic structures with the expert system containing some or all of the decision logic for the model. This provides a suitable structure for a mixed architecture model, and a number of researchers such as Moreira da Silva and Bastos [41], Flitman [123], Bhattacharyya, Roy and Huang [42] Marsh and Williams [98] and Walker et al. [121] have followed this approach in developing or proposing mixed language architectures.

Flitman [123] has developed a link between an expert system component developed in PROLOG and a procedural simulation model developed using the FORTRAN simulation library Micro- Vision. The configuration of the system was developed on two machines connected through an RS- 232 cable in order to ensure '..that the full capacities of the simulation, and in particular the PROLOG expert system were maintained..' through being completely separate programs. The communication link between the two programs was facilitated through a number of assembly language predicates/ routines linked to the respective high level languages, and providing a common data area and a handshake protocol to synchronise the data transfer between the two machines. The feasibility of the mixed architecture was demonstrated in a number of areas such as expert system development through monitoring interactive users of the simulation, controlling the simulation through parameter adjustment, and the incorporation of simulation control logic into the expert system to control the routing of AGVs through a model of an FMS.

Marsh and Williams [98] used a hybrid approach to develop simulations of battle scenarios containing psuedo- continuous processes. PROLOG was used for the logical and static structural data sections of simulations, and FORTRAN for the numerical/ algorithmic and dynamic data sections. It was concluded that this hybrid architecture gave a number of benefits for developing simulations in this environment. The advantages arising from the PROLOG part of the architecture were stated as '..rich data structures are possible (not just arrays), hence enhancing clarity;' '..the dynamic data-structuring capability of PROLOG makes modifications easier;' and '..the unification algorithm makes it possible for one piece of code to handle many different data structures, thus simplifying the code.'. The advantages arising from using FORTRAN for the dynamic part of the model included '..a substantial amount of existing FORTRAN code can be retained;' '..the speed of FORTRAN in performing the frequent environment data updates is not lost' and '..it is possible to add psuedo- continuous simulation processes without

losing efficiency.'. The architecture was assessed in terms of the clarity of mixed programs, the ease with which they could be written or modified, the performance of a mixed architecture program relative to an equivalent all FORTRAN program, and implementation issues covering hardware and software. It was concluded that the use of PROLOG improved the clarity of programs making them easier to develop and modify, and using it to control the simulation improved the structure of the FORTRAN code. The performance of the architecture was slower with the degree of degradation depending to a large extent on the complexity of event logic and run control coded in PROLOG. The complexity of the tactics modelled in PROLOG also had a small effect on performance, although the benefits in terms of clarity for their representation outweighed this drawback. As far as implementation was concerned it was concluded that suitable hardware/ software configurations are now available to achieve satisfactory performance from the PROLOG portion of the models.

5.4.1.Difficulties with Developing a Mixed Architecture:

There are certain difficulties associated with adopting a mixed architecture hybrid approach. Round [104] states 'There are two requirements for using an existing numerical simulator with a knowledge-based component. First, the outputs of one component must be compatible with the inputs of the other component.' 'Second, the numerical simulator and the knowledge- based components must be compiled and linked together into an executable form.' Marsh and Williams [98] comment that the dynamic environment data must remain '..resident in memory throughout a simulation run: the values must not be lost when control returns from FORTRAN to PROLOG'.

To make the outputs of one component compatible with the inputs of the other component may require overcoming a number of difficulties as discussed by Flitman [123] and Borchardt [124]. These difficulties include accounting for

different data structures in the languages used, different methods of program execution (interpreted or compiled), different variable types between the two languages, and the same data types but different internal representations. In the approach taken by Flitman [123] the duplication of integer and atom data types from PROLOG in FORTRAN was not seen as a problem, but replicating PROLOG list data structures in FORTRAN proved more difficult. PROLOG lists can contain mixed data types so they had to be represented in FORTRAN using separate arrays with pointers between the list elements. Borchardt [124] discusses an approach which attempts to overcome the necessity to duplicate data between a symbolic and non- symbolic language. He describes the symbolic language STAR, which uses data structure which can contain pointers to data and procedural functions in non- symbolic languages.

The configuration of two machines used by Flitman [123] overcame the problem of compiling and linking the numerical simulator with the knowledge-based component. A similar solution can be used in a multitasking environment. Lin and Yang [125] describe communication using a virtual communications port set up as a common file between an OPS5 application and FORTRAN with the handshake being co- ordinated through this file instead of via registers in the serial port control chips of two linked machines. These types of configuration also overcome any problems with potential loss of data, because each program will always have its dynamic data resident in memory.

The use of a handshake protocol will ensure that the transfer of a piece of data between the two programs is synchronised whether they are interpreted or compiled. However, a declarative language such as PROLOG has a very different mode of execution from a procedural language because it's mechanism includes automatic backtracking in order to try and prove goals. Thus another difficulty to be overcome is ensuring that the PROLOG part of the system does not backtrack and re- attempt a data transfer which has

already been carried out, so that the overall execution of the two programs remains properly co- ordinated.

It must be determined where to split the model between the declarative language and the procedural language. Marsh and Williams [98] discuss the "boundary" in an event based model with psuedo- continuous components. The boundary was put between the event logic and the numerical computation procedures to ensure that the PROLOG component was not involved in a large number of small increment time steps which would lead to unacceptably high run- times.

## 5.5.DISCUSSION

As discussed in Chapter 3, the incorporation of global decision rules and a flexible data representation is an important feature in improving the value of simulation for scheduling and control. The use of a declarative language such as PROLOG to build a generic simulation tool is therefore a way in which this can be acheived. However, the use of a declarative language such as PROLOG to develop the procedural event scheduling and activity co-ordination part of the simulation will not give any advantages in terms of the way in which the simulation executes, but it may well give some disadvantages through slowness of execution and requiring awkward programming constructs. The use of an expert system tool such as OPS5 also presents difficulties because of the lack of facilities for synchronising parallel activities in inference engines. In addition procedural simulation languages and tools have been undergoing development for many years and can offer many facilities such as graphics and interactive capabilities. There would be little benefit to be gained from re-programming a simpler version of these features in a declarative language.

The use of a mixed PROLOG/ FORTRAN hybrid architecture approach for simulation has been shown to be feasible by Flitman [123] and Marsh [98]

among others, and it overcomes the difficulties described above associated with using declarative languages or experts system tools on their own for developing knowledge-based simulation systems. The part of the simulation which contains the complex control logic can be transferred to the knowledge-based portion of the system, and the application of rules can be controlled by a inference engine. The basic behavioural part of the model can be retained in the procedural language with a simpler but more efficient simulation executive which will maintain the state of the system correctly. The benefits of frame-based programming for flexibility in structuring of data can be obtained through holding the model database in the declarative part of the model. In terms of scheduling and control using a generic system this architecture will overcome the reservations put forward by O'Keefe and Haddock [37] on the specific level of detail which is required for this type of application because it will allow the necessary flexibility in modelling the structure of a particular system and its control rules.

# CHAPTER 6 CURRENT APPROACHES TO SIMULATION AND AI MODELLING OF BATCH PROCESS PLANTS

## 6.0.INTRODUCTION

In Chapter 2 the representation of batch process plants used in analytical approaches to developing short term schedules was described. In order for these approaches to work, a number of simplifying assumptions had to be made about the plant network representation and other features such as product transfer times and storage capacity. It has been argued that the use of simulation and AI based tools can enable these simplifications to be overcome so that schedules produced and controlled using systems based on these tools would be feasible and could be implemented in the plant.

This chapter will review the documented approaches which use simulation and AI tools specifically for modelling batch process plants and how they currently address the features which should be present in a hybrid generic system which can be used for short term scheduling and control purposes.

## 6.1.THE USE OF SIMULATION AND AI TOOLS FOR MODELLING BATCH PROCESS PLANTS

The use of simulation for the study of batch process plant operations is by no means a new phenomenon. An early example of it's use is described by Youle [126], who built a simulation model to study the operation of a hypothetical multiproduct, multistage batch plant. It used the Activity Scanning (AS) approach, and recognised the activities that batch process plant units would typically go through, such as being charged from a previous stage unit, processing, waiting empty, waiting full, and discharging to a unit at the next stage. The plant modelled was of a fairly complex configuration, consisting of seventeen vessels in four stages. The discharge of a reactor at one stage to a

reactor at another stage was handled by a specific transfer routine for each stage involved in the transfer in order to account for '..restrictions due to plant layout.' Since that time a number of researchers have developed generic modelling tools specific to batch process plants for example Joglekar and Reklaitis [127], Vaessen [105], Ready, Simmonds and Taunton [128] and Roberts et al. [129] who approach the modelling of specific class features in a number of ways. The initial example of a batch plant model by Youle [126] illustrates some of the specific class features of batch process plants such as plant unit activities and the configuration of the plant, that need to be accounted for in a generic simulation system for scheduling and batch management. Features which need to be included in a system of this type are:

1.Plant unit activities.
2.Plant layout and network structure.
3.Production recipes
4.Product representation.
5.Plant scheduling and configuration.
6.Product transfer and material balance.
7.Cleaning in Place (CIP) systems.

## 6.2.MODELLING OF UNIT ACTIVITY

Any generic simulation system for a batch process plant must model the activities of the basic system components at a certain level of detail. If the results of the simulation are to be used for scheduling and batch management of the plant, then its activities should "map" onto the activities of the real plant items at the relevant level of detail. For example as discussed by Cott and Machietto [5], a batch reaction carried out in a reactor vessel will include a number of "phases" such as Fill Reactor, Carry out Reaction, Empty Reactor. These phases will naturally be reflected in the activities of the simulation. The low level process control, such as the sequencing of valve and pump operations

associated with each of these phases does not need to be represented at the level of plant control concerned with batch management. It is implicit that each activity requires a number of low level control operations to happen. The plant activities must be modelled at the right level of abstraction for efficient execution without sacrificing accuracy of output.

### 6.2.1.Continuous Activities and Discrete Events

Some activities may need to be modelled as continuous processes if the value of a variable associated with the process changes continuously over time. For example, a storage vessel may be in use both filling and emptying, and the level of product in this vessel may be changing over time. There will also be discrete events associated with this activity, for example when the vessel becomes full there must be a "stop filling" event. The state of a batch of product undergoing some process in a reaction vessel will be changing continuously over time, and there may be a discrete event which occurs when it reaches a threshold such as a particular temperature. Other types of discrete events such as the arrival of vehicles into the system can be scheduled by sampling from a distribution, or by using some real deterministic data in a scheduling application. Roth [130] said that the overall mix of discrete events and continuous activities in a batch process plant implies the need for a combined continuous/ discrete event approach rather than a straightforward DES approach.

A few general purpose simulation languages such as SIMAN [131] include facilities for carrying out combined simulation through the use of state variables which are used to represent continuously changing values over time. The user can define state equations, and differential equations for the derivatives of state variables which are integrated over the course of the simulation to update the values of the state variables. Threshold levels can be set by the user for these variables to model the occurrence of discontinuities

such as a storage vessel becoming empty. If a state variable value crosses a threshold level in a particular direction during the course of the simulation, it will be detected and can be treated as an event with appropriate action being taken. A number of researchers describe the use of simulation languages with these facilities to model batch process plants, for example Knopf [24] and Felder [132].

The continuous facilities are used to model the flow of product through a plant. However, the use of state variables requires that the simulation executive must do a lot more work than a straightforward DES, because the integration routines require that the simulation time is moved forward in small increments, and the values of the state variables are updated at every step. The simulation system BOSS by Joglekar and Reklaitis [127], (now renamed BATCHES), attempts to address the problem by only calculating "active" state variable values as the simulation progresses rather than updating all the defined values each time. In addition, as described by Joglekar, determining the actual time that the threshold level for a state event was crossed during the integration time increment may require the use of an iterative backtracking procedure if the threshold tolerance is exceeded, and if more than one state event occurs during the time increment then they have to be correctly ordered as well. However, if it is accepted that the internal dynamics of a batch reaction can be ignored then it can be modelled as an activity bounded by a discrete conditional start event, and a discrete scheduled finish event, with the duration of the activity based on observations of the time required for such reactions in the plant. This reduces the continuous modelling requirement on the model substantially, and most models described which are built to look at the behaviour of a complete plant, as opposed to process dynamics, use this approach, for example models described by Felder, Mcleod and Moldin [133], Leggett [134], White [135], Kuriyan and Reklaitis [136] and Young and Reklaitis [137].

Researchers have also described approaches to avoid the use of state variables in modelling product transfers in a batch plant model. Secker [51] comments on modelling a complex batch process plant in which the time to transfer batches between vessels was significant, saying that using a "time-slicing" approach caused the simulation to run unacceptably slowly. He overcame this problem in modelling batch transfers between vessels by determining the transfer time for a batch as the quantity of material divided by the pumping rate. The time that a transfer was complete could then be scheduled as a discrete event on the batch. However, the transfer '..activity could not be interrupted until all the material had been transferred.' Morris [138] describes a similar approach, in which batches are modelled as discrete entities which sieze groups of resources for the times required for transfer and processing. Fisher [52] developed a different psuedo-continuous approach in his model for visual interactive management of a continuous process chemical plant. In this model he was concerned with maintaining the balance of material within a number of interconnected vessels in the plant. For this purpose he needed to add a continuous activity modelling facility to the event scheduling system Micro-Vision [139]. This was achieved by adding a vessel entity data structure and a set of procedures to operate on it to the Micro-Vision modelling library. Each vessel possesses a number of valves. The flows are modelled by setting a rate on a valve; in-flows are positive, and out-flows are negative. Adding all the flows for a vessel gives a nett rate, whose sign indicates whether the vessel is filling or emptying. Fisher [52] comments; 'Given the current contents and a maximum and minimum level it is possible to determine the time at which one of these critical conditions will be reached. An event may then be scheduled for this time, at which the necessary control must be exercised to stop the in- or out-flows which caused the condition. This permits the continuous flows to be modelled by an event scheduling approach.' Each time a change is made to the nett rate on a vessel the system events on that vessel are rescheduled in line with the new rate and current contents of the vessel. To ensure that the efficiency of DES is maintained the current contents

of a vessel are only updated when a request is made by the user program, or a change to a vessel parameter is made. This is much more efficient than updating the contents of the vessel at each time step within the simulation. However, Fisher comments that in order for this appoach to be applicable the rates must be constant over time otherwise first order differential equations need to be solved as in other continuous approaches. This is a realistic assumption to make in many cases because the plant will be designed to maintain constant flow rates and through the common use of flow control devices between process units. Flow controllers compensate for the changing flow characteristics of plant items such as plate heat exchangers where product builds up between the plates of the unit over the time for which it is operated and reduces the nominal flow rate of the unit.

If the psuedo- continuous method can be used the model will be more efficient than if the state variable approach is used, which is an important consideration if modelling a complex plant with a large number of interacting units. In a number of examples cited where continuous transfers are modelled using integration routines for example Knopf [24], Roth [130], Kuriyan and Reklaitis [136] and Young and Reklaitis [137] the transfer rate is kept constant so the integration routines could be replaced by an event scheduling mechanism.

### 6.2.2.Vessel Threshold Levels

In a process plant simulation the user needs to be able to set threshold levels on vessels which correspond to warnings that a level in the vessel has been crossed in a particular direction, so that some action can be taken to prevent the system "cycling" around a critical level without moving forward. This can be done by setting warning thresholds against which state variable values can be tested when they are updated during the continuous phase of the simulation. Knopf [24] describes the use of in line storage tanks in the

simulation of a liquid milk processing plant in which he says 'To avoid rapid on-off cycling of these units, certain threshold levels in the tanks are normally set. For example, a tank may have to be 75% full before downstream units can begin withdrawing product. In addition, once a tank has become full, an arbitrary level of 75% empty may be used before the same product can again be put in the tank.' Kuriyan and Reklaitis [136] also describe the setting of fixed arbitrary levels to contain the cycling effect in a liquid milk plant.

Fisher [52] describes the modelling of threshold levels using the constant rate scheduled event approach. In addition to the basic maximum and minimum critical conditions which exist on a vessel, facilities are provided to model two warning threshold levels, one upper and one lower, for rising and falling contents levels. For a given nett rate an event is also scheduled for the appropriate warning level as well as the critical level so that an appropriate action can be taken by the model builder/ user if desired.

## 6.3.MODELLING OF PLANT LAYOUT AND CONSTRAINTS ON PRODUCT ROUTING

A batch process plant typically consists of a number of plant process and transfer units, and storage vessels, in one or more stages, which are directly linked together by a network of pipes. Movement of product batches in the network is controlled by setting particular configurations of valves open and closed to setup specific routes between the plant items. The fact that the items are directly linked together limits the number of potential routes which could be set up through the plant. It is recognised that these physical constraints on the flexibility of the plant are very important in the operation of the plant as stated by Ready, Simmonds and Taunton [128] and White [135] for example. However, the approaches to modelling the plant network used vary considerably in the degree to which this is taken into account.

Some models of batch process plants such as that of Secker [51] ignore the constraints of the piping network between plant items altogether, allowing complete freedom in the interconnection of vessels. Fisher [52] models the storage vessels at each stage of a plant at an aggregate level to overcome problems in modelling complex connections between a number of individual tanks. Joglekar and Reklaitis [127] describe the modelling of a batch process plant as a directed acyclic graph, with the nodes representing plant items, and the arcs representing the feasible connections between them. Vaessen [105], Hofmeister, Halsz and Rippin [140], Roberts et al. [129], and Ready, Simmonds and Taunton [128] all describe object- oriented approaches to the modelling of batch process plants which allow users to define the possible connections between plant units through slots in the object class instances.

In addition to describing the possible connections of plant items the system must be able to account for the constraints imposed on the connectivity of plant items when physical connections are set up between plant items for routing purposes. Therefore, these physical connections need to be represented. Secker [51] describes the use of a link procedure, and a LINK interaction in his model for connecting vessels together to transfer batches in a batch process plant. In the BOSS system [127] when a connection is made between two units this is modelled by a transfer line resource. Young and Reklaitis [137] describe the use of specific transfer lines assigned to particular storage units and raw material sources to limit the number of connections that can be made at any one time.

### 6.3.1.Dynamic Connectivity

Just defining the possible connections and current connections between plant items is not enough to model the true restrictions on routing through a plant caused by the plant valve arrangements, even if limited transfer resources are modelled. Although in an "empty" plant any feasible connection of two or more

plant items can be made, once a number of process and Cleaning In Place (CIP) routes are set up through the plant, the remaining feasible connections of plant items may be substantially reduced because the valve arrangement which makes one connection disallows another potentially feasible connection even though there are transfer resources available. Thus the problem of determining which items could connect to each other while the system is running becomes dynamic and dependant on the state of the plant at any particular instant. Thus batch plant items are subject to a dynamic connectivity constraint which critically affects their availability for use in production activities. Representing this constraint and accounting for its effects when scheduling using a simulation is the key issue which must be addressed in order to produce feasible schedules.

Modelling Dynamic Connectivity

At the process control level a number of areas of research have involved modelling batch process plants in great detail. Work has been done on modelling the dynamics of batch processes by Walters, Terroux and Waye [141]. Research has also been carried out on computer aided operating plan synthesis, in which the sequence of operation of the individual valves, and adjustment of process parameters required to carry out a particular process operation involving a number of plant items are determined. O'Shima [142] reviews a number of approaches such as depth first search of the plant network to find a route for a process stream between two plant items, Lakshmanan and Stephanopoulos [143] describe a non- linear planning approach, and Machietto [17] describes a rule-based approach. Work has also been carried out by a number of researchers on the use of real- time expert systems for on- line control of processes, for example the ESCORT system by Sachs, Paterson and Turner [59] which monitors the status of a plant in process to aid the plant operator in making control decisions. Hofmann, Stanley and Hawkinson [144] describes the features of G2, which comprises a

combined continuous/ discrete simulator and a real- time expert system for process control. The simulator can be used for testing a real- time knowledge base before it goes on- line, or can be run in parallel with the plant control system to deduce the internal states of the plant which cannot be measured.

All these applications related to process control involve modelling every valve and sensor involved in a process and accounting for their status over time. At the plantwide level of scheduling and batch management, where interest is focussed on the co-ordination of all the current and future activity of the plant, rather than the detailed control of individual processes, it is desirable that "dynamic connectivity" can be modelled without having to represent the status of every valve in the plant. There may be several thousand of them, so the model is likely to be very inefficient if decisions about routing have to take the status of all valves in a potential route into account. An appropriate data representation and reasoning procedure is required to take dynamic connectivity into account without resorting to a low level modelling approach.

## 6.4.PRODUCTION RECIPES AND PROCESS ROUTING

A recipe is a specification of the process to be carried out to make a particular product. It describes the materials and amounts required at each step in the process to make a standard size of batch in terms of a 'Bill of Materials' (BOM). A description of the constituent status of the batch before and after each step enables the material balance of the system to be maintained. It details the duration of each step and their temporal precedence relationships so that the batch is processed through the correct sequence of production activities and these activities are properly co-ordinated. Temporal information may indicate whether activities can overlap, or whether a particular intermediate is unstable, and must be used for the next stage of the process within some short timescale. Recipes are typically described in terms of a

network or sequence of interlinked steps as by Joglekar and Reklaitis [127], Vaessen [105], Roberts et al. [129].

Cherry and Preston [145] describe the BATCHMASTER system and some characteristics of a batch process recipe; 'Splitting and merging of the batch may occur, thus dividing the batch into separate blocks. There may also be reuse of material from batch to batch (i.e. recycles). The most important characteristic for our purposes is that the recipe contains no references to particular plant items- it specifies minimum work required to carry out the process..' 'It can thus form the basis for developing a sequence and a schedule to run the process on any set of plant items as it should not need to be changed, only extended with plant specific information'.

The plant specific information details which plant items are suitable to use for a specific step in the recipe and can comprise a process route. For example, Jogleker and Reklaitis [127] describe the facilities in BOSS; 'The assignment of plant items to particular tasks is defined through the process routing information, and consists ..for each task, of the i.d.'s of the equipment items which are suitable for executing that task.'

The recipe and routing information must be easy to generate and update. Roberts et al. [129] describe a "process specification assistant" for generating process plans based on a process specification knowledge base which contains rules about chemical reactions, and process plan syntax knowledge. Daugherty and Felder [23] discuss the representation of recipe information for batch processing as a directed graph, and describe a PROLOG based natural language interface for user input of recipe descriptions based on sentences made up of a precedence clause, a task clause, a task duration, and the resources required.

## 6.5.PRODUCT REPRESENTATION

As well as representing the physical plant items, their behaviour, and their relationship with each other, the product has to be represented in the model. In a discrete parts manufacturing simulation, each part may be represented individually, or in a batch containing a number of parts. These discrete entities may move through the model via queues in front of machines where they possibly undergo some transformation or may be combined with one or more other parts into a larger assembly. Thus the representation of the product in this case is fairly straightforward. A number of approaches to the simulation of batch process plants account for the product in the same way by describing each batch as a discrete entity which moves through the plant being processed at different stages for example Secker [51], Kuriyan and Reklaitis [136], and White [135]. However, in a batch process environment, processing typically involves starting from a few simple raw materials held in bulk, which are processed through a number of stages in intermediate bulk batches of variable size which may be merged or split, and then possibly packaged into very large numbers of discrete final product units. Thus there is a problem of exactly what constitutes a batch of product within the system and how to represent it at different production stages in the plant.

The representation of the product in its final packaging size is typically not a practical proposition. Pidd [146] discusses the simulation of automated food plants which might be producing 1000 units per minute, and dismisses the idea of representing this number of entities even scaled down by a factor of 100. Instead he says the focus of the simulation should be on the plant items as the main entities; 'If the simulation concentrates on the plant sections (for example, wrapping machines) rather than the product... The product may be treated as a continuous result of the co-operation of the entities.'. This is a feasible approach to take if the main aim of the simulation is to look at typical plant behaviour, but where detailed reporting of product progress is required

as in a scheduling study the product must be represented at a suitable level so that it can be tracked.

The use of a discrete representation of batches means that compromises have to be made in modelling some of the plant processes. Kuriyan and Reklaitis [136] describe the representation of product batches in a model of a dairy plant by composition and final packaging size in order to evaluate scheduling heuristics. The drawbacks of the representation can be seen through an example. A single product composition of chocolate milk is represented throughout the model as two separate products because of a difference in packaging size. They say 'This is not an exact representation of the dairy plant since, in reality, the two products can be distinguished from each other only after the packaging step begins. Thus, while it is possible, in practise to simultaneously store both of the chocolate milk products in single surge tank, the simulation model does not permit this.' They say that to model the actual plant more accurately a generic chocolate milk product from which both sizes could be packed from a single storage tank would have to be modelled. This is a very common situation in this type of environment and should be addressed by the modelling system.

The product representation should reflect the fact that material is processed at a number of different stages through the plant, so that the output from one or more stages becomes one of the inputs for another stage. The size of a batch processed at a particular stage will not necessarily correspond with the size of a final product batch. Therefore, it may be more appropriate to represent batches of intermediate product which are produced in the plant at different stages and supply the production process at the next stage with an amount required by the process recipe. Barnes and Gardner [40] describe a scheduling simulation of an agricultural plant consisting of three stages at which product batches from a preceding stage may be combined or split. Therefore, rather than define a number of batches which will enter the simulation and progress

through the stages, the final products requirement is used to generate intermediate product batch requirements at each of the preceding stages with corresponding due dates for these stages.

In some batch process plants, the production scheduling is not driven entirely by final product due date, but may also be driven by raw material arrivals. For example, in dairy plants which take in raw milk from dairy farms in the surrounding area, there is a requirement to ensure that the milk reception area of the factory always has sufficient available capacity to take in milk irrespective of the final product demand. This requirement drives part of the scheduling of the factory, and it is often necessary to move milk out of the reception area and process it as stock, to be held in storage further downstream in the plant process network. Thus, there may be a requirement in the simulation model to represent "stock" which is only assigned to actual production requirements when they become known.

## 6.6.PLANT SCHEDULING AND CONFIGURATION

The scheduling problem in batch process plants basically involves matching the recipe requirements for a particular product to the plant such that temporal and physical constraints are not broken, while keeping to the rules under which the plant is being operated.

### 6.6.1.Scheduling Approaches

Where batches of product are represented as discrete entities, scheduling is typically based on the use of local priority rules. White [135] describes the incorporation of local rules into a batch process simulation to decide what batches to progress through the system based on due date, and batch characteristics, and how the currently available plant items should be configured to achieve this. Young and Reklaitis [137] describe how batches are

progressed through a model constructed with BATCHES; 'The flow of batches of material through the process is controlled through user specified product-processing sequences and queue- priority disciplines. The former define the quantity and order in which the required batches are made, while the latter is the mechanism for assigning batches to downstream units during the course of recipe execution.' As previously discussed the representation of product batches as discrete entities is inappropriate but the use of priority rules forces it. Kuriyan and Rekliatis [136] comment on the drawback of using priority rules in prohibiting the use of generic intermediate products with a batch size different from the final product batch size. The model has to use products differentiated throughout by packaging size, because the processing sequence of the final product batches is determined by a FIFO queue in front of a number of filling machines which process batches based on packaging size.

In Chapter 3 the limitations of using local dispatching rules for control were discussed for scheduling simulation generally. Because of factors such as the interlinked nature of batch plants, and the finite intermediate storage capacity, the ability to consider the global status of the system is an important consideration in scheduling. Secker [51] and Fisher [52] used VIS to allow the user of the model to make decisions based on the global status of the model. White [135] also describes the use of look- ahead rules, rather than local priority rules, which assign batches to plant units on the basis of estimating when the batch will finish processing, and when downstream plant items will be available. Barnes and Gardner [40] describe a three stage system in which the activities of the stages are synchronised. The overall operating strategy of the plant is determined by the second stage which is perceived as the bottleneck. The batch sequence derived for this stage is used to drive the sequencing of the first stage units so that they supply the ingredients for the second stage process at the right time. Because the batches in this approach are not entities which move through the model from start to finish, it is possible to schedule the production of intermediates in batch sizes suitable for

the stage, which then feed the production process at the next stage; a more natural approach for batch plants.

### 6.6.2.Configuration of Process Routes

As described above the recipe defines the process steps required to produce a batch/ batches of product, and each step must be assigned to a plant item or items during the scheduling process. Cherry and Preston [145] describe the assignment of plant items to steps from the recipe to develop a sequence from which the plant schedule will be derived; '..if the steps can be done in several alternative items a considerable problem can arise.' because the assignment process becomes combinatorially complex. The actual choice of a plant item or items for a step is constrained in a number of ways. Plant items must be in the right state, for example the use of clean plant items is an important constraint in a lot of batch process plants. The physical layout of the pipe network and routing valves in a typical plant constrains the items which can be assigned to transfer and continuous process steps to items which can be physically linked together in the plant. Dynamic connectivity further constrains potential choices at any given time. Temporal requirements mean that the assignment process should consider whether items required for different steps will be available at the correct time. In some cases the timing between steps can be very important, for example between two No Wait steps, when the intermediate produced by one step in the process is unstable and must be processed at the next step straight away. The schedule must ensure that when the first step finishes the plant items will be available, and can be reached, for the second step to start immediately. In other cases it will be possible to have a delay between steps in the process until plant items become available. The finite capacity of intermediate storage is an important constraint on whether plant items can be assigned to process steps. Ward [147] describes tank management as one of the most important tasks in the scheduling of a plant, and the intermediate tank filling policy is often

important to the performance of the plant. Knopf [24] describes the use of storage tanks to allow upstream and downstream units to operate at different rates, and to absorb process fluctuations. A tank filling policy could be fill or empty only, or fill and empty. Young and Reklaitis [137] indicated that the storage policy could have a significant effect on the capacity of a plant.

The assignment of plant items to process steps is generally based on user preference where choices exist, and is approached in a number of ways. Secker [51] describes the use of "preferred routes" which define a number of vessels at the next stage which could be linked to and pumped to. The list of vessels at the next stage for a given vessel is searched in priority order, to attempt to set up a link. If it is not possible to do this, then the model will stop and request input, which will either be an identifier of a non- preferred vessel, or an instruction to hold and wait until a vessel at the next stage becomes free. Vaessen [105] describes the mapping of product specifications onto the factory layout of available equipment which are both represented as directed graphs, while taking into account a number of preferences. In Ready, Simmonds and Taunton [128] scheduling is rule based, with user written rules expressing the scheduling strategy, and equipment preferences. Roberts et al. [129] describe the "batch process manager assistant", which attempts to match equipment to a process plan for a batch. The process produces groupings of equipment that could be used to manufacture the batch. There may be several different groupings for each possible process plan for the same product. The module uses the database of current orders, and a knowledge-base which contains rules about customer priority, equipment cleaning, process preference, equipment history, batch sequencing rules, cost history, cost estimates, and other knowledge to aid the user in the selection of the appropriate grouping.

## 6.7.BATCH TRANSFER, PLANT ITEM CO-ORDINATION AND MATERIAL BALANCE

The appropriate modelling of batch transfer and processing through semi-continuous plant units is directly related to modelling the plant configuration as described previously. In a typical batch plant, the transfer and continuous processing of a batch from one plant unit to another may require that a number of plant items are linked into a chain and their individual activities are co-ordinated together. The chain may contain branches where product is split, or two or more intermediates are combined affecting the material balance of the system.

This requirement to co-ordinate the linked plant items in batch process plants for batch transfer and semi-continuous process routing is recognised and described by a number of researchers. In the approach of Secker [51] the link procedure ensures that vessel contents are correctly updated when a batch transfer is made between two linked vessels, and the events associated with the transfer are correctly co-ordinated on each vessel. In the model described by Morris [138], batches queue in front of a charging unit and a pair of batch reactors, and sieze the charging unit and one of the reactors at the same time to represent the transfer of a batch into a reactor. White [135] describes the modelling of a multiproduct plant in which batches have to wait for "trains" of equipment items to become available for them to progress at the filling stage of the process; 'Once the batch is released for fill-out it continues to sit in the formulation/ mix unit until all the necessary equipment is available.' 'Often a batch released for fill-out must wait for a fill-out train, holding up the use of the formulation/ mix unit. Once all of the equipment in the train is available and assigned, the batch moves through it; then all of the equipment is cleaned and available for use on another batch.'. Kuriyan and Reklaitis [136] describe the modelling of batch movement in BOSS for studying the daily production scheduling at a multiproduct dairy plant. The transfer of batches throu

plant is determined by reserving downstream plant items for batches as necessary, for example, a pasteurisation task on a batch will not commence unless there is a surge tank reserved for it downstream.

In some cases the transfer of a batch may be via a semi- continuous process entity such as a separator, or evaporator which alters the volume of the batch, or may split it. The model should be able to calculate these volume changes to ensure that the material balance of the system is correctly maintained.

## 6.8.MODELLING OF CIP ROUTES

The need to model the cleaning of plant items is recognised as important in correctly representing the performance of the plant [146]. However it is usually modelled in a simplified way as an activity which is undertaken by a specific plant item with no effect on the rest of the plant. Individual plant items which require cleaning are set to clean with a scheduled event to indicate the end of the cleaning activity. In fact, the modelling of cleaning requires as much consideration as the modelling of batch reactions and batch transfer, because it is an integral part of many batch process operations, and has a considerable effect on the availability of plant items for production in the plant. In a typical set up for a CIP system in a plant such as a dairy, there will be one or more CIP units. Each CIP unit will supply the required cleaning solutions to one or more CIP circuits which clean specific plant items in the plant via one or more cleaning routes. A given CIP route will probably use at least part of the pipework used in one or more process routes. Therefore a set of relevant structures to represent this Circuit/ Routes/ Units hierarchy are required, and the setting up of cleaning routes, and starting and finish of cleaning activities must be properly co-ordinated with the process routes which are run.

## 6.9.DISCUSSION

There are a number of features which are important in modelling of batch process plants, and have been addressed to some degree as described in this review. However, a number of features need to be taken further than at present.

1. The activities of plant items should be represented so that they will "map" onto the process control functions directly. In most cases the use of continuous modelling facilities should not be necessary at this level of detail, so a psuedo- continuous event scheduling approach will be more efficient.
2. The representation of the plant layout needs to address the problem of dynamic connectivity constraints without resorting to the level of modelling all the valves in the plant. This is one of the key requirements in a system which is going to be used for scheduling and control to ensure that it produces an activity schedule for plant resources that is actually feasible. Therefore, a mechanism is required in the modelling system for determining the dynamic connectivity constraints on any plant item as the system runs.
3. The representation of routing in the plant needs to be flexible with respect to the size of route, and numbers of choices which can be represented.
4. The model should be able to co- ordinate the activities of variable numbers of plant items linked together in order to correctly represent the behaviour of complex transfer and process routes which have been set up.
5. The modelling of CIP routes must be fully addressed in conjunction with process routes, as they are often a significant factor in the operation of a plant.

6.The representation of batches as discrete entities which move through the model is not necessarily the most appropriate one for a batch process plant. The representation of intermediate batches at different stages which feed other stages may be more appropriate in a lot of cases.

7.The scheduling of the plant should be based on it's global status, rather than the use of local priority rules, with user preference for plant item assignment to process routes properly represented. The mechanism by which plant items are assigned to routes must take into account the interlinked nature of plant items and the dynamic connectivity constraints.

8.The management of intermediate storage is an important part of batch process management, and the system should address the use of dynamic level triggers on storage vessels, and the operating policies which can be assigned to individual vessels within a model.

# CHAPTER 7 THE MODELLING OF BATCH PROCESS PLANTS FOR SHORT TERM SCHEDULING AND BATCH MANAGEMENT

## 7.0.INTRODUCTION

As discussed in Chapter 1, the area of this research is short term scheduling and control of batch process plants at the "batch management" level. This is the scheduling of production activities in the plant on individual plant items such that the details of the process control level are not represented, but the resulting schedule can correctly "map" down onto this level so that it can be implemented and managed through the plant. The review chapters covered a number of different approaches to research and practise in scheduling and modelling in manufacturing systems with reference to the batch process industry where appropriate. As described in the review, analytical approaches which attempt to develop optimum schedules typically have to make simplifying assumptions which mean that the schedules are actually infeasible. Researchers in the areas of simulation and artificial intelligence have demonstrated that much more realistic models for scheduling can be developed which can overcome these simplifying assumptions. They are no longer concerned with "optimality", but are concerned with developing a good schedule which takes into account the true complexity of the environment and the real criteria in scheduling, the most important of which is generally considered to be the meeting of due dates. There are a number of features which should be in such a short term scheduling tool for a batch process plant if it is to produce good feasible schedules which can be implemented in practise in the plant. The features of primary importance were described in Chapter 6 and can form a specification for a knowledge-based simulation tool. The incorporation of many of these features has been addressed by a number of systems described in the literature. However, some of the key features of batch process plants have not been developed far enough. The main emphasis of the work described in this thesis has therefore been carried out in the

following key areas concerned with modelling batch process plants at the batch/ unit level:

1. The development of a representation scheme for representing the structure of a plant at the appropriate level of detail for batch scheduling, and the development of procedures to infer the actual availability of plant items for production activities subject to the constraints imposed by dynamic connectivity.
2. The development of an approach to the dynamic configuration of routing within a plant network, taking into account the constraints on plant item availability and preferential considerations for the allocation of plant items to routes.
3. The development of a dynamic simulation to take account of finite capacity constraints of a batch plant when developing a schedule based on route configuration.
4. The development of an activity scheduling framework incorporating the dynamic configuration of production routes.
5. The development of a hybrid simulation structure for the implementation of these elements as a declarative language based Control Module for activity scheduling and route configuration, and a procedurally based Simulation Module.

These features form the basis of a generic system called the Batch Process Scheduler (BPS) which will allow realistic feasible schedules to be developed for batch process plant management. The development of the plant representation scheme, configuration procedures and simulation will be described in this Chapter. The framework which has been developed for activity scheduling will be described in Chapter 8. The way in which the BPS has been implemented as a hybrid knowledge-based simulation will be described in Chapter 9.

## 7.1.A GENERIC SPECIFICATION FOR BATCH/ UNIT LEVEL MODELLING

The BPS should enable a model of a specific batch plant to be built which incoporates sufficient detail to schedule at the level of a "batch" so that the schedule of batch operations will map down onto the process level of operation correctly. Thus the features which are described in this specification were developed on the basis that a set of process level operations could be bounded by an activity described at the batch level. A model which is to be used for scheduling must be data driven as described in Chapter 3. A model of a specific plant must be able to be built and altered through data input only in order to allow the model of the plant and the way it is scheduled to be developed incrementally by the BPS users. Batch plants are often subject to change, so in order for the model to remain useful over a long timescale, it must be possible to easily change the structural representation and control rules. Therefore the BPS must be generic with respect to the plant which it is being used to model.

The manufacture of a given product requires carrying out a number of activities or tasks according to a production "recipe". Scheduling and control involves determining which activity or activities to carry out at any given time, and how to allocate the plant resources to them. The feasibility of the schedule fundamentally depends on properly accounting for the constraints on resource allocation to activities. Therefore, the knowledge representation in the BPS has been developed primarily from the point of view of developing the generic features for representing the production activities, plant items and structure of the plant, and rules for allocation of resources to production activities. The BPS will correctly maintain a model of the status of the plant and product under changing dynamic circumstances, so that the allocation of resources to activities can only take place if the dynamically changing connectivity constraints of a plant are not broken. In addition to this

fundamental structure, the features required for the representation of recipes, the status of products with respect to production requirements, and reasoning about production activities for scheduling have been addressed to develop a full specification for the generic BPS.

### 7.1.1.Batch Plant Coverage by the BPS

In order to determine the features which should be incorporated into the BPS and enable it to be tested with real data as it was developed, a project was set up with Eden Vale, part of Northern Foods plc, to model a large multiproduct dairy situated at Minsterley near Shrewsbury in Shropshire. The specific goal of the project from the point of view of Eden Vale was to look at the handling of skimmed milk within the factory and the control of its supply to the main production areas in the factory. Thus the initial development of the BPS has resulted in features that are generic with respect to the Minsterley plant. In terms of its coverage of all aspects of all batch process plants it is therefore not complete. However, because the Minsterley plant contains a large number of features which are common to other batch process plants it is envisaged that the BPS could be applied to a wide range of plants with little or no modification to the program. In some other cases some expansion of the BPS features would be required, which would make it more complete in its coverage of batch plants overall. The determination of the generic features required in the BPS was partly based on personal experience of the plant, partly on the batch plant scheduling literature and partly on information collected during data collection and knowledge elicitation at the plant itself. The following sections will describe the generic features derived for modelling at the batch management level of operation. The reasoning and assumptions behind their derivation, and the methodology by which a model of a specific plant should be developed and used will be described. Examples will be given of the use of these features to model a batch plant for short term scheduling

and control based on the data collected and scheduling rules elicited during the project at Minsterley.

## 7.2.SPECIFICATION FOR THE MODEL AND RESOURCE ALLOCATION TO PRODUCTION ACTIVITIES SUBJECT TO PHYSICAL CONSTRAINTS

The basic generic components required to build a model of a batch plant are structures to represent the classes of production activities carried out in the plant and the classes of plant items used in these activities. Most systems which model batch plants contain this type of classification scheme. However, in order to produce a feasible schedule, it must be possible to specify the structural constraints on the use of resources for activities, and to correctly take account of the dynamic connectivity constraints when allocating resources to activities. This aspect is treated in a simplified way by the systems described in the literature which leads to infeasible and unrealistic schedules.

### 7.2.1.Batch Plant Production Activity Classification

Approaches to modelling batch process plants vary somewhat in the terminology used to represent the individual production activities carried out in a plant. For example, the approach of Cott and Machietto [5] describes production procedures which are split into a number of phases such as FILL REACTOR| CARRY OUT REACTION| EMPTY REACTOR. The BATCHES simulation system of Joglekar [127] describes tasks which are split into sub-tasks to achieve the same thing. Both phases and sub-tasks represent the use of resources to carry out some function which is part of the production process over a period of time. In this approach, the term production activity has been used to represent the use of resources for a particular function at the phase or sub-task level of detail, such that the carrying out of production activities will correctly "map down" to the process control level of plant operations.

Production activities in a batch process plant such as a dairy can be represented as belonging to two classes: processing activities and cleaning activities. These must be represented at the right level of detail as described above. In collecting data on how product was processed through the Minsterley plant it became apparent that although a number of plant specific processing activities were carried out, they could all be classified as either a batch reaction type of activity involving a single plant item, or a semi-continuous process/ transfer type of activity involving routing of some sort. A batch reaction activity can be viewed as simply holding product in a reaction vessel for a period of time equivalent to the overall time for the reaction phases and accounting for any process related changes to the product at the end of the reaction. A transfer or semi-continuous processing activity typically involves linking a number of plant items together serially and in parallel to form a route, through which the product moves and may be processed over time at a constant or variable rate. In addition semi-continuous processing may well involve "splitting" an intermediate product at some point into two or more derivatives, or the combination of two or more intermediate products into a single final or intermediate product. All that is required of the model of the process at this level is that it accounts for the process related changes to the product, and correctly maintains the material balance of the system. A simple model of reaction processes and semi-continuous processes is all that is necessary at the level of batch management because the main concern is how the time and resources required to carry out the processes and changes in product composition affect the operation of the plant rather than the exact details of the processes involved.

The representation of cleaning activities needs to be treated equivalently to processing activities. In a number of areas in which batch plants are used cleaning is of particular importance for good product quality and represents a considerable constraint on the time during which plant items are available for

production purposes. A batch plant typically has a Cleaning In Place (CIP) system, which is used to carry out cleaning activities. A cleaning activity can be viewed as similar to a product transfer activity typically involving linking a number of plant items up into a cleaning route which is then put through a cleaning cycle. A typical cycle might include a number of phases such as purge out product with rinse water, circulate detergent at a set temperature and concentration for a set period, rinse out detergent, and possibly circulate a sterilising solution for a set period. However, for the purpose of a batch scheduling a relatively simple model in which all the phases of a cleaning activity are considered together as a single block of time will give the appropriate level of detail.

An example of a semi-continuous process activity which is common to a number of batch processes including dairy processing is separation, in which an intermediate product is separated out into two or more component products. A specific example of this activity from the Minsterley factory will be used throughout this chapter to illustrate how the generic features described can be used to build a model for batch level scheduling and control purposes. Figure 7.1.a. shows a process flow diagram of the milk reception, dairy and main production department storage areas of the Minsterley plant. (Figure 7.1.b. provides a key to the plant items in this diagram and their descriptions defined in the model of the Minsterley plant and used in the examples which follow). The dairy department separates milk to provide the main production areas with their skimmed milk and cream requirements by a number of routes. In the case of the evaporator department, one production activity which can be defined is the separation of milk for evaporator skim, **sep-milk-ev-skim** and this will be used for illustrative purposes throughout this chapter. Scheduling this activity involves making choices about routing through the plant, and the plant items to be used in a route, and can also illustrate the additional complexity that arises when consideration has to be given to more than one product. When this activity is carried out milk from

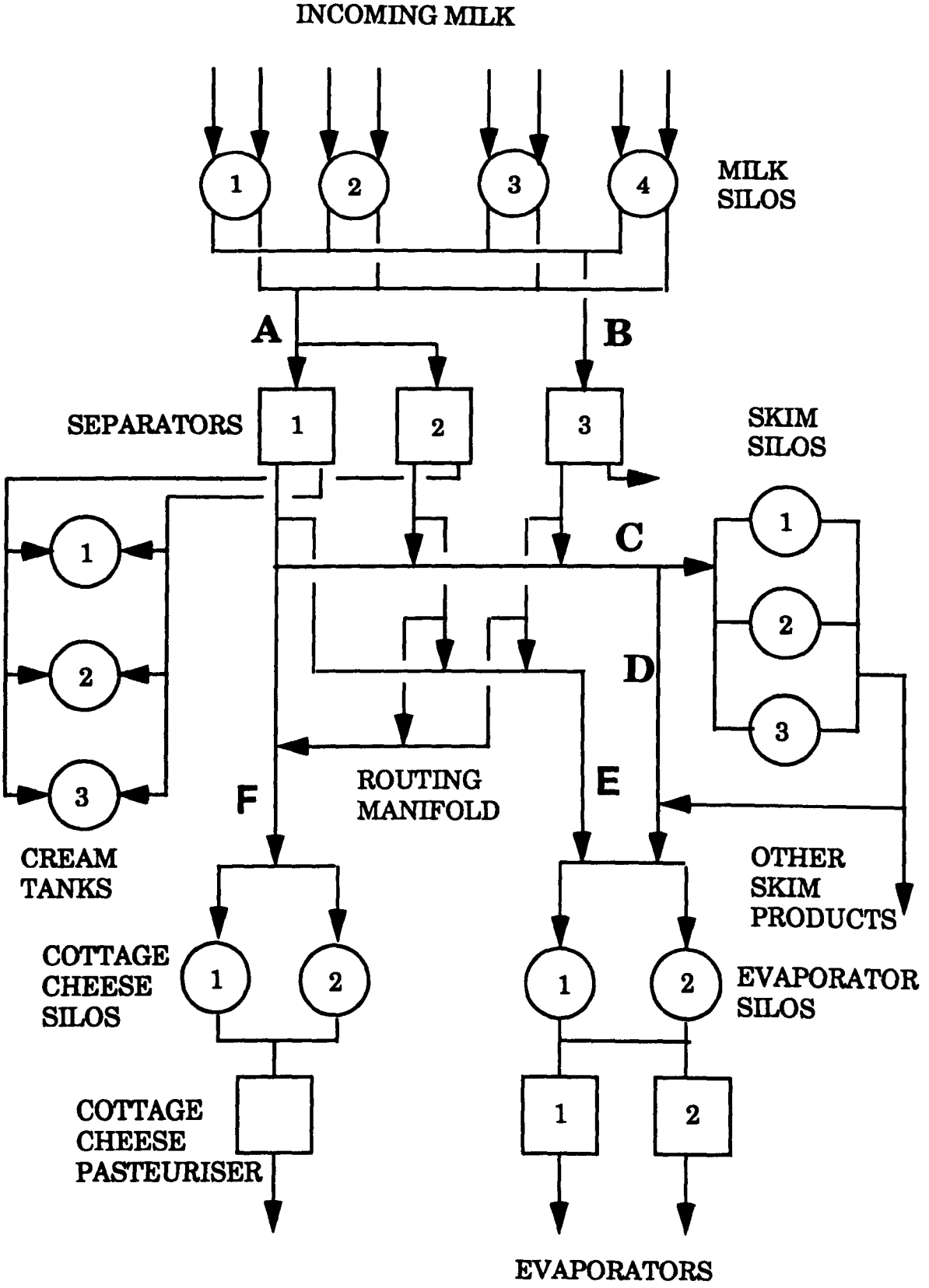



Figure 7.1.a.Minsterley Process Flow

| FLOW DIAGRAM DESCRIPTION | MODEL CODE |
|---|---|
| **STORAGE VESSELS** | |
| MILK SILO 1 | MSL1 |
| MILK SILO 2 | MSL2 |
| MILK SILO 3 | MSL3 |
| MILK SILO 4 | MSL4 |
| SKIM SILO 1 | SKM1 |
| SKIM SILO 2 | SKM2 |
| SKIM SILO 3 | SKM3 |
| EVAPORATOR SILO 1 | EVS1 |
| EVAPORATOR SILO 2 | EVS2 |
| COTTAGE CHEESE SILO 1 | CCS1 |
| COTTAGE CHEESE SILO 2 | CCS2 |
| (STANDARDISING) CREAM TANK 1 | CTK1 |
| (STANDARDISING) CREAM TANK 2 | CTK2 |
| (STANDARDISING) CREAM TANK 3 | CTK3 |
| **SEMI-CONTINUOUS PROCESS ITEMS** | |
| SEPARATOR 1 | SEP1 |
| SEPARATOR 2 | SEP2 |
| SEPARATOR 3 | SEP3 |
| COTTAGE CHEESE PASTEURISER | CPS1 |
| EVAPORATOR 1 | EVR1 |
| EVAPORATOR 2 | EVR2 |
| **PROCESS LINES** | |
| A: RAW MILK LINE 1 | RLN1 |
| B: RAW MILK LINE 2 | RLN2 |
| C: SKIMMED MILK STORAGE SUPPLY LINE | SKLN |
| D: EVAPORATOR STORAGE SUPPLY LINE 1 | ELN1 |
| E: EVAPORATOR STORAGE SUPPLY LINE 2 | ELN2 |
| F: COTTAGE CHEESE SUPPLY LINE | CRLN |

Figure 7.1.b. Minsterley Flow Diagram Model Code Key

the milk storage area of the factory is separated out into its two component parts of skimmed milk and cream in the approximate proportion of 9 parts skimmed milk to 1 part cream. The skimmed milk resulting from this activity is moved down to the evaporator department storage, while the cream is moved to a cream storage vessel in the cream department via one of the cream standardising tanks illustrated in Figure 7.1.a.

### 7.2.2.Plant Unit Classification

The plant items in a typical batch process plant can be classified according to their functions at the batch level which enable them to be used for the production activities carried out in the plant. If these basic plant item classes and their attributes are defined it is possible to build up a model of a specific plant through data only. The plant unit classes attributes have been developed based on an analysis of the functions of the different plant items in the Minsterley plant and descriptions of plant item functions in the batch process literature. Attributes of particular importance in scheduling and control relate to the connectivity, capacity, and rate of operation.

Most of the plant unit classes in a batch process system are "static" units through which product moves and is processed and have been defined as follows in the BPS:

1. Vessels with a finite capacity for holding product. These can be further classified into two sub-classes; reactor vessels in which some form of process reaction can occur, and storage vessels which are used for product storage and to break up semi-continuous processes into stages. The vessels also enable semi-continuous processes which operate at different rates to be linked together by providing a buffer between them.

2. Semi-continuous process elements are in-line plant items which move product from one point to another at a constant or variable rate and may alter the composition of the product while doing so. There are a large number of different pieces of process plant which perform a function that can be classified in this way. For example, a separator, evaporator and pasteuriser can all be generally classified as semi-continuous process elements, and then further classified into sub-classes based on more specific features than the general classification. Even at the end of a process where bulk final product is often packaged into a large number of discrete items, the packing machines can essentially be regarded as semi-continuous process elements which operate on bulk product at a specific rate.

3. Process lines which provide the routing in the plant to carry product from one point to another, either as part of a semi-continuous process or simply as part of a product transfer route. These items are passive, because they do not have a specific rate associated with them, but take on the net rate of the routes in which they are being used.

A classification for discrete entities which can move through the system has also been defined:

4. Transport entities, which have finite product holding capacity and are mainly used for bulk product transfer functions, either between parts of the plant which are not directly linked together, or to provide input of raw materials and output of finished product from the system. These entities move between queues in front of production areas and the plant items in the production areas themselves. Therefore, the BPS also contains a classification of queues and production areas to enable the movement of transport entities to be represented.

7.2.3.Plant Structure

A batch process plant is typically an enclosed network of linked process plant items and reaction and storage vessels, with routing in the network controlled by the valve configuration of the plant. The structure of the network and the valve configuration can impose severe constraints on the potential routing of product within the plant. Therefore, the connectivity of plant items is the final determinant of whether a particular plant item can actually be included in a route, and the representation of this connectivity is a vital attribute for entities for the BPS to be able to develop feasible schedules. As described in Chapter 6, the connectivity of a plant item is not static, but depends on what the current connections made with other plant items are and whether they constrain its currently feasible connections. Therefore, the dynamic connectivity of plant items must be taken into account when modelling the routing for production and cleaning activities in a plant. At any time during a scheduling/ batch management process the BPS must be able to determine whether the relation **CAN-CONNECT-TO** holds between two plant items which could be used as part of a product transfer or semi-continuous processing route or a CIP route. The relation **CAN-CONNECT-TO** is defined to mean:

> A Plant Item $P_1$ can make a connection to a Plant Item $P_2$ under the physical constraints imposed by the current state of the plant.

There are three aspects of plant representation which need to be considered in order to achieve this aim:

1. The static network showing the feasible connections between plant items in an "empty" plant.
2. The constraints on feasible connections due to dynamic connectivity.

3.The plant item connections which are current at any particular point as routes are set up.

The dynamic connectivity within the plant network is controlled by the valve configuration. Combinations of valves can be set open and closed to set up a particular route through the plant to carry out a product transfer, semi-continuous processing activity or a cleaning activity and this may constrain the use of other plant items in other routes. However, there may be several thousand valves in a large plant, so it is not desirable to have to model them individually for a plant wide application such as scheduling and batch management. There is thus a requirement for a suitable knowledge representation structure to use as a building block for constructing a model of a plant at the right level of detail. Appendix A contains two engineering drawings of parts of the Minsterley plant network which has been used to test the representation structures and control routine described in this thesis. Diagram A.1. provides a schematic representation of the part of the plant concerned with milk separation, storage of milk and supply to the various production departments. This representation shows all the details of the valve arrangements concerned with routing in these parts of the plant. It would be undesirable to have to include all this detail in a batch scheduling model as it would lead to a complex cumbersome network representation which was difficult to understand, to update its status during execution and to modify in the light of changes to the real plant. Diagram A.2. shows a process flow diagram of the milk, skimmed milk and cream routing in the factory, incorporating the parts of the plant represented in Diagram A.1. In this case, the level of detail is not enough to be able to correctly model the routing in the plant, and its status as scheduling progresses. A knowledge representation structure that enables construction of a model of the plant network for batch scheduling should have a number of features:

1. It should represent the way that a plant item can directly connect to other plant items.
2. It should be flexible enough so that the number of feasible connections which can be defined is not limited to some arbitrary figure that would restrict the number of entities that can be adequately represented.
3. It should be "modular", and based on a representation of the entities and how they can fit together, rather than a single structure representing the whole network. This means that a model can be easily reconfigured in line with changes to a real plant. The model can be developed incrementally, and different sections of the network can be represented at the desired level of abstraction.
4. From the representation it should be possible to infer the constraints on the use of a plant item and other related plant items for routing purposes at any point in the scheduling process.

## 7.3. AN AND/ OR STRUCTURE FOR NETWORK REPRESENTATION

A process plant network can be viewed from the point of view of the elements in it and the feasible connections that they can make to other plant items, and this is typically the approach taken in a number of object-oriented systems for batch plant modelling as described by Vaessen [105] Hofmeister et al. [140] and others. However, as described in Chapter 6 this representation does not fully express the constraints imposed by the valve configuration on different combinations of the connections that can be made.

In the light of this, the objective of the work was to develop a representation scheme that compactly and accurately represented plant item connectivity. The use of production rules to represent these constraints was rejected because, as described in Chapter 4, although feasible it would require a rule to represent every constraint on a connection in the plant. This would lead to a

rule-base of considerable size that was subject to problems such as incompleteness, contradictions and maintenance difficulties. It was the authors view that the plant item representation of connections can be extended to include these constraints in a compact and accurate manner by using an AND/ OR construct in the description. For example, for the simple plant network in Figure 7.2.a. Plant Item A can connect to Plant Item B AND Plant Item C; Plant Item C can connect to Plant Item A OR Plant Item B. Thus the AND/ OR structure is a suitable way of representing the alternative feasible connections in the plant, and the degree to which they are constrained by other connections which already exist. This provides a suitable mechanism to test the relation **CAN CONNECT TO** between a batch plant equipment item and its immediate neighbours in the plant. In the definition of an AND/ OR graph as a way of representing a problem solving procedure given by Bratko [148] an AND expansion from a node means that all nodes which are the immediate children of the node must be solved. In the case described here the AND/ OR structure has been used, but the meaning has been slightly relaxed to represent the physical structure of batch plant item connectivity. Thus the definition of an AND expansion from a node when it is being used to represent the physical structure of the plant in terms of the **CAN-CONNECT-TO** relation is:

> up to AND INCLUDING these immediate nieghbours **CAN-CONNECT-TO** this plant item at the same time.

The definition of an OR expansion from a node in terms of a plant items structural representation is:

> one OR other ONLY of these immediate nieghbours **CAN-CONNECT-TO** this plant item at the same time.

The view taken of the connections is thus from the perspective of each plant item including the constraints on the connection which is compact and simple, and easy to visualise. Taking a plant item view of the network gives advantages in terms of modularity, because the description of the static configuration of the plant can be altered by changing a few affected entities, rather than altering a whole network structure. In terms of the user describing the plant it enables a natural incremental approach to be taken, plant item by plant item.

### 7.3.1.The Importance of a Plant Items Perspective

This AND/OR description is still not enough to fully represent the dynamic connectivity of the plant, because each item's view of its immediate neighbours is not the same in terms of feasible connections and the constraints on them. For example, consider a directed network of three plant items A, B, and C, as illustrated in Figure 7.2.a. in which the valve configuration controlling the output from A means that product can flow from A to C directly and also via B. From a plant item point of view this can be described as:

Plant item A can connect to B AND C
Plant item B can connect to C

as illustrated in Figure 7.2.b.

However, the view of the the connections from the perspective of plant item C and its input valve configuration is:

Plant item C can connect to A OR B

as illustrated in Figure 7.2.c.

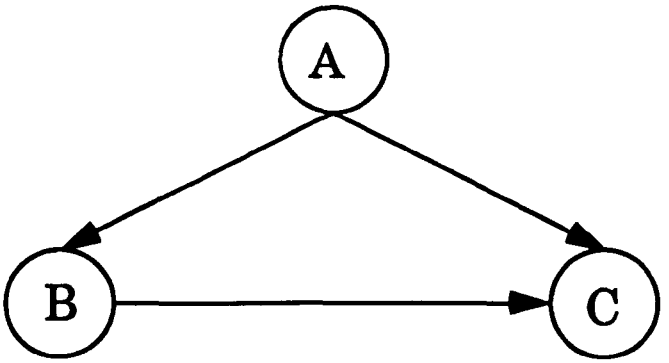



Figure 7.2.a. Simple Plant Network

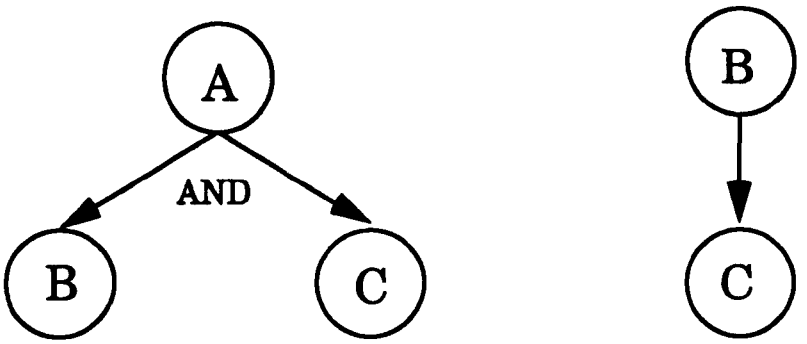



Figure 7.2.b. Output Connections from A and B

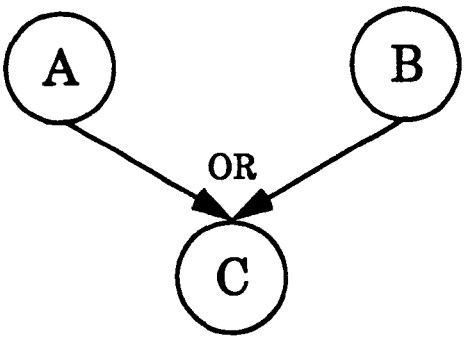



Figure 7.2.c. Input Connections to C

Thus the ability of A or B to connect to C depends on whether there is already a plant item connected to C. However, this constraint on the connection can only be seen from the perspective of C as A and B are only looking at feasible connections from their own perspective of the plant. Associating a direction with the connection allows the perspective of the item to be taken into account when testing the relation **CAN-CONNECT-TO**. In order to represent the direction of a connection each plant item therefore has to have an INPUT "port" and an OUTPUT "port" defined. To determine whether a connection can be made between two plant items it is necessary to look at the feasible connection from the perspective of both plant items, and in the light of any current connections. The basic representation of a plant item is shown in Figure 7.3. and this representation is flexible enough to be used to model the structure of the plant at the required level of abstraction; the level of detail of representation depends on the use of the model. The level of abstraction to which the representation is taken depends to a large extent on the complexity of the routing within the plant. If a plant item can have permutations of other plant items on its input or output, then the level of detail must be closer to the valve arrangement of the plant itself, possibly incorporating the modelling of valve manifolds as entities and feasible routing from these. The AND/ OR structure has been applied to modelling the connectivity of the plant items in the Minsterley plant network by going to the correct level of detail as required. For example, the plant items specific to the skimmed milk routing for the **sep-milk-ev-skim** activity are shown in figures 7.4a.- 7.4d. It can be seen that some plant items only have a single plant item on their input or output port. In this case it is treated as a potential AND connection because it cannot be constrained by any other connection the plant item already has on this port.

In the terminology of this representation each item has an INPUT and an OUTPUT. This does not fix the direction of product flow through the item which is independent of the physical plant representation but simply provides

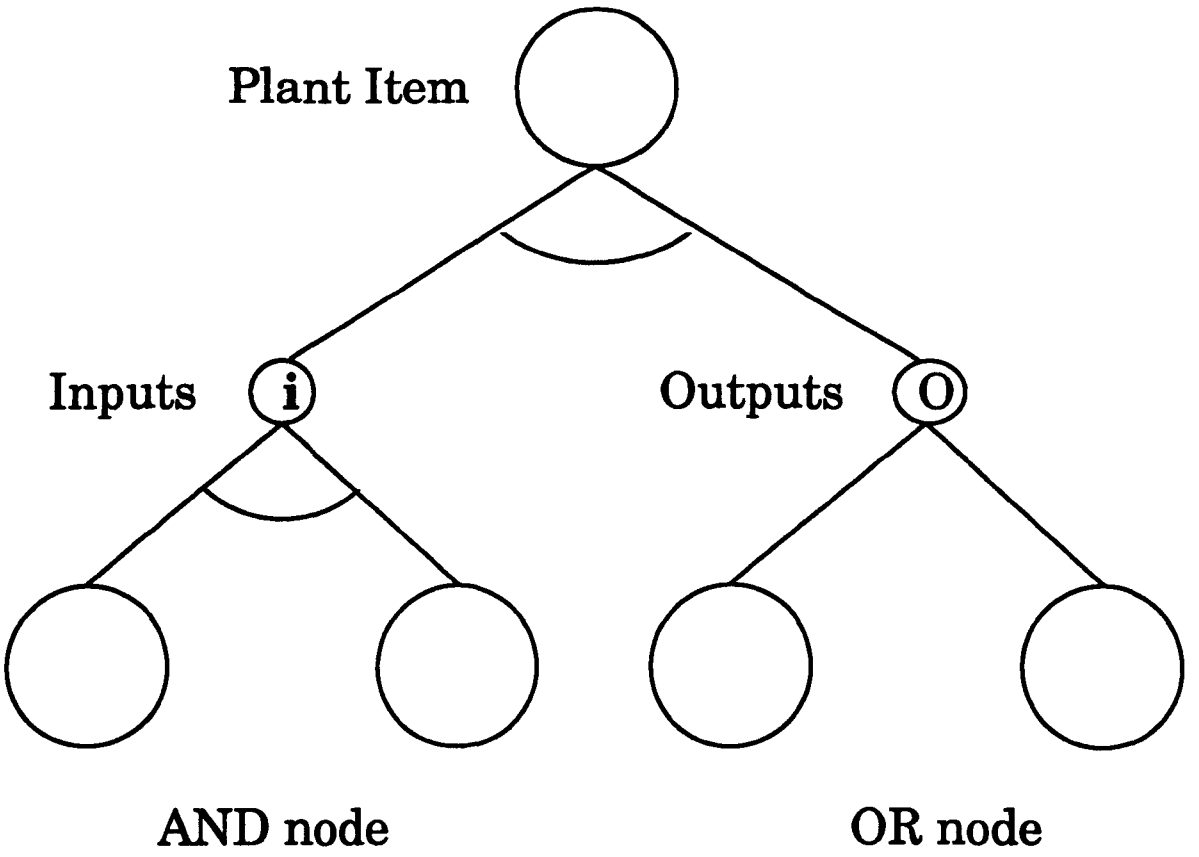
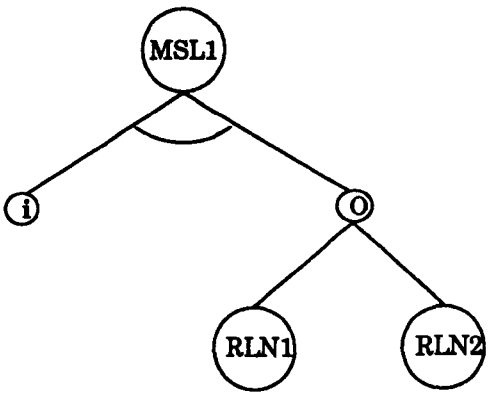
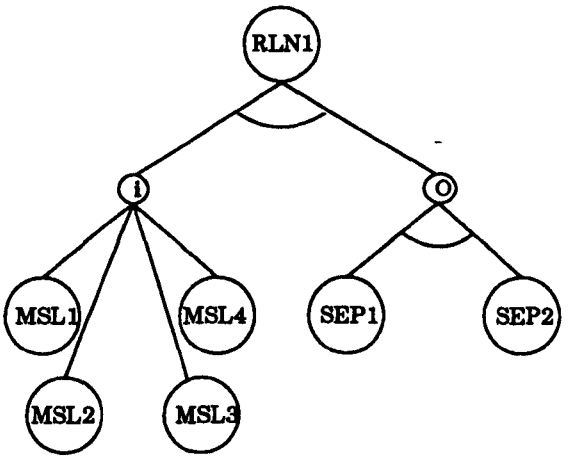


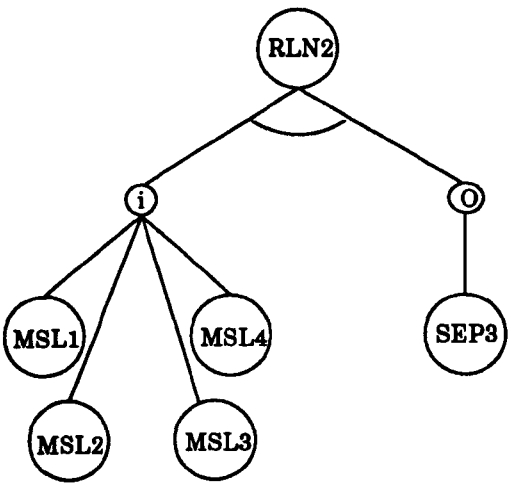

Figure 7.3. Generic Entity Configuration




Milk Silo 1

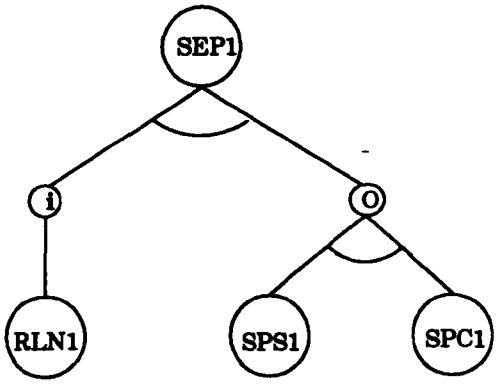



Raw Line 1

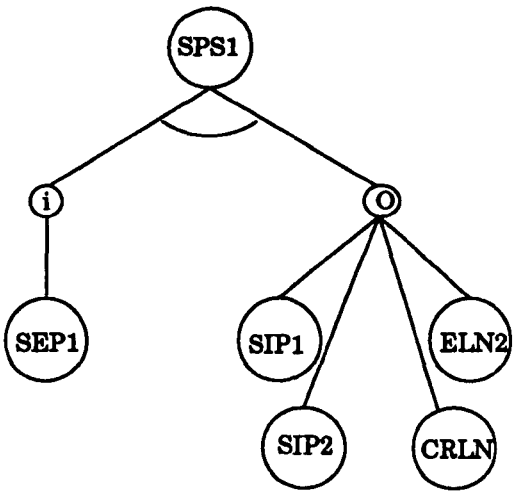


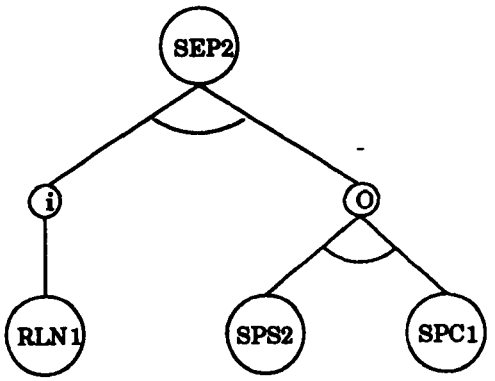

Raw Line 2

Figure 7.4.a. Milk to Separators

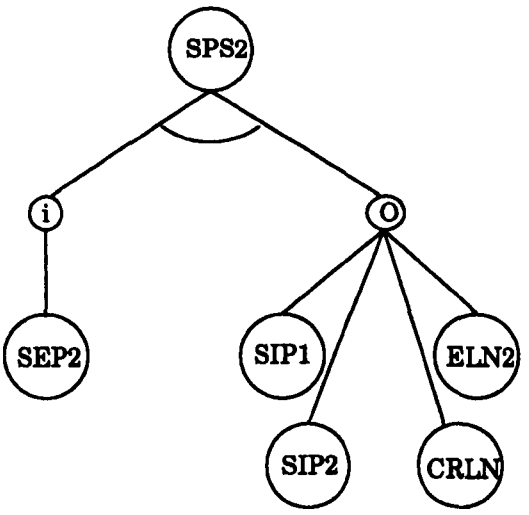


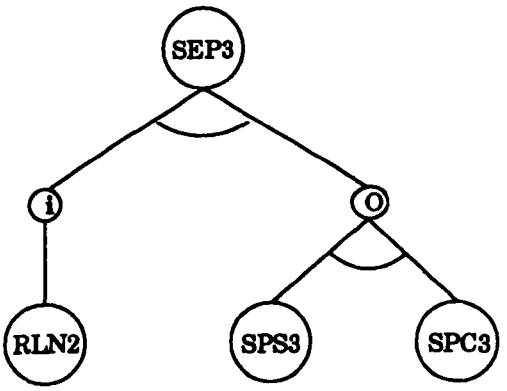

Separator 1 and Skimmed Milk Ouputs




Separator 2 and Skimmed Milk Ouputs




Separator 3 and Skimmed Milk Ouputs

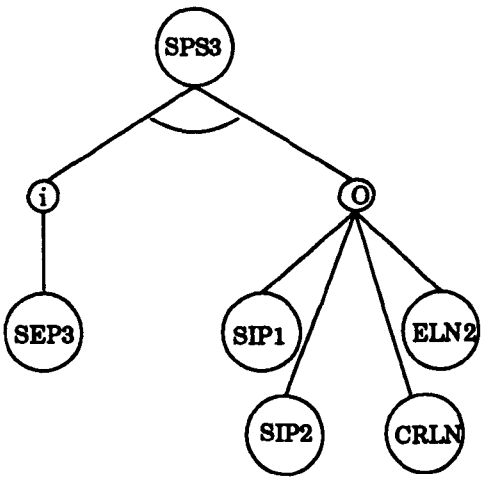
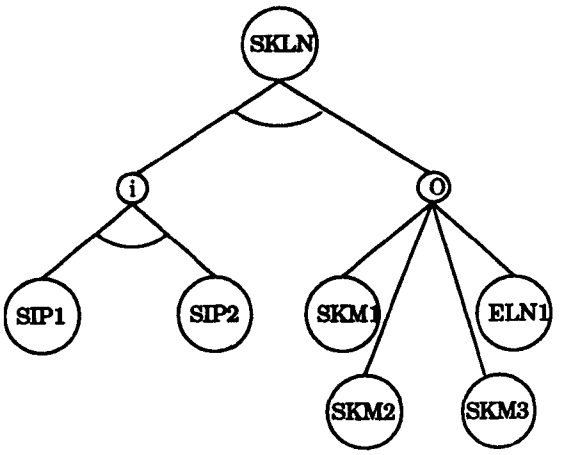
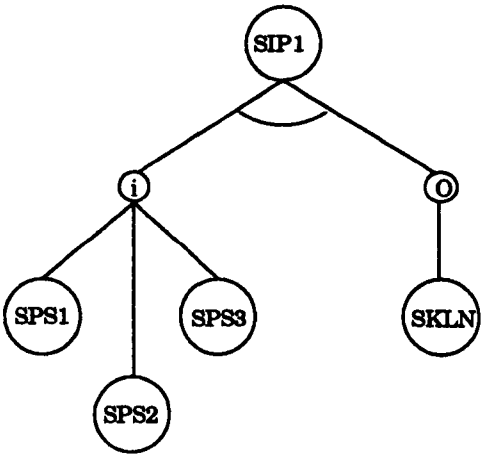


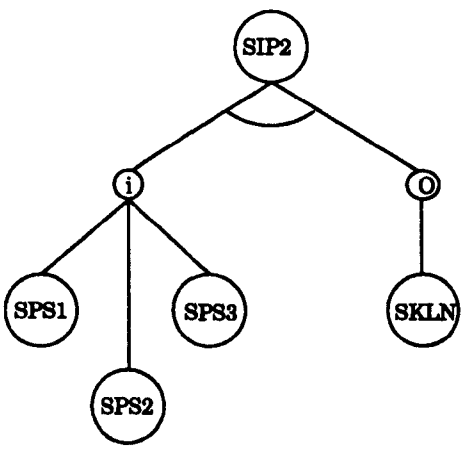

Figure 7.4.b. Separators and Skim Outputs




Skim Line

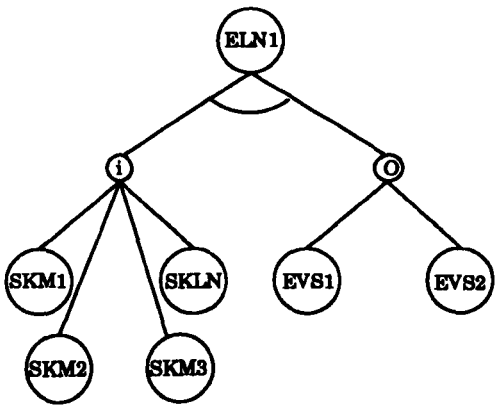



Skim Line Inputs 1

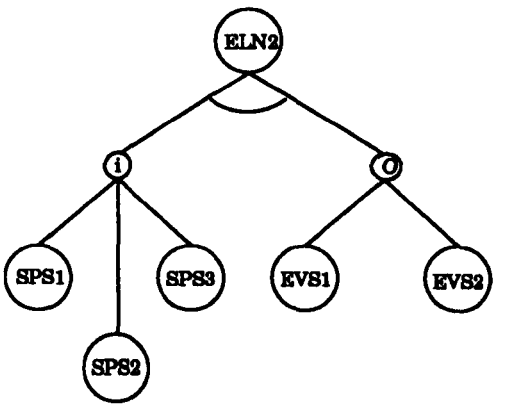



Skim Line Inputs 2

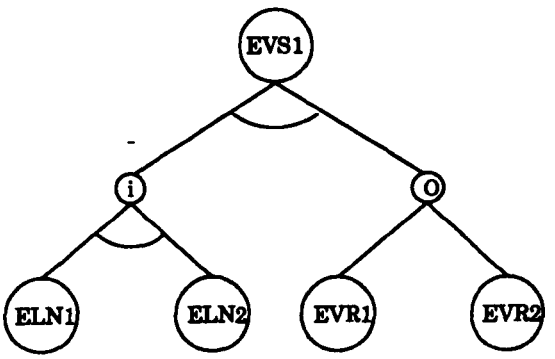
Figure 7.4.c. Skim Line to Skim Storage and Inputs to Skim Line



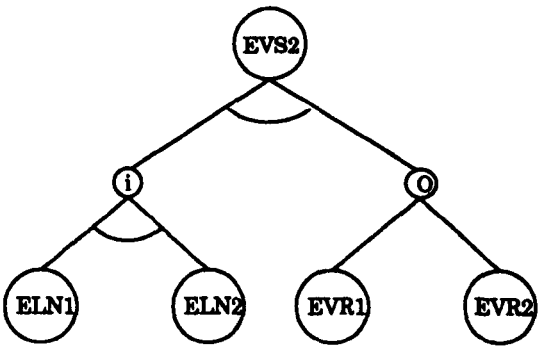

Evaporator Skim Line 1




Evaporator Skim Line 2




Evaporator Skim Storage 1




Evaporator Skim Storage 2

Figure 7.4.d. Evaporator Skim Storage Lines and Storage Silos

a mechanism for testing the feasibility of a connection bewteen two plant items from either direction. This is described in detail in Section 7.5 on the availability of plant items for routing purposes.

In order to infer what the constraints on the feasible connections of a plant item are the structure of a plant item representation must also include its current direct connections to other plant items. In order to allow this to be done from each item's perspective of the plant, these current connections must be split into current input connections and current output connections. Therefore, to fully represent the dynamic connectivity constraints in a plant network, each plant item has an attribute to hold its static configuration as an AND/ OR structure, an attribute to hold its current input connections, and an attribute to hold its current output connections.

## 7.4.SPECIFICATION OF RESOURCES FOR PRODUCTION ACTIVITIES

There are often a number of alternative ways of carrying out a particular production activity. This could be because of potential alternative choices for plant items at a batch reaction stage, or product transfer, or semi-continuous process or a CIP route. There may be the possibility of running two or more routes in parallel, or carrying out a number of reactions in parallel for a given activity. Having determined a classification scheme for production activities, and the plant items in a batch process plant, a representation scheme which would allow the specification of choices for plant item allocation to these production activities is required.

### 7.4.1.Specification of Resources for Batch Reaction Activities

The alternative ways for carrying out a batch reaction activity are represented by the set of reaction vessels which could be used. This also potentially represents the number of reactions which could start in parallel or overlap.

This is actually physically constrained by the number of routes which are available to supply materials required for a reaction and the number of routes which are available to remove the products of a reaction. The number of reactions which can start within a given timespan is also practically constrained by other factors such as **No Wait** reactions in which the products of the reaction must be removed immediately, thus requiring downstream plant items to be available. However, the basic physical specification for a batch reaction activity is the set of vessels which could be used to carry it out, which can be represented as an attribute of the batch reaction activity class.

### 7.4.2.Product Routing Specification for Semi-Continuous/ Transfer Activities

The number of items involved in a process route may vary considerably, ranging from a simple transfer between two immediately adjacent storage or reactor vessels, to a complex semi-continuous processing activity involving several plant items linked both serially and in parallel. The generic structure for the specification of a product transfer or semi-continuous processing activity must take into account the variation in length and complexity of the routes that could be used to carry it out, and the potential for two or more routes to be operating in parallel.

A natural boundary between each of a series of transfer/ processing activities involved in the manufacture of a product can be seen as the input source(s) of product(s), and output sink(s) for product(s) for each activity. There must be something to move or process the product from the input(s) to the output(s), so a very basic route for a transfer/ processing activity must involve at least one source, one processing unit, and one sink. There may well be choices of plant items for some of the positions, and these can be simply represented as a list of plant items for each position. The representation of sources or sinks for a route can include queues from which a transport entity can be taken as the actual source or sink if there is one present. This three position structure can

therefore form the basic product route. However, as described earlier, a route may involve considerably more positions than this, linked both serially and in parallel. Therefore, a more flexible approach to describing the routing in the plant is required. This has been achieved by incorporating a mechanism by which the basic routing structures, which will now be referred to as sub-routes, can be linked together to form long "chains" of route positions with the potential for cross linking if necessary. The process activity structure also needs to allow for the number of routes that could run in parallel, which is achieved by enabling sets of sub-routes to be associated with a particular activity. This generic activity/ routing structure is shown in figure 7.5.

Using this linked sub-route structure, the routing for product transfer and semi-continuous processing activities within a particular plant can be specified. The maximum number of routes which may need to be specified for an activity is entirely dependent on the structure of the plant network and the potentially feasible connections between plant items. In specifying the routing for activities within a plant it is possible that a plant item may not belong exclusively to a particular route and may have to be shared among several routes serving a particular activity or indeed several activities as well. Also not all of the plant items which make up the choices for a route position need to have a feasible connection with all the choices in an adjacent route position. The route description structures are flexible in that they allow the user to specify only those routes that they want to run, or are allowed to run by some other control structure such as a low level process control system for example, (even if more are physically possible). The structure is robust, both in terms of the numbers of routes that are described and the units that can go in them, as long as the structural configuration of the plant has been described correctly. Figure 7.6a and Figure 7.6b. show the representation of the choices of plant item for the three routes which could carry out the process routing for the skimmed milk component of the **sep-milk-ev-skim** activity. This also illustrates the way that a number of basic route structures can be linked

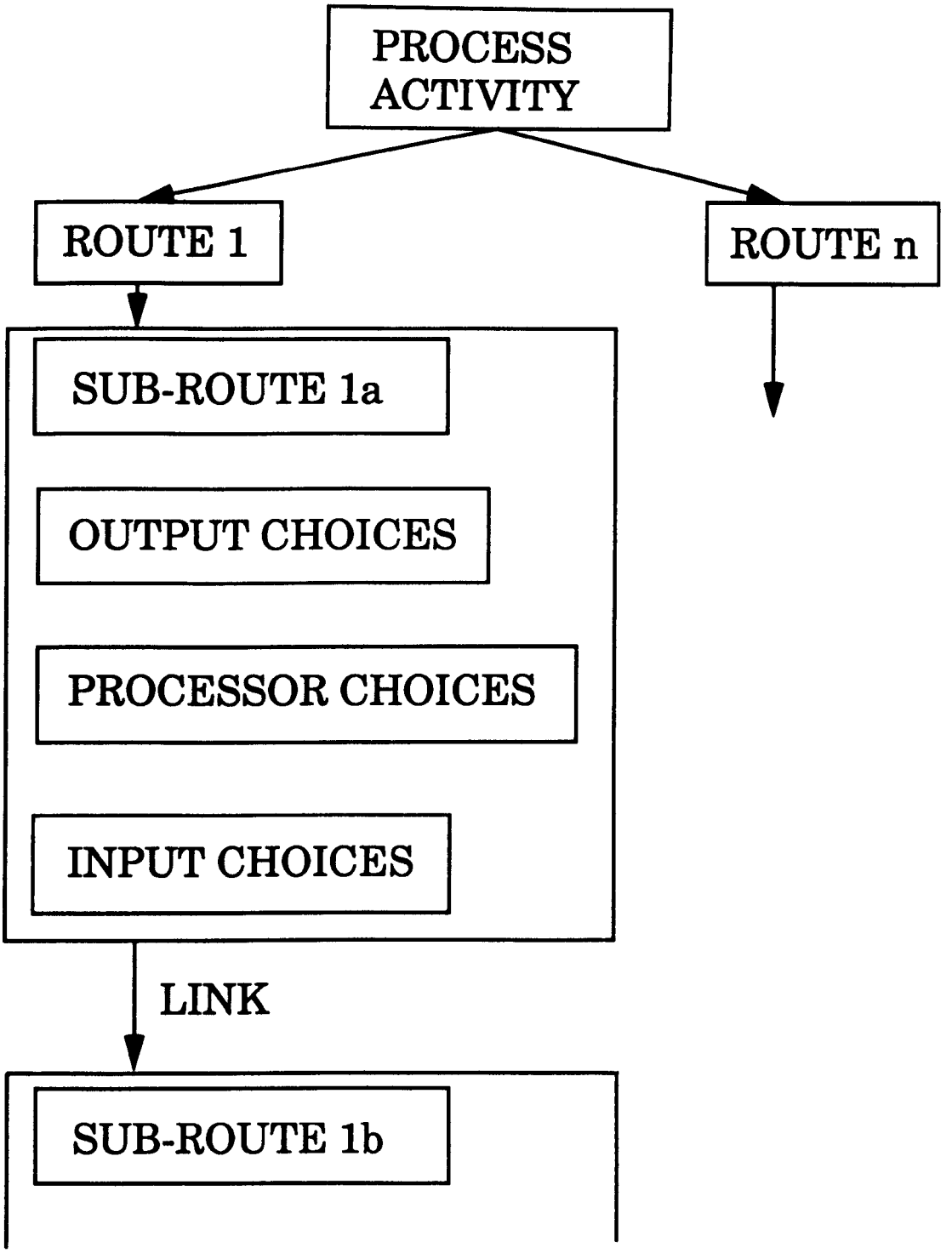
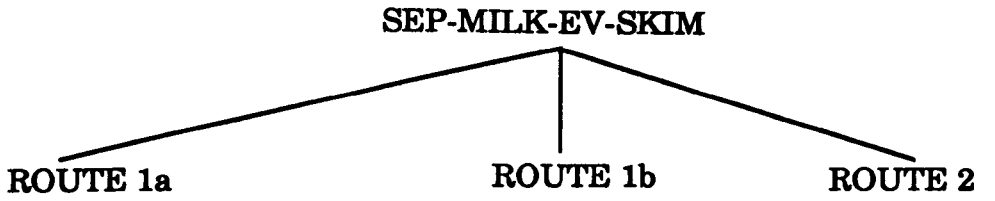



Figure 7.5. Generic Activity Route Structure




SUB-ROUTE - E1SL
OUTPUT CHOICES - EVS1, EVS2
PROCESSOR CHOICES- ELN1
INPUT-CHOICES - LK09

Plant Item Choices

Link between Subroutes

SUB-ROUTE- E1LN
OUTPUT CHOICES - LK09
PROCESSOR CHOICES - SKLN
INPUT CHOICES - LK08

SUB-ROUTE - E1SI
OUTPUT CHOICES - LK08
PROCESSOR CHOICES - SIP1, SIP2
INPUT CHOICES - LK07

SUB-ROUTE - E1SS
OUTPUT CHOICES - LK07
PROCESSOR CHOICES - SPS1, SPS2, SPS3
INPUT CHOICES - LK06

SUB-ROUTE - E1PR
OUTPUT CHOICES - LK06
PROCESSOR CHOICES - SEP1, SEP2, SEP3
INPUT CHOICES - LK05

SUB-ROUTE - E1ST
OUTPUT CHOICES - LK05
PROCESSOR CHOICES - RLN1, RLN2
INPUT CHOICES - MSL1, MSL2, MSL3, MSL4

All the choices shown for Route 1a are common to Route 1b, and the earlier route choices and evaporator silos are also common to Route 2

Figure 7.6.a. Separate Milk for Evaporator Skim Route 1a


SUB-ROUTE - E2ST
OUTPUT CHOICES - EVS1, EVS2
PROCESSOR CHOICES- ELN2
INPUT-CHOICES -LK12

SUB-ROUTE - E2SS
OUTPUT CHOICES - LK12
PROCESSOR CHOICES - SPS1, SPS2, SPS3
INPUT CHOICES - LK11

SUB-ROUTE - E2PR
OUTPUT CHOICES -LK11
PROCESSOR CHOICES - SEP1, SEP2, SEP3
INPUT CHOICES - LK10

SUB-ROUTE - E2ST
OUTPUT CHOICES - LK10
PROCESSOR CHOICES - RLN1, RLN2
INPUT CHOICES - MSL1, MSL2, MSL3, MSL4

Figure 7.6.b. Separate Milk for Evaporator Skim Route 2

together to form chains representing a full process route. It can be seen that a considerable number of the choices across positions in the routes are the same. However, this level of abstraction of the routing information is not enough to indicate whether or not process items can actually be shared across the routes. This is one of the particular problems in a batch process plant; the allocation of limited shared resources across a number of activities depends on the configuration of the plant network, and the dynamic connectivity constraint which was described in Chapter 6. The determination of plant item availability for an activity subject to this constraint is described in detail in section 7.5. When the representation of the routing for the separation activity is expanded to include the cream component as well as the skimmed milk component, then the route branches after the position representing the choice of separators. There may be a number of potential routes for this part of the activity. One composite route for both parts of the activity is shown in Figure 7.10.a. illustrating the branching after the choice of separators to the cream standardising tanks shown in Figure 7.1 (the Minsterley process flow diagram). Figure 7.10.b. shows the AND/ OR configuration of the cream standardising tanks involved as the sink for the route involved with the cream component.

As described in Chapter 4 constraint propagation is an effective technique for ensuring that the consequences of actions taken which result in a change in one part of a model are propagated to all other parts of the model that may be affected. In the case of the routing structures, as connections are set up between plant items the effects must be propagated through the route and taken into account by other routes in the model when they are being set up. A constraint propagation mechanism that accounts for current connections in the plant has been developed for this purpose. It ensures that no more routes are ever set up than is physically possible, and only those items which are actually physically available will be able to be set up in those routes. This

mechanism is the key component of resource allocation to routes and is described in detail in section 7.5 on resource allocation.

### 7.4.3.Cleaning Activity Cleaning In Place (CIP) Routing Specification

As carrying out a cleaning activity involves linking together a number of plant items, it can be viewed as setting up a cleaning route in the same way as setting up a process route. In a typical batch plant there may be a number of CIP "sets", which are a set of cleaning units which supply rinse water, detergent, and sterilising solution to a particular cleaning circuit in the plant. Each circuit can be configured into a number of different routes to clean different combinations of plant items. Sometimes the CIP set is simply the balance tank of a process plant item such as a pasteuriser to which the cleaning fluid is added by a plant operator, but the function is still the same as a dedicated CIP set, so it does not need to be modelled differently.

A hierarchical structure can therefore be defined for CIP routing within the model, based on the same activity/ route structure as for process routing. This hierarchical structure is defined as CIP Circuit/ CIP routes. The individual units which make up a CIP set do not need to be represented; it should be sufficient at this level of detail to represent a circuit and the CIP routes it supplies. Each route will be used to clean a specific plant item or items. As once again it is important to build up routes of a variable length, and there may be choices for the plant items which could be put into a particular route position, the same basic three position structure that is used to build up a representation of the process routing can also be used to represent the CIP routing within the plant. Hence in the specification of a CIP route each position can contain one item or a list of choices which could fill that position. Figure 7.7. shows the generic structure for representing a CIP circuit and its associated cleaning routes. A number of different plant items may be cleaned by different routes on the same circuit supplied by a particular CIP set.

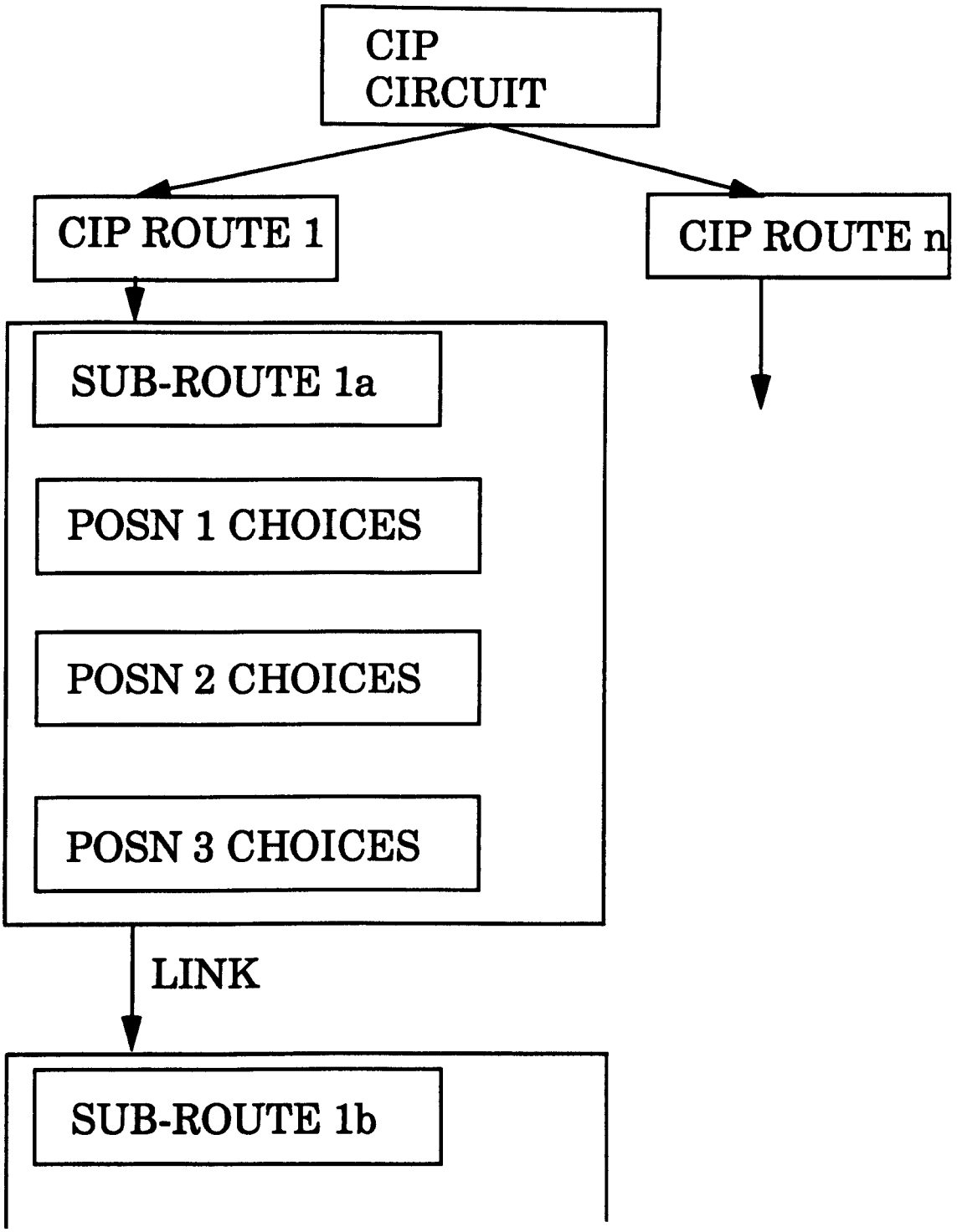



Figure 7.7. Generic CIP Circuit/ Route Structure

The CIP routing to carry out a cleaning activity on one or more plant items may impose constraints on the availability of other plant items for process routing in the plant. Therefore, the representation of plant item connectivity needs to include CIP connections so that they can be included when determining the dynamic connectivity constraints on the availability of a plant item. The cleaning of plant items cannot be treated separately from processing in this respect.

## 7.5.AVAILABILITY OF PLANT ITEMS FOR USE IN PRODUCTION ACTIVITIES

Whether a choice of plant item is currently available to be used in a production activity is determined by the physical constraints on its use. Whether it is actually used depends on preferential factors such as whether it has the most capacity, or has already been used as part of the current production process. However, it is very important that the physical constraints on plant item availability are not breached before any other factors are considered or the schedule produced by the BPS will be invalid. The availability of a particular plant item for a product transfer, semi-continuous process or CIP activity depends on the activity state of the plant item, its current connections in the plant network and the availability of the items to which it could connect in the route. A batch reaction activity must be carried out in conjunction with some material input activity which involves routing, so it is the choice made for the routing activity which will determine which batch reactor plant item is actually used for the reaction activity. If the batch reaction is looked at in isolation, then the choice of items would be erroneously constrained only by their activity state.

### 7.5.1.Availability Based on Activity State

A plant item from a particular class can have a number of activity states defined for it, and its behaviour in terms of the way that it can progress from state to state can be defined in terms of an activity cycle. The generic activity cycles and states for the classes of plant items defined earlier to model the Minsterley factory have been determined based on data gathered at the Minsterley plant and descriptions of plant item behaviour in the batch processing literature. At any stage of its activity cycle, a plant item from a particular class will be in one of its defined states. For example a continuous process plant item could be processing, cleaning, waiting dirty or waiting clean. A vessel could be filling, emptying, filling and emptying, cleaning, waiting empty and clean, or waiting empty and dirty. Some of these states will physically preclude a plant item from being a potential choice for an activity. Some of these states may make it undesirable to use the choice for the activity but do not physically preclude its use. In determining which items are currently available as choices for a production activity only those states which physically preclude the item should be considered as unbreachable constraints. Undesirable states must be subject to preferential considerations only when making the final selection of plant items for activities. The only activity states which would physically preclude a plant item from being used for a batch reaction independently of any routing involved would be if it was cleaning, physically empty or out of service. The determination of plant item availability for routing taking into account activity state and connectivity is described in the next section.

### 7.5.2.Determination of Plant item availability for Process and CIP Routes

The route representing a particular activity in the plant can be viewed as a network structure at a level above the representation of the plant itself. Each node in this network represents one of the positions in the route, with the arcs

in the network representing direct connections between route positions. Each node may hold a number of potential plant items which could be used for this position in the route. What must be determined is whether they are physically currently available to be assigned to the route; this depends on their activity state, and whether they can make a connection to any of the plant items which are potential choices in immediately adjacent nodes in the route network. Making a connection depends on whether a feasible connection to any of the choices in an adjacent route position is defined, and on the additional connectivity constraints imposed on the plant item by the current connections in the plant network.

Determining which elements are currently available as a potential choice for a route position taking into account the current physical constraints which exist can be viewed as a Constraint Satisfaction Problem (CSP). This is because in order to find a feasible configuration for a route, each node representing a position variable must be assigned a single plant item which does not breach node specific constraints and the connectivity relation **CAN-CONNECT-TO** represented by the arcs between the nodes in the network. One of the approaches used to aid in the solution of a CSP is to make the network representing the problem consistent to a certain degree with respect to the constraints acting on the network. Consistency means that the choices at a node which breach a particular constraint have been removed from the range of potential choices, so an infeasible choice cannot be made. The degree to which a network is made consistent is referred to as its n-ary consistency. A network may be made consistent with respect to constraints which only apply to the nodes in the network; (unary consistency), or with respect to relations representing constraints between immediately connected nodes; (binary consistency).

Normally, in a CSP, even if the network is made node and arc consistent for binary relations, it does not necessarily mean that a solution to the problem

exists; infeasibility can arise because of path inconsistency due to relations between nodes that are not directly linked. However, in the case of the routing representation of the batch plant network and the availability of plant items that can go into a route we are only concerned that immediate plant item to plant item connections are feasible. Therefore, if the route network is made node and arc consistent so that at least one choice exists for each position which can connect with at least one choice in the immediately adjacent positions, it will be possible to assign at least one set of plant items which can connect to each other.

On this basis, at any point in the model execution when it is desired to attempt to set up a process or CIP route unary and binary constraint checks should be done on the route positions in conjunction with constraint propagation procedures (Mackworth [87]) in order to check the availability of plant items. This will make the route node and arc consistent and will mean that, in respect of the physical state of the plant, all the items remaining as choices are currently available to be assigned to the route. The intention at this stage is not to make connections, but simply to determine if connections can be made. It is quite possible that during this initial checking procedure all the potential elements will be lost from a route position because they fail the constraint checks, making it impossible to configure this route for the activity.

Making a route consistent with respect to the physical constraints does not mean that any combination of the available plant items can be assigned to the route. In a route specification it may well be that not all the elements in each position can connect to all the elements in an adjacent position, although the overall structure of the plant network makes them all valid potential choices for the route. In addition, when plant item assignments are made to the individual route positions it introduces further constraints on the choices that can be made for the remaining positions in the route. The important point about ensuring consistency at this stage is to check that at least one

configurable route exists and explicitly defining the remaining choices based on the current physical constraints. This will be discussed further in section 7.6. on dynamic route configuration where the procedures for making actual choices of plant items and the making of connections are described. There are three parts to making a route consistent with respect to the constraints on physical availability; unary constraint checking, binary constraint checking, and constraint propagation. The following sections will describe the implementation of these parts in a procedure developed as part of this research for the BPS.

Unary Constraint Checking

The first set of physical constraints checked are those based on the activity states of the plant items. These can be treated as unary constraints because they do not involve a relation with another plant item. Making the route node consistent with respect to the unary constraints is straightforward in that all that is required is to make a single pass through the network, removing those plant items amongst available choices which fail the constraint checks. Mackworth [87] describes the NC-1 procedure for making a network node consistent which is implemented in the BPS as follows:

**procedure** NC-1($P_{i,n}$)
  **for** each route position $P_i$, from i=1 **until** i=n **do**
    remove any plant item choice from $P_i$ which fails one of the unary constraint checks
**end** NC-1 **procedure**

The BPS currently checks the following unary constraints based on activity state which preclude the physical availability of a plant item for process routing purposes.

1.A plant item in any position cannot be cleaning.

2.A plant item in any position cannot be out of service.

3.If the plant item is a storage or reactor vessel in an output position of a route, it cannot be full or it has been full but not fallen back to a "threshold" level where product can reasonably be put back into it.

4.If the plant item choice is a storage or reactor vessel in an input position of a route it cannot be empty or it has been empty but not yet refilled to a "threshold" level where product can reasonably be taken out of it.

5.A queue cannot be empty.

The checks done in cases 3 and 4 use the notion of falling and rising "triggers" or threshold values, in conjunction with the maximum and minimum capacity limits of vessels. These triggers were described in Chapter 6 and are used to set user defined values on finite capacity vessels to stop the model going into an undesirable stop/ start cycle around one of the finite limits. For example, consider that a vessel could be used in two routes as a sink for one and a source for the next. If the rate of the first route was faster than the rate of the second route, then the vessel would eventually hit its maximum finite capacity limit, and cause the route to shutdown. However, if the vessel had no falling trigger then as soon as the second route removed a small amount of material from the vessel it would become eligible to be placed back as the sink in a configuration for the first route. It would then fill up to maximum capacity again, and cause the first route to shutdown again almost immediately after it had started, potentially leading the BPS into a long cycle of the route stopping and restarting. By including these rising and falling triggers in the unary constraint checks, the potential for the BPS to go into an undesirable start/ stop cycle is removed as long as the triggers are set to suitable values.

The unary constraints checks done for cleaning purposes are as follows:

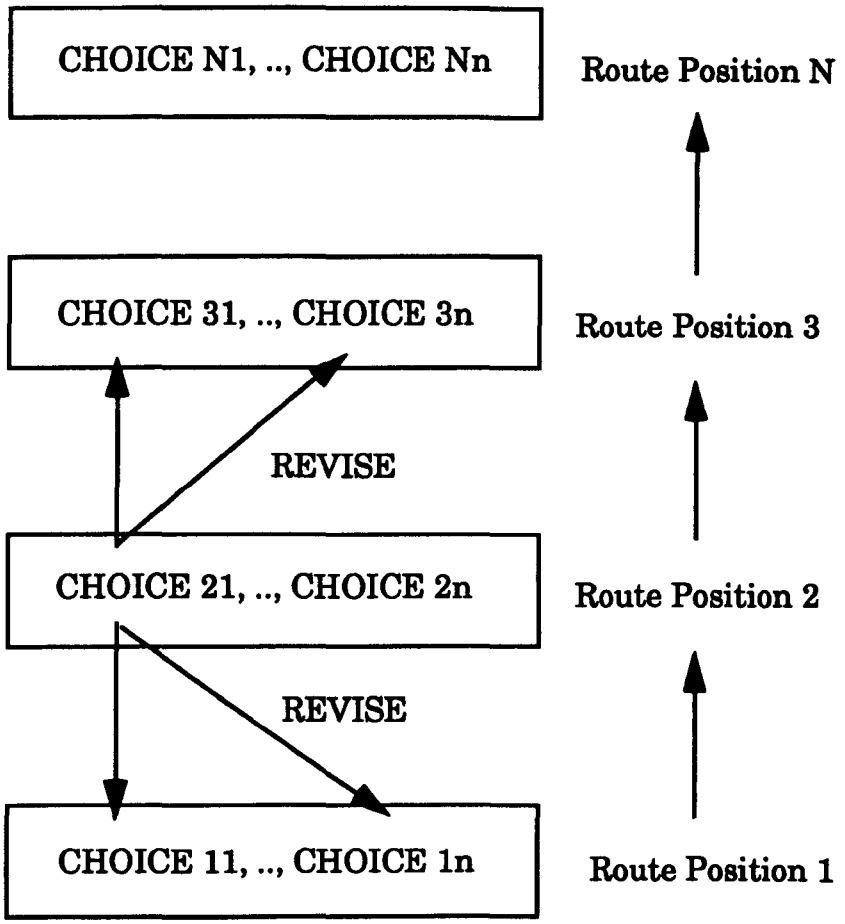
1.The plant item cannot be processing.

2.The plant item cannot be cleaning.

3.The plant item cannot be filling, emptying, or filling and emptying.

Binary Constraint Checking

The relation tested between the choices in adjacent nodes is **CAN-CONNECT-TO**. If a plant item **CAN-CONNECT-TO** at least one choice in each of its immediately adjacent route positions, then it is still physically available as a choice for the route under the current constraints that exist. The way that the search would progress for a simple serial route with no branching is illustrated in figure 7.8. The input/ output ports of an element's configuration are used to determine which adjacent position in the route to test for feasible connections. In the convention which has been adopted, the positions in a serial route are numbered upwards from the route source position to the route sink node. In considering whether a plant item is available as a choice for some position $P_i$, its potential output connections are checked against the entities which are current choices in position $P_{i+1}$, and its potential inputs are checked against those entities which are current choices in position $P_{i-1}$. Entities in the first position of the route will only have their potential outputs checked, and entities in the final position in a route will only have their potential input connections checked. The actual form of the test of the **CAN-CONNECT-TO** relation between two plant items depends on their class and uses the data contained in their configuration attributes and current connection attributes. The configuration of the plant item describes the constraints on its feasible connections in an "empty" plant. The current connection attributes describe the additional constraints on the availability of an item due to its use in other routes. As described earlier, the perspective of the connection is important, because the configuration of individual plant




The search moves up through the route positions from 1 to N as follows:

1. REVISE CHOICE 11 to CHOICE 1n against Choices in Position 2.
2. REVISE CHOICE 21 to CHOICE 2n against Choices in Position 1 and Position 3.
3. REVISE CHOICE 31 to CHOICE 3n against Choices in Position 2 and Position 4.
4. REVISE CHOICE N1 to CHOICE Nn against Choices in Position N- 1.

Figure 7.8 Single Pass Binary Constraint Check

items means that even if a connection appears currently feasible from the point of view of one plant item in a pair, it may not be currently feasible from the point of view of the other plant item. In order to account for this effect, the **CAN-CONNECT-TO** relation must be tested from the perspective of both plant items in two adjacent nodes in a route which is being searched.

## The CAN-CONNECT-TO Cases

Plant items such as vessels, semi-continuous process plant items and process lines are all static and it is the product that moves through them in the system. Transport elements (such as bulk milk transporters) move through the system either waiting in queues or being connected to static plant items. These differences affect the cases of the test.

There are three main cases tested in the binary constraint check between two static plant items for the **CAN-CONNECT-TO** relation, with conditions about feasible and current connections determining whether the relation is satisfied. The basic premise of the tests is that if both items have a potentially feasible connection between them as described by their configuration, this may or may not be currently feasible depending on the current connections in the plant. In considering whether a plant item can actually make a feasible connection to another plant item in an adjacent position in a route, the degree to which current connections act as a constraint varies depending on the class and configuration of the plant items involved. If the plant item in the adjacent position already has a current connection to this plant item in another activity/ route then the connection may still be feasible in this route. Process lines and storage vessels can share the same current connection across a number of routes. For example, a storage vessel may be filled via a single input line, which is taking input from two separate sources. However, semi-continuous plant items cannot be used in a number of different production activities at the same time, so a current connection in one activity/ route

would make them unavailable for any other activity/ routes. In some cases the AND/ OR configurations of the plant items involved are such that all the current connections of the items must be checked to determine whether the potential connection is still feasible. In other cases, the AND/ OR configuration of the plant items means that the connection is always feasible in terms of the **CAN-CONNECT-TO** relation so current connections do not need to be considered. The test of the **CAN-CONNECT-TO** relation takes these factors into account.

The following cases test whether Entity1 and Entity2 as potential choices for route positions $P_i$ and $P_{i+1}$ respectively can make a connection. The connection must be feasible as an **input** to Entity1 and as an **output** from Entity2. The first case tests whether the two potential choices already have a connection between them. Because some plant items can be shared across routes, then they may be able to form part of this route. In the remainder of the cases, the severity of the constraining effect of any current connections to other plant items is tested where a feasible connection exists. Where a feasible connection exists between two items which both describe these connections by an AND representation then the constraining effect of other current connections is much less severe than when OR connections must be considered. This is illustrated in the examples in Figure 7.9a. and Figure 7.9b.

Case 1 only applies to process lines and storage vessels which can share the same connection across a number of routes.

1.IF Entity1 has a current input connection to Entity2
THEN the relation holds.

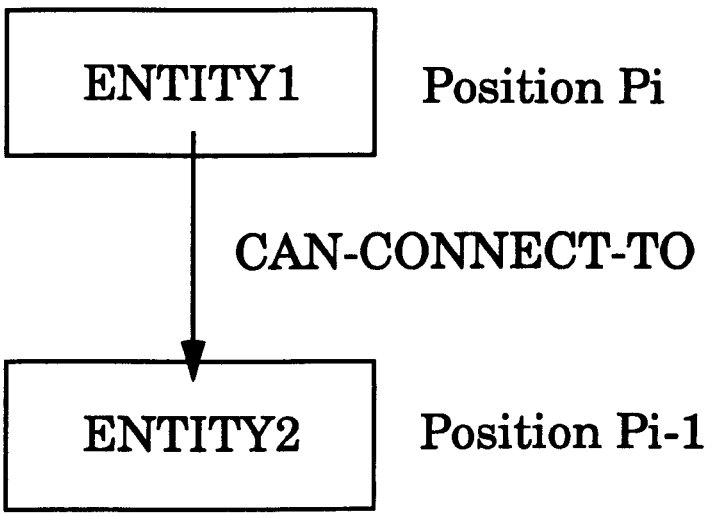
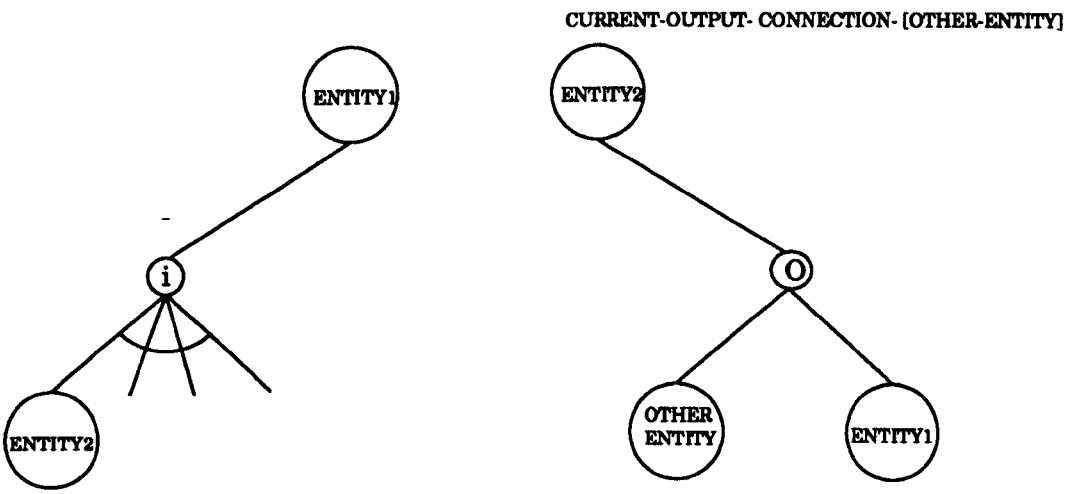


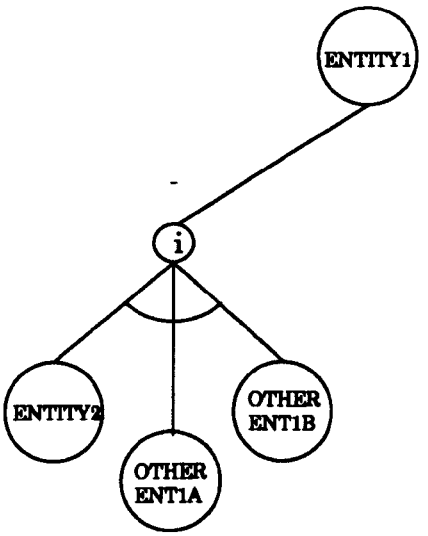

Figure 7.9. CAN-CONNECT-TO Test Between ENTITY1 and ENTITY2



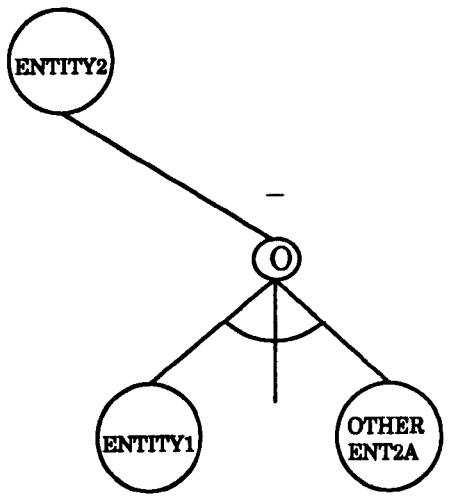

Case 2a) Entity1 has Entity2 as a potential AND input, and Entity2 has Entity1 as a potential OR output. However, the test of the relation CAN-CONNECT-TO will fail because Entity2 already has a CURRENT-OUTPUT-CONNECTION to Other-Entity

Figure 7.9.a. Example test of CAN-CONNECT-TO Case 2a

CURRENT-INPUT-CONNECTIONS- [OTHER ENT1A, OTHER ENT1B]




CURRENT-OUTPUT-CONNECTIONS- [OTHER ENT2A]




Case 2b) ENTITY1 has a potential AND input connection to ENTITY2, and ENTITY2 has a potential AND output to ENTITY1. In this case the CAN-CONNECT-TO relation holds even though both entities already have current connections in the directions being tested.

Figure 7.9.b. Example test of CAN-CONNECT-TO Case 2b.

Cases 2 and 3 apply to process lines, storage vessels, and semi-continuous process plant items.

2a)IF Entity2 is on the inputs of Entity1 as an AND connection
AND Entity2 has OR outputs
AND Entity2 has no other current output connection
THEN the relation holds

2b)IF Entity2 is on the inputs of Entity1 as an AND connection
AND Entity2 has AND outputs
THEN the relation holds

3a)IF Entity2 is on the inputs of Entity1 as an OR connection
AND Entity2 has OR outputs
AND Entity1 has no other current input
AND Entity2 has no other current output
THEN the relation holds

3b)IF Entity2 is on the inputs of Entity1 as an OR connection
AND Entity2 has AND outputs
AND Entity1 has no other current input
THEN the relation holds

Another set of these cases looks at the output side of Entity1 when testing its connections against its adjacent output route position.

Transport entities may be used as source or sink elements for process routes which provide input and output to the system or they may be used within the system to provide interstage product transfer where a direct link between two plant items does not exist. In both these cases they will typically be making a connection with a static plant item. However, it would not be practical to

represent a large number of transport entities as potential connections to a static plant item. Transport entities move through the system via queues associated with static plant items. Therefore the queues in the BPS can be represented as part of the configuration of the static plant items. When the **CAN-CONNECT-TO** relation is tested between a static plant item and a queue listed as a choice in a route position it simply checks whether it has the queue as a possible source of a transport entity on its AND/ OR configuration. The status of a queue as an available choice only requires a unary constraint check on whether it does or does not have any entities in it.

The test of the **CAN-CONNECT-TO** relation between one Position $P_i$ and one or two adjacent positions $P_{i+1}$ or $P_{i-1}$ in the route is embodied in a version of Mackworths [87] REVISE procedure.

**procedure** REVISE($P_i$,$P_{i-1}$,$P_{i+1}$):
  **for** a route position $P_i$ **do**
    **if** there is a position $P_{i-1}$ **then**
      **for** each potential choice in $P_i$ **do**
        test the relation **CAN-CONNECT-TO** against potential input connections in $P_{i-1}$
        **if** the potential choice in $P_i$ fails the test **then** remove it from $P_i$ and set CHANGES to **true**
        **endif**
    **endif**

    **if** there is a position $P_{i+1}$ **then**
      **for** each potential choice in $P_i$ **do**
        test the relation **CAN-CONNECT-TO** against potential output connections in $P_{i+1}$
        **if** the potential choice in $P_i$ fails the test **then** remove it from $P_i$ and set CHANGES to **true**
        **endif**
    **endif**
**end** REVISE **procedure**

CHANGES is a Boolean variable indicating whether any change occurred to the range of choices available for a route position as a result of testing the **CAN-CONNECT-TO** relation. It is used to determine the amount of constraint propagation which must occur in the overall procedure to make the route consistent as described in the next section.

Constraint Propagation

The REVISE procedure can be applied to each position in a route starting from position $P_1$ and and moving through to the last position in the route.

Immediately after carrying out the REVISE procedure the residual choices in a position $P_i$ will remain valid provided that there is no further change in the availability of plant items in positions $P_{i+1}$ onwards. However, as described by Mackworth [87] in respect of networks generally this assumption will often not hold because there will be changes to nodes checked later in the network which affect the consistency of the nodes checked earlier. If there are changes in the availability of items in $P_{i+1}$ after $P_i$ has been revised, then the availability of items in $P_i$ and all positions preceding it can no longer be guaranteed. In general each time a range of values applicable to a node in a network is constrained by a binary relation the effects must be propagated through at least some of the other nodes in the network. Mackworth [87] describes a number of multi-pass algorithms to do this known as AC-1, AC-2, and AC-3. In the most basic case of the AC-1 algorithm each time a change is made to a node in the network on the basis of a binary relation constraint another full pass must be made through the network to propagate the effects of the change. When a pass is made through the network which does not result in a change then it is arc consistent. Mackworth [87] commented that the AC-1 algorithm would be very inefficient for large networks, where a change to a node might only affect a small portion of the rest of the network. The AC-2 and AC-3 algorithms were developed to address this problem by only rechecking affected parts of the network. In the case of the BPS as it is currently developed, the AC-1 algorithm has been used to test that the approach would work as it is the simplest to implement. Also, since the representation method used for the routes produces small networks consisting of only a few nodes inefficiency because of multiple passes through the whole network should not be a major problem.

The complete procedure based on NC-1, REVISE and AC-1 used in the model to make a serial route (with no branches) consistent with respect to unary constraints and binary connectivity constraints is as follows:

Stage 1.Carry out a unary constraint check on each route position through the NC-1 procedure and remove all plant items which fail the appropriate tests. If any route position becomes empty as a result of the tests, then exit with failure as the route cannot currently be configured.

Stage 2.Set CHANGES to **false** to indicate that no change has been made to the range of choices applicable to one or more route positions.

Stage 3. Commence a binary connectivity constraint check of the potential route choices remaining in each position through the AC-1 procedure starting at the input position $P_1$ to the route. For each position $P_i$ carry out the REVISE procedure against adjacent input and output positions as applicable with the following results:

3a).Remove any potential route choice from the position $P_i$ which no longer **CAN-CONNECT-TO** at least one plant item in both adjacent input and output positions $P_{i-1}$ and $P_{i+1}$. (The first position $P_1$ in the route is only checked against an adjacent output, and the last position $P_n$ in the route is only checked against an adjacent input).

3b).If a position $P_i$ has had one or more choices removed, then "post" the fact that a change has occurred by setting CHANGES to **true**.

3c).If a position $P_i$ is now so constrained that it has no potential choices, then exit from the procedure with failure as the route currently cannot be configured.

Stage 4.At the end of a pass through the route test whether any CHANGES have been posted as a result of the binary constraint check.

4a)If a CHANGE has been posted, then return to Stage 2 of the procedure and carry out another pass through the route for the binary constraint check to propagate the effects of the change.

4b)If no CHANGE has been posted, then exit from the procedure with the route now node and arc consistent.

If the checking of a route successfully exits from the procedure at Stage 4b) then for each of the remaining route choices there exists at least one configuration of the route including this choice and some set of the other remaining route choices. This guarantees that the configuration procedure for a route from this point on will not fail because of physical constraints. If the procedure exited at Stage 1 or Stage 3c) then at least one route position is now empty of choices so no way of configuring the route currently exists.

### 7.5.3.Additional Complexity in Routing

There may be production activities where the routing involves some branching. As far as making this sort of route consistent with respect to unary and binary constraints there is no difference from the basic procedure used for a serial route. However, the route representation and search mechanism required to carry out the procedure are more complex. For example, the production activity **sep-milk-ev-skim** involves two output products skimmed milk and cream. If the capacity of the plant for dealing with the cream output from separation is considered important then its routing must also be included in the representation as well as the skimmed milk routing. This involves expanding the route representation. The expanded route representation, including one potential branch route to cope with the cream output product, is shown in figure 7.10.a. The representation of the separator configurations

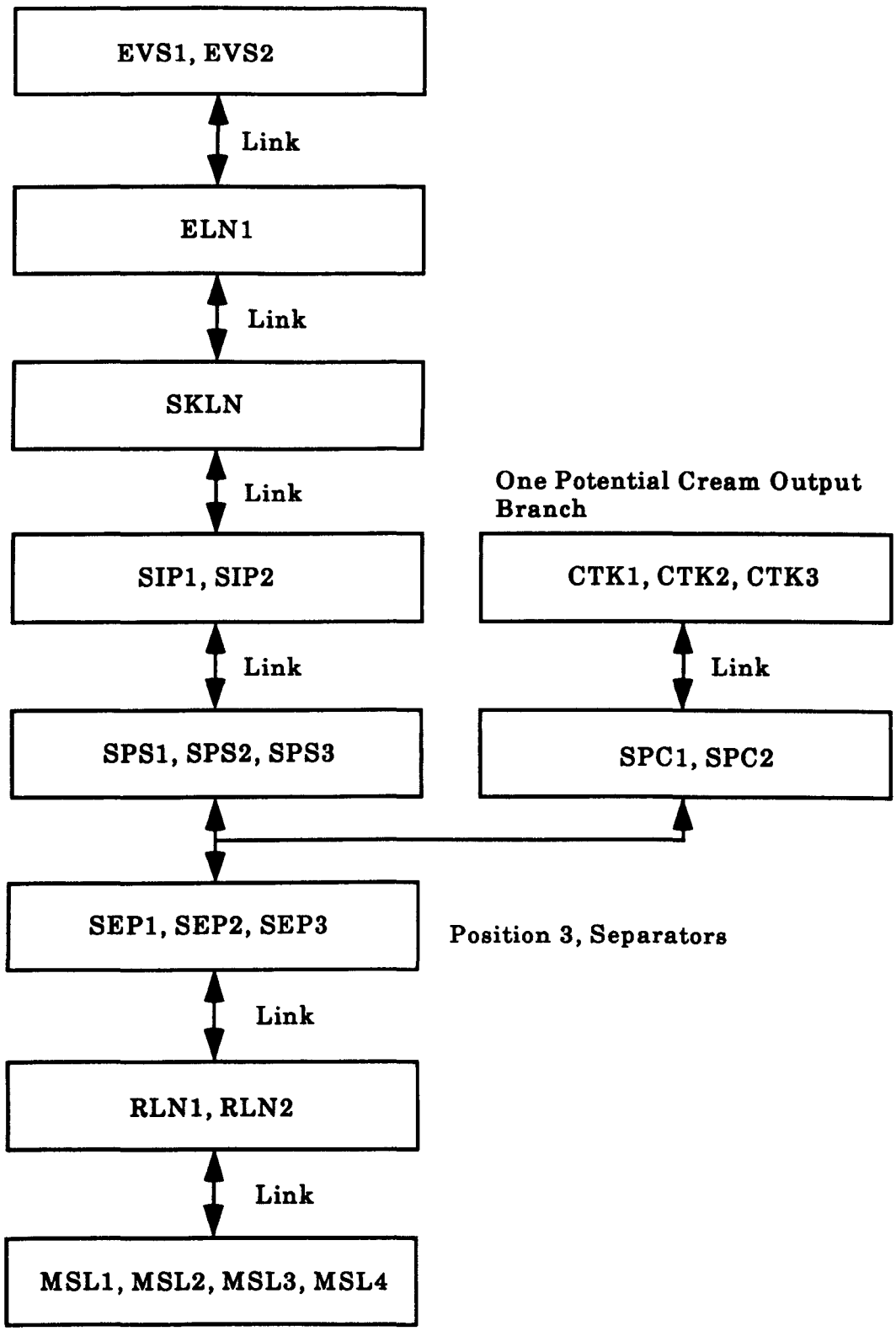
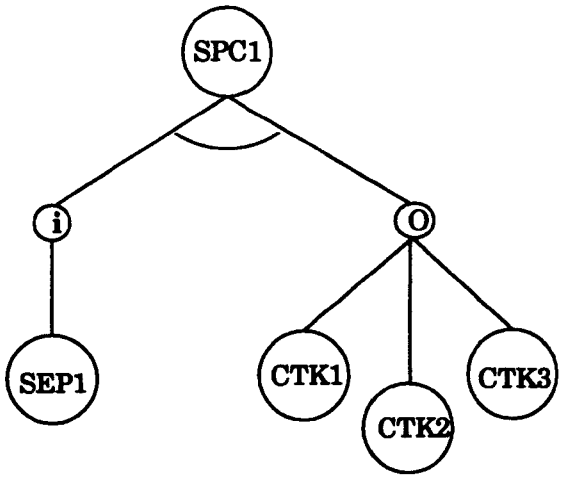



Figure 7.10.a. Initial Item Choices for Process Activity
sep-milk-ev-skim Route 1, including one potential cream output branch

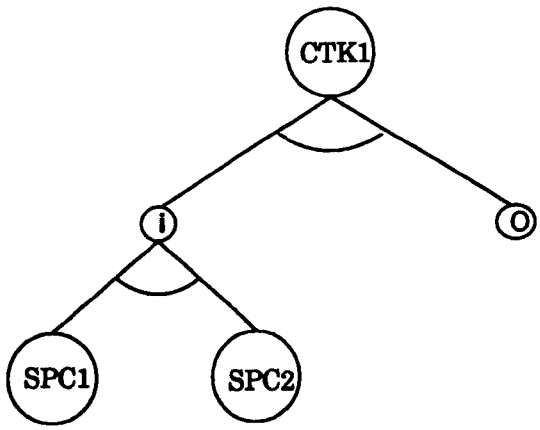
includes a cream output line and a skimmed milk output line as illustrated in Figure 7.4.b. These product output lines now make up the choices in the first position of each branch of the route. As illustrated in Figure 7.10.b. the AND/ OR configurations of the standardising cream tanks which are the sink choices for this route can only take the cream output lines from separators 1 and 2. and the cream output line from separator 3 is therefore not included as a choice for this branch. When the route consistency procedure is applied to this route it works in exactly the same way as before, with changes in the branches of the route being propagated back to positions checked earlier. For example, when the branch of the cream component route chosen is searched there will be no available output connection to a cream output line for separator 3 causing it to be lost as a currently available choice. The effects of this will be propagated back through the route, possibly causing further loss of choices earlier in the route. However, there are other potential routes for the cream portion of the separation activity which do involve separator 3, so an alternative branch route would have to be considered if it was desired to be able to use this separator for operational reasons. Therefore, some initial consideration has to be given to the reasons for an activity involving branching when determining which composite route to attempt to configure.

## 7.6.PREFERENTIAL CONSIDERATIONS

The activity state of plant items and their dynamic connectivity serve to represent the physical restrictions on the use of plant items for carrying out particular activities and as described above, these attributes can be used to determine what plant items are actually available for inclusion in a process or CIP route at any point in the operation of a plant. However, it is unlikely that a route which could be used for an activity will be so constrained at a given point in the operation of the plant that there will only be one choice available for each position in the route. Although the structure of the batch plant network makes routing a special case of the Constraint Satisfaction Problem




Separator 1Cream Output Line




Cream Tank 1 Inputs

Figure 7.10b. Example of Separator Cream Output and Cream Tank Input Configuration

in that a route only needs to have unary node and binary arc consistency to be sure that a configuration solution exists, not all route position choices will be compatible with all other route position choices as described earlier. Therefore, a route cannot simply be set up as a permutation of the available choices for each position. Resources must be allocated to route positions in some order, and each time a resource is allocated to a position from a set of choices, the constraining effects of this must be propagated to the other choices in the network. Preferential considerations about the allocation of plant items to route positions can be taken into account at this stage to find an acceptable configuration rather than just a purely feasible one. Preferential considerations are situation specific and are related to making choices about how to meet operating policies on the use of plant items and the movement of product through the plant. For example the age of product held in a vessel was determined as an important feature in the Minsterley plant, because the quality of product degrades with age. This is a common feature of many batch processes ranging from the situation where the product quality degrades relatively slowly over time to the **No Wait (NW)** situation, where the product resulting from a particular activity is so unstable that it should be used at the next stage of the process immediately. Another common operating policy which must be considered as a preferential constraint on the use of an item is whether a vessel should be able to be filled and emptied at the same time or just filled or emptied. The order in which plant items are allocated is important because of the additional constraints imposed on the remaining choices in a route as each choice is made. Therefore, the preferential considerations should include what the best order to allocate plant items to a route is. It is typically the case that when a plant supervisor or operator is setting up a route to process a batch of product they will be using some decision order based on the current state of the plant and their knowledge of the characteristics of the plant item or items. As described in Chapter 4 the allocation of plant items to a process route or CIP route using a decision ordering to guide the process can be viewed as a configuration problem under

resource availability constraints. On this basis the preferential allocation of resources to routes under dynamically changing constraints can be achieved by coupling a rule-based configuration approach with the network consistency and constraint propagation procedure to ensure that the availability of plant items for use in routing can always be correctly determined.

### 7.6.1.Dynamic Rule-Based Plant Configuration for Carrying out Production Activities

A powerful and flexible way to represent situation specific preferential considerations in a configuration problem is to use production rules as described in Chapter 4. They allow a route specific decision order to resource allocation to be represented in order to implicitly take the connectivity constraints into account which is an important factor in the outcome of the process. On this basis, having ensured that only those entities with the potential to connect to each other remain as route choices through the network consistency and constraint propagation procedure, user defined plant configuration rules for the preferential choice of route entities can be applied to determine the final configuration of a route. A sample of rules from the configuration rule-base for the Minsterley plant in its current form is listed in Appendix B. These rules represent how decisions about configuring a route are made based on the current system state and the resources which are available. The decisions about which plant items to allocate to a particular route are plant specific and incorporate things such as the different "fill and empty" policies for intermediate storage vessels in different areas of the plant. The rule format uses the convention from R1 [89] of defining as a context the active configuration goal being pursued, the conditions which must be matched and the consequences of the rule firing. The consequences will include the selection of an element and the next part of the configuration to be achieved through a sub-goal. A sub-set of the rules for configuring a route associated with an activity defines a dynamic decision making process for

configuring a route, which takes into account connectivity constraints by ordering the decisions through the use of contexts. From a number of knowledge elicitation sessions at the Minsterley plant with production supervisors a small rule-base of process route configuration rules was developed for testing on some of the production activity routes in the model of the Minsterley plant. Figure 7.11. shows an example of two rules for the selection of a milk silo as a source for the activity **sep-milk-ev-skim**.

The cycle for applying configuration rules and progressively configuring a route is shown in figure 7.12. The choice of one particular element for a choice position will constrain the remaining choices due to the existence of the binary connectivity constraint, and each time an element is selected the consequences are propagated through the route to constrain the choices remaining for unfilled positions in the route. Thus it is not possible to select an infeasible element for placement in the route. The initial constraint checking of a route ensures that if at least one element exists as a choice in each route position it will always be possible to configure the route whatever decision order is adopted. This means that using a general set of rules a default mechanism for configuring a route can be implemented based on an arbitrary decision order for selecting plant items for route positions. At their simplest these rules could simply specify that the route is configured in order of the route positions and the first item in the list of choices for a route position should be selected each time. As long as the constraining effects of each plant item selected are propagated through the route each time a selection is made the default decision ordering will result in a feasible route configuration. This property of the approach means that feasible routes can still be configured by the BPS even when there are gaps in the configuration rule-base for specific routes. An incremental approach to model development can therefore be adopted initially using general route configuration rules and introducing more specific cases when deemed necessary. A full example of dynamic route configuration

rule evs3a:

IF GOAL configure activity sep-milk-ev-skim
AND SUB-GOAL choose-milk-silo
AND ROUTE is route1a
AND there are input CHOICES which contain grade 2 milk

THEN sort these CHOICES by age
AND select the oldest CHOICE
AND set new SUB-GOAL choose-separator

This is the more specific case for choosing an input for the separation activity. It differentiates between two grades of milk, choosing grade 2, (a lower quality grade ), for this activity. It also uses the oldest available milk to ensure that it is moved through the plant as quckly as possible. The select action triggers binary constraint checking to occur.

rule evs3b:

IF GOAL configure activity sep-milk-ev-skim
AND SUB-GOAL choose-milk-silo

THEN sort ROUTE CHOICES by age
AND select the oldest CHOICE
AND set new SUB-GOAL choose-separator

This is the default choice. If there is no low quality milk available for the activity, then choose the oldest milk available to ensure that it is moved through the plant as quickly as possible.

Figure 7.11. Example Configuration Rules for Sep-milk-ev-skim

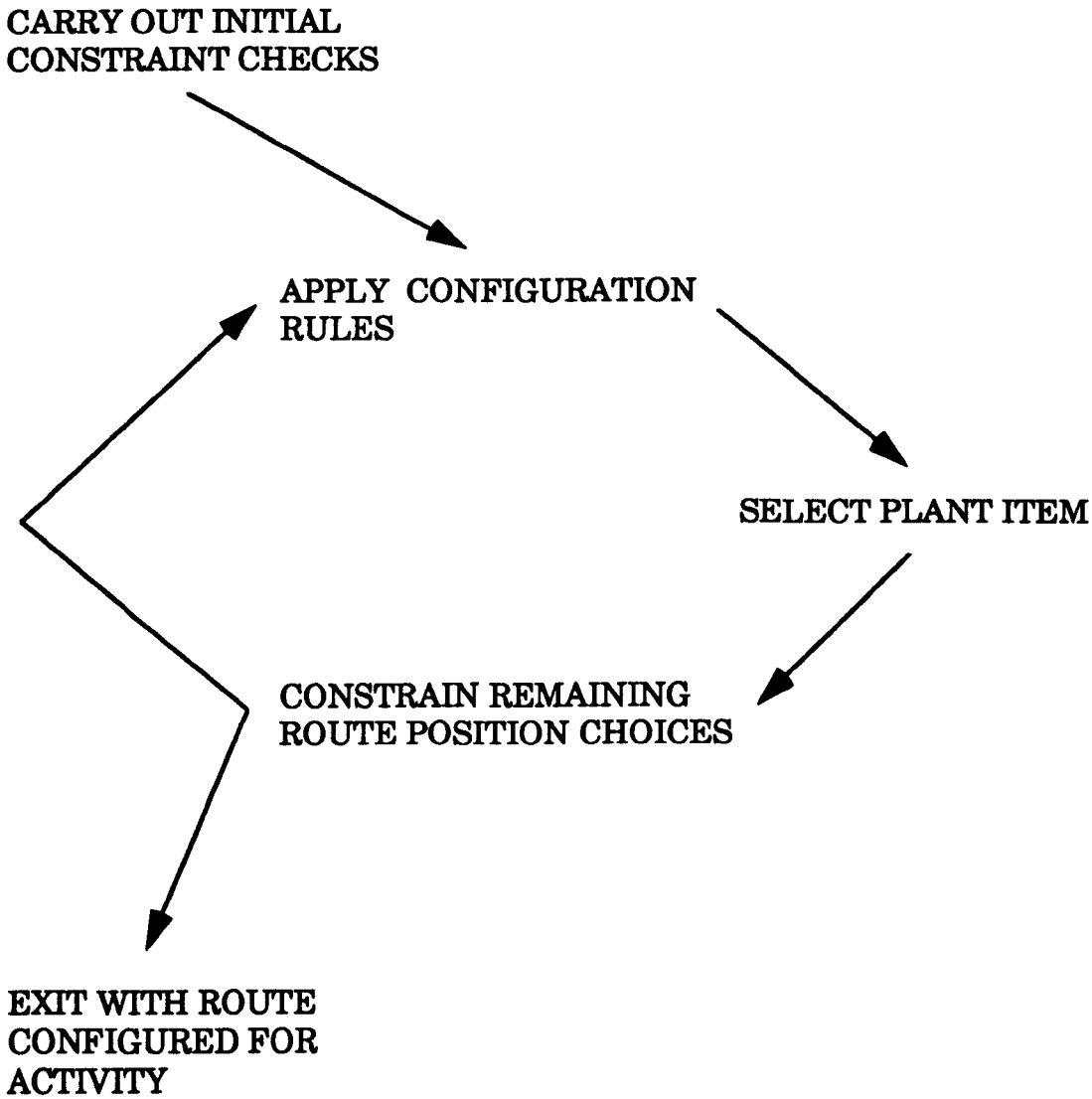



Figure 7.12. Basic Process Route or CIP Route Configuration Cycle

carried out as part of the BPSs activity scheduling cycle is given at the end of this chapter.

The knowledge elicitation exercise carried out to develop the configuration rule-base also indicated a number of plant item attributes which needed to be incorporated in the BPS to make it generic for modelling the Minsterley plant, and which could be useful in modelling other batch plants. These attributes include product age, the time an item was last used, grade of product, and actual product type either currently held in or being processed by the plant item or last processed by it. These attributes have been incorporated into the relevant generic plant item classes. In using the BPS to model other batch plants it is likely that the BPS capabilities would need to be expanded (or changed) to incorporate plant item attributes specific to that batch plant.

## 7.7.MAINTENANCE OF PLANT MODEL STATUS OVER TIME

The representation structures and mechanisms through which resources can be correctly allocated to production activities while taking into account the physical constraints based on the current system state have been described in the preceding sections, and are vital in order to be able to derive a feasible schedule. This configuration procedure must at any point be based on an up-to-date system status. This status is currently based on a dynamic simulation model, although it could be based on the status of a real plant if operating in real time; and the generic production activity configuration module is effectively unaware of how the plant status is derived.

A dynamic simulation model is an important part of the BPS, both for its use in the role of an off-line forward scheduling tool and also to aid in the development and testing of the rule-base for scheduling and configuring the plant. Therefore a generic simulation model for batch plant operation at the level of batch/ unit operations has been developed, which can be fully

integrated with the production activity configuration and scheduling module. It provides an input to the configuration module of current plant status over time when it is used in an off-line scheduling mode. The output from the simulation provides a schedule of the use of resources for production activities over time based on the allocation decisions made by the configuration module.

From a review of the operation of the Minsterley plant and relevant literature on batch process plants it was determined that a generic dynamic simulation model providing plant status input to the configuration module should have a number of specific features. In addition some features needed to be represented for determining a schedule by running the model. Some of these features are reflected in attributes associated with specific plant item classes. For example representing the processing rate of a semi-continuous process plant item as one of its attributes is vital to determine how much product flows through the system over a given time when that plant item is allocated to a route. Some features required the incorporation of procedures for correctly maintaining the dynamic status of the model when a number of process routes use the same finite capacity storage vessel as a source or sink. In a highly capacity constrained system it is very important that the material balance of product moving through different stages is correctly maintained. For a given production activity, there may be a change in the product composition, for example through separation into two or more components. This requires a representation of a product's Bill of Process (BOP), and procedures for determining how the volumes of product in the plant are changed through processes at different stages.

### 7.7.1.Physical Product Representation

The processing activities are concerned with the movement and processing of product through the plant so the physical representation of the product must be addressed. As described in Chapter 6, the typical representation of a batch

as a discrete entity which moves through the plant from unit to unit is not a particularly appropriate one to a batch process environment, where the units at different stages can vary considerably in size, and the intermediate products held at these stages may be supplying a number of different production processes at later stages in the plant. It is not always possible to dedicate raw material input to specific final product batches in many batch process environments, where the process goes from a few basic raw materials to a large number of final products. In an environment such as a dairy, where there is a considerable amount of semi-continuous processing, using discrete entities to represent a batch of product does not facilitate material balance calculations. In most batch plants, the product moving through the processing stages can be considered to be a fluid. Although in some cases it may actually be a powder, pellets, or even a product which is "set", such as yoghurt, in terms of semi-continuous processing and transfer at the level of detail with which we are concerned these products are all treated as a fluid and transferred or processed at a rate which is usually fixed. Thus the representation of the product within the physical plant model is defined as the volume of product within storage and reaction vessels, and what the product held in the vessels is. The representation of the product in relation to production recipes and production requirements should be considered separately and this is described in Chapter 8.

### 7.7.2.Co-ordination of Plant Item Activity in the Plant Model

In a simulation model which is going to provide input plant status to a configuration module and provide a time phased output of plant activity as a short term schedule it is important to correctly co-ordinate the activities of the plant items over time. Simulation is primarily concerned with the co-ordination of activities of interacting entities over time through an event scheduling mechanism. Events are scheduled on entities which determine when they will change state assuming other events do not occur in the

meantime which affect this. If an event occurs which does affect an already scheduled event, then this already scheduled event can be rescheduled as necessary. A change in state may alter the availability of a plant item for inclusion in a production activity when unary constraint checking takes place.

The scheduling of events corresponding to the times at which plant items will finish a production activity requires consideration of the specific class of production activity involved. In a production activity such as a batch reaction which only uses a single plant item it is simple enough to schedule an event corresponding to the time when a static phase of the reaction not involving product transfer will be completed. When a number of plant items have been linked together into a CIP, process or transfer route, the events relating to the items in that route must all be scheduled for exactly the same time, so that the route will run for an appropriate length of time and the shutdown of the route will be properly co-ordinated in time. In the case of a CIP route, this is still simple to implement. An event can be scheduled on the elements being cleaned corresponding to the length of time that the route should run for including all cleaning phases such as water rinses and detergent washes. However, in the case of a product transfer or semi-continuous processing activity a number of factors have to be taken into account:

1. The batchsize or amount of material required for the activity.
2. Any physical change which is involved for the product.
3. The capacity of the storage vessels involved as both sources and sinks for the activity.

The first two factors require a mechanism to maintain the material balance of the plant in relation to a specific batchsize for an activity, and the third factor requires that the scheduling of the events related to a processing activity takes the physical capacity constraints of the storage and reactor vessels into account. For example, storage vessels can be shared by two or more routes, as

an input, an output, or both, and only have a finite capacity, so the effects of a vessel reaching a capacity limit must be propagated throughout all affected routes.

Accounting for Production Activity Batchsize

Each process activity which is assigned resources in the model must have a **Batchsize** associated with it to represent how much product is to be processed through a route configured for the activity. Initially the simulation schedules events on the process elements in a route relating to a particular batchsize of product, without taking account of the effects of the capacity of finite storage and the actual available materials. The length of time for which the route should run is calculated by:

**Batchsize/Process-rate**

where **Process-rate** is rate of the processor assigned to the route and **Batchsize** is the amount of product to be processed. It is obviously important that the **Batchsize** and **Process-rate** are both expressed in the same units. Calculation of the batchsize for a route takes this into account as well as the output product requirement for a route which will have been derived from a production recipe and volume of the final product required. This is described in detail in Chapter 8.

It is obviously very important that the finite capacity of the system is taken into account. This infinite capacity approach to scheduling events on an individual route is taken initially because available storage capacity in a sink and available materials in a source cannot be easily determined from a simple static analysis of the state of the the plant. The actual available capacity and material available for a process changes over time, and may depend on the interactions of a number of process routes operating at different rates. Where

routes can be configured with shared storage units either acting as a sink for one route and a source for another, or being shared across a number of route positions, there are a number of factors which could affect the true available capacity for a route:

1. If there are one or more rates set as an input to a source of a route this will tend to increase the amount of available raw material.
2. If there are one or more other routes using the same source for their processing activities this will tend to decrease the amount of available raw material.
3. If there are one or more rates set as an output on a sink for a route this will tend to increase the amount of available storage capacity.
4. If there are one or more other routes using the same sink for their processing activities this will tend to decrease the amount of available storage capacity.

### 7.7.3. Finite Capacity of Vessels and Material Balance

The effects of finite capacity must be taken into account in the operation of the simulation, which depends on correctly calculating how the volumes of product in vessels change over time as process routes are running. The change in a vessels volume over time is represented by setting a net rate on the vessel as described by Fisher [52]. The effects of the finite capacity of a vessel are accounted for by scheduling capacity specific events on the vessels in the system based on its current net rate. The consequences of these events for routes which are running in the model are described in the following section on finite capacity storage.

The net rate on any vessel in the system is determined by calculating its input and output rates for each route it is being used in. By determining a separate input and output rate for each route that a source or sink element is in it

becomes easy to alter the overall input and output rate on that element when there are any changes to the routes that the element is in. An input or output rate is not necessarily the same as the process rate for the route, because the process may result in a physical change to the product. Where a production activity results in a physical change to the product this must be correctly accounted for in the model through a material balance mechanism. The material balance of the system can be maintained quite simply if the amount of product within the plant at any stage is considered to be a function of the rates of processing and the time which the model has been running for, adjusted by input factors and yields on the routes through which the product is being processed. The **input factor** and **output yield** figures used for the processing routes which carry out a particular activity are based on a simple Bill of Process (BOP) for that activity. For example, the figures required for the material balance calculation for the separation of milk into skimmed milk and cream are shown in figure 7.13, which shows the proportional yields of skimmed milk and cream from the separation of milk. The material balance procedures are used to maintain the correct volumes of product in the plant items which have capacity to hold it, such as storage vessels and bulk transport vehicles. For each process route, the **output yield** and **input factor** are used to convert the **process rate** of the process plant item, for example a separator or pasteuriser into an **Output rate** on the source element for the route, and a **Yield rate** on the sink element for the route. The **process rate** of a plant item in a transfer or semi-continuous route is converted into an **Output rate** on the source element as follows:

**Output rate = process rate*input factor**

and the **Input rate** on a sink element for a route is determined as follows:

**Input Rate = process rate*output yield.**

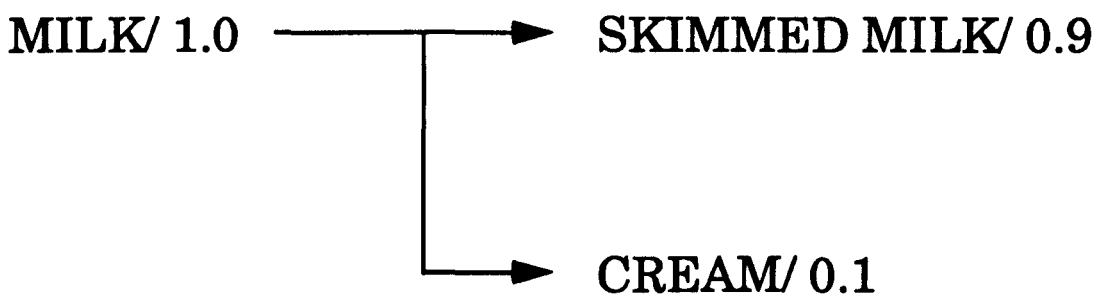



**Figure 7.13. A Bill of Process for separation of milk**

In the case of a product transfer in which no structural change occurs to the product, both the **input factor** and **output yield** on the route can simply be set to 1, so the **Output rate** on the source, and **Input rate** on the sink are the same as the processor element **process rate**.

This enables the material balance of the system to be maintained in cases where the structure of a product is changed by a process as follows:

1. An intermediate product may be split into two or more other intermediate or final products, in which case there will be two or more routes for the output products each with an appropriate yield for the process.
2. Two or more intermediate products may be combined into a third product, in which case there will be two or more routes for the input product each with an appropriate input factor based on the proportion of that intermediate used in the process.
3. A single product may be physically changed by a process so that its volume is altered. In this case the yield will reflect the change in volume seen at the sink element of the route.

The overall event scheduling procedure is carried out as follows. Once a route has been configured, and an appropriate batchsize has been defined for it, events relating to the time the route should run for are scheduled on all elements in the route by following the links up and down a chain of linked sub-routes. At the source and sink positions of the chain, the appropriate source output rate, and sink input rate are set up on those entities based on the input factor and output yield. These route specific rates are used to calculate the amount of product used and produced by the route. They are also used to adjust the net rate on the source and sink vessels which may be in use in other routes as well as this one so that the finite capacity of the system can be properly accounted for.

The material balance procedures enable the physical representation of product volumes in the system to be correctly maintained, (including known material losses if desired), but they must also be related to the production requirements of the plant. At the end of the scheduling period, the output of the model should show how much product moved through the plant and what production activities the product within the plant at different stages was used for. This part of product representation is described in Chapter 8.

### 7.7.4.The Effects of Finite Capacity Storage on Process Routes

Although a route can be set up and scheduled to run for a period of time to process a given batchsize of product with the correct material balance as described above, this does not take into account the finite capacity storage in a batch process plant. Such capacity limits affect how much raw material is actually available for a route, and how much storage space is available for the products of a process. The actual availability of raw material to enable a specific route configuration to run continuously for the period required to process the full batchsize depends on whether the source(s) of the route become empty or not during the period for which the route has been scheduled to run. Likewise, the availability of storage capacity in the sink(s) of the route for an uninterrupted run depends on whether one or more of the sink(s) becomes full or not during the process. This does not mean that there is not enough raw material or storage capacity available overall to be able to process the batch, but if a finite capacity limit is hit on a vessel then any affected routes must be reconfigured if possible with other available storage vessels in order to continue processing. In some situations, such as liquid milk processing, a number of routes through a number of processing stages can be set up using shared storage vessels. The effects of one or more vessels at a one or more stages becoming full or empty could therefore affect a number of routes; events scheduled on them on the basis of a batchsize requirement may have to be rescheduled to account for the finite capacity constraints.

In order to account for these requirements in the generic model where the length of the routes involved in a process/ product transfer, and the numbers of routes using shared storage is variable, an event rescheduling propagation mechanism through the affected chains of sub-routes is used. Whenever a vessel hits a maximum or minimum capacity level, the effects are propagated to all affected route items through the links in the chain of sub-routes. Two cases have to be considered when determining which routes are affected by one of these events. If a a vessel hits maximum contents, then all routes in the model which are currently using this vessel as an output will have their route specific scheduled events rescheduled to the current system time to account for this entity specific event. In addition, route branches which are not directly using this storage vessel, but are using the same semi-continuous processing plant item because they are part of the same production activity, must also be shutdown as a result of this event. For example, in the activity **sep-milk-ev-skim** there are two output product route branches as shown in Figure 7.10a, which each have a different sink. If either one of these sinks was to hit maximum capacity when the production activity was being carried out the effects would have to propagated through all route branches. If a vessel hits minimum contents, then the effects are the same, but act on all routes which directly use the vessel as an input, and again indirectly on routes using the same procesing element. The procedures are carried out as follows:

**procedure** Vessel Maximum Contents

**for** a vessel which has hit maximum contents **do**

set the vessel input rate to zero, and appropriately adjust the vessel net rate

**for** each process route which is currently running with this vessel as a sink **do**

reschedule to the current clock time the end of this processing activity on all plant items in the route including those in any branches

**end** Vessel Maximum Contents **procedure**

**procedure** Vessel Minimum Contents

**for** a vessel which has hit minimum contents **do**

set the vessel output rate to zero, and appropriately adjust the vessel net rate

**for** each process route which is currently running with this vessel as a source **do**

reschedule to the current clock time the end of this processing activity on all plant items in the route including those in any branches

**end** Vessels Minimum Contents **procedure**

Threshold Levels and Falling and Rising "Triggers"

As described in the section on unary constraint checking, the status of falling and rising level triggers are checked to prevent routes being set up which could cause the system to continually "cycle" around a finite capacity limit. These triggers also have an event scheduled on them on the basis of the net rate on the vessel as described by Fisher [52]. When the trigger is hit they become available for use in process routes again.

## 7.8.AN EXAMPLE OF THE DYNAMIC CONFIGURATION OF PROCESS ROUTES

This chapter has described the generic modelling features and procedures developed for resource assignment to production activities while taking into account the true constraints inherent in a batch process plant. The ability of the BPS to correctly configure production activities using these procedures and taking into account preferential considerations has been tested on a model developed of the Minsterley plant skimmed milk routing network. A process flow diagram of this network is shown in Figure7.1a. The AND/ OR configuration of the individual plant items in the network which are referred to in this example are shown in Figure 7.4.a - d.

For the purposes of this example a number of batches of skimmed milk needed to meet demand for a 24 hour period are 'active' in the system including a batch of skimmed milk for the evaporator department with a code of **ev01**, and a batch of skimmed milk for the cottage cheese department **cc01**. To produce these batches of skimmed milk requires two separation activities to be carried out: **sep-milk-ev-skim** for batch **ev01**, and **sep-milk-cc-skim** for batch cc01. The first activity/ batch presented for configuration is **sep-milk-ev-skim/ ev01**, and a search for a potential route for the skimmed milk component of the activity results in the choices shown in Figure 7.14. The configuration of this route under dynamic connectivity constraints and other factors affecting resource availability will be carried out according to the configuration cycle illustrated in Figure 7.12. and as described in the following sections.

### 7.8.1.Current State of the Plant

The activity state of a plant item can represent a constraint on its availability. Other plant item attributes are taken into account as part of the preferential consideration in assigning a plant item to a route position. At this point in the

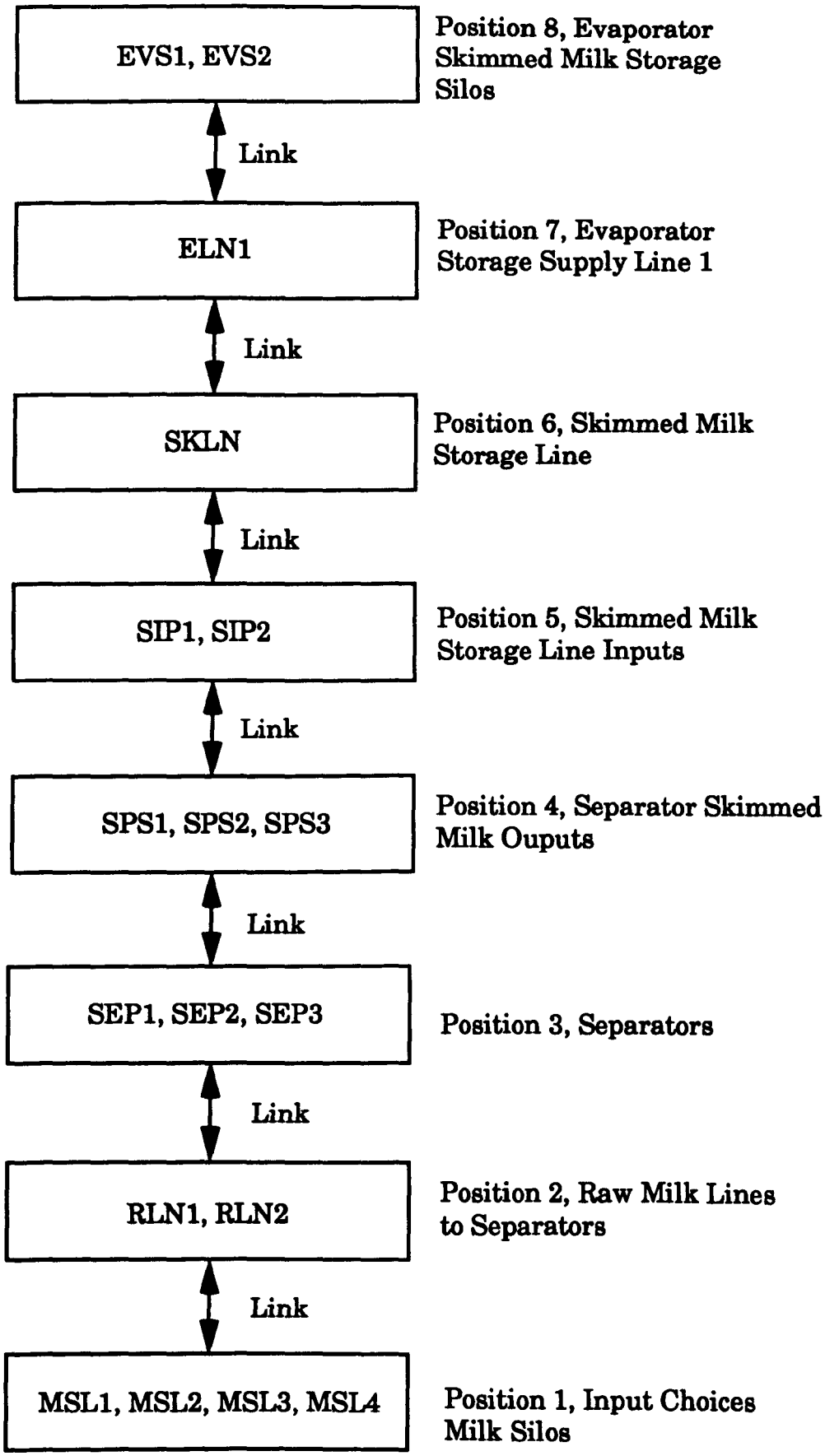



Figure 7.14. Initial Item Choices for Process Activity sep-milk-ev-skim Route 1

configuration procedure some key attribute values of the plant items which are potential choices for the route are as follows:

1.Milk Silo1 (MSL1) contains the oldest milk which is also 'grade 2'.
2.Milk Silo2 (MSL2) has an activity state of waiting empty and thus has a 'contents status' of 'fill only'.
3.Milk Silos 3 and 4 (MSL3, MSL4) both contain 'grade 1' milk, although the milk in MSL3 is older than that in MSL4.
4.At present none of the plant items are in a route which is already running so they all have an activity state of waiting either clean or dirty

Before any preferential considerations are taken into account, the route is made consistent with respect to unary and binary constraints using the procedure described in section 7.4. During this initial constraint check the vessel MSL2 is knocked out as an input choice for the route during the unary constraint check because it is empty and is therefore 'fill only'. On entering the binary constraint check part of the procedure each route position is checked to see if the **CAN-CONNECT-TO** relation holds for the choices for positions in the route, with the outcome in this case that none of the possible choices are lost. Thus the BPS enters the rule based phase of the configuration process with the route choices as in Figure 7.15.

### 7.8.2.Preferential Element Selection

The rules concerned with configuring this activity were developed from a number of knowledge elicitation sessions held with production personnel at the Minsterley factory. The documentation associated with these sessions is given in Appendix B. Part of the process of developing the rules for the configuration of this activity was concerned with determining a suitable decision ordering to give the best route configuration possible based on preferential considerations while taking into account the connectivity

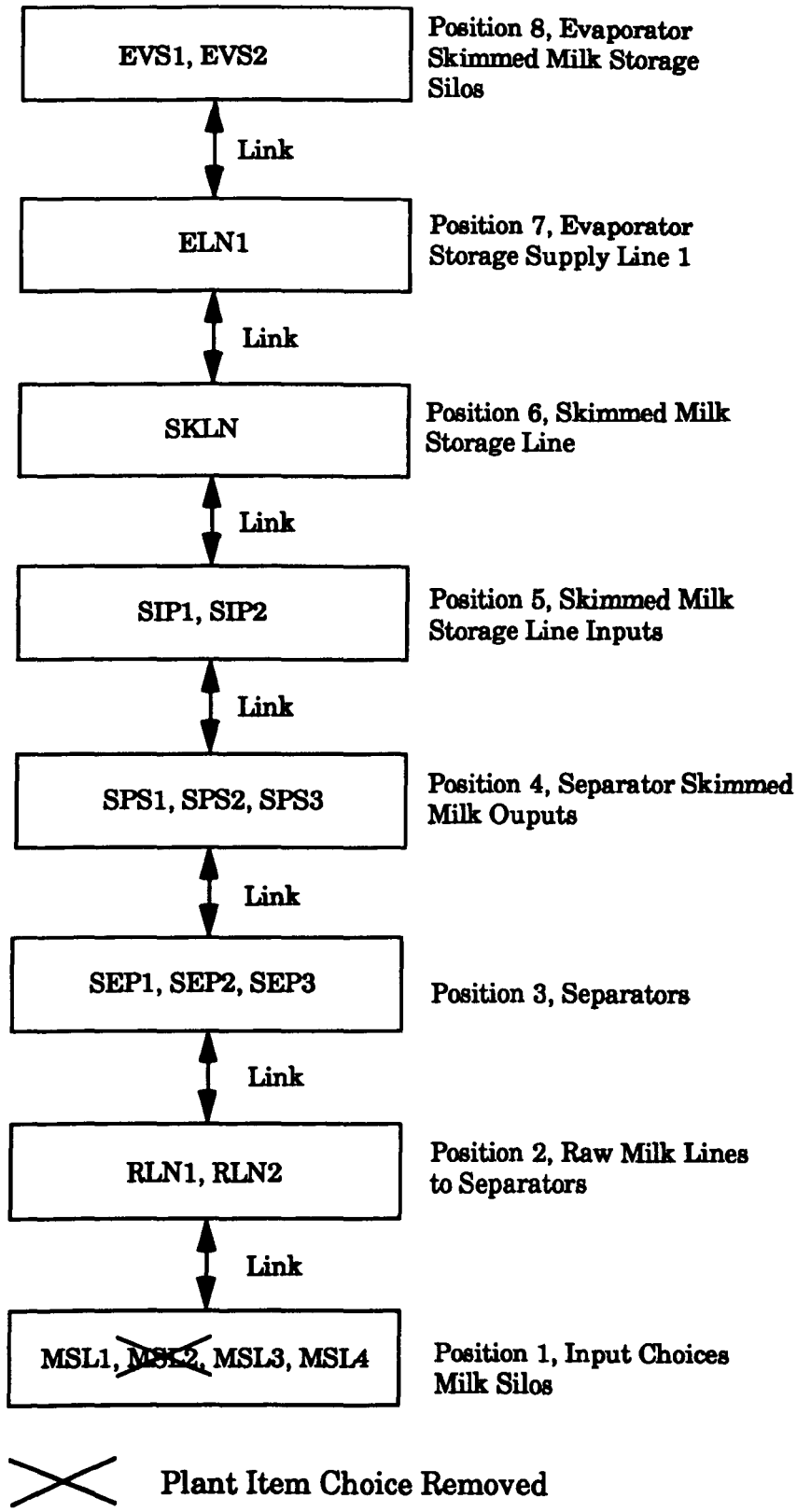



Figure 7.15. Choices Remaining After Initial Constraint Check

constraints in the plant. The effect of connectivity constraints varies from activity to activity. In the case of this activity the choice of the route's source has the most constraining effect on the rest of the choices for the route choices so the decision order developed for this activity was as follows:

1. Choose a source for the route, because it is important to move incoming raw material through the plant as quickly as possible and to take the product grade into account. The choice must therefore be as unconstrained as possible.
2. Choose a separator for the route taking into account any other separation activities which also require configuring.
3. Choose a skim line input for the route. The configuration of the two skim line inputs is identical and it does not really matter which one is used. However, one must be chosen because the route position containing them may not be so constrained due to the state of the plant as to ensure a default choice.
4. Choose a sink for the route. This choice is made last because it is not really constrained by the other choices so it can be left until the other choices have been made.

Decision One: Choose Route Source

The first active goal is to choose a source for the route, and the rule that succeeds is to choose a milk silo which contains the oldest milk in order move the milk through the plant fast enough to prevent quality problems and to clear the milk reception area for incoming milk. On this basis the input choice MSL1 is selected for the route. The selection of the element automatically triggers constraint propagation to occur to determine whether any of the other choices no longer apply because they can no longer satisfy the **CAN-CONNECT-TO** relation. The procedure checks through the route positions checking the relation. When the second route position is checked against the

assignment of MSL1 made for position 1 the element choices currently available are Raw Milk Line 1 (RLN1) and Raw Milk Line 2 (RLN2). Both RLN1 and RLN2 can take MSL1 as an input via an OR connection and MSL1 also has a potential OR output connection to RLN1 or RLN2. The **CAN-CONNECT-TO** case which applies for RLN1 and RLN2 is therefore:

IF Entity2 is on the inputs of Entity1 as an OR connection
AND Entity2 has OR outputs
AND Entity1 has no other current input
AND Entity2 has no other current output
THEN the relation holds

where Entity1 is either RLN1 or RLN2 and Entity2 is MSL1.

Since neither RLN1 or RLN2 already have a current input and MSL1 has no other current output they both remain as choices for the route with respect to this check. After checking the other route positions all the choices still remain, so the configuration goal next made active is to choose a separator based on the decision order represented in the rules.

Decision Two: Choose a Separator

In this case the separator chosen is Separator 1 (SEP1) because this will leave separator 3 available for the cottage cheese skimmed milk separation activity. The selection of this plant item again triggers constraint propagation to occur. When position 2 containing RLN1 and RLN2 is tested against position 1 containing MSL1 there is no change because the situation is the same as described above. Position 2 is then tested against position 3 which now contains only SEP1. When the potential outputs of RLN1 are checked against the potential inputs of SEP1, the case which applies is:

IF Entity2 is on the outputs of Entity1 as an AND connection
AND Entity2 has AND inputs
THEN the relation holds

SEP1 is on the output configuration of RLN1 as a potential AND connection, and RLN1 is represented as a single potential AND input on the configuration of SEP1. RLN1 passes this test with respect to SEP1 so it is kept as potential choice for the route.

When RLN2 is checked against SEP1, none of the **CAN-CONNECT-TO** cases apply because SEP1 is not on the output configuration of RLN2, only separator 3 (SEP3) is. However, because SEP3 was removed as an available choice for the route when SEP1 was selected RLN2 no longer has this as a feasible potential output choice, so it fails the test for the **CAN-CONNECT-TO** relation overall against position 3 and is therefore removed as a potential choice. The fact that a change has been introduced to the route by removing RLN2 is "posted" by the constraint checking procedure and the remaining choices in the route are checked. The removal of SEP2 and SEP3 through the selection of SEP1 causes their skimmed milk output lines, SPS2 and SPS3, to be lost through constraint propagation because they no longer have feasible inputs. Because a change has been posted the **CAN-CONNECT-TO** relation must be tested through the route again from the beginning to ensure consistency, and on the second pass all element choices pass the constraint check so element selection succeeds. The status of the choices for route positions is now as shown in Figure 7.16. This shows that some route positions can become so constrained that there is no need to apply situation specific rules to select a plant item for them. There is no need for a rule for the selection of the separator input line as RLN1, as this is taken care of by constraint propagation which will always result in a single remaining choice for this position under the current decision order imposed. There is also no

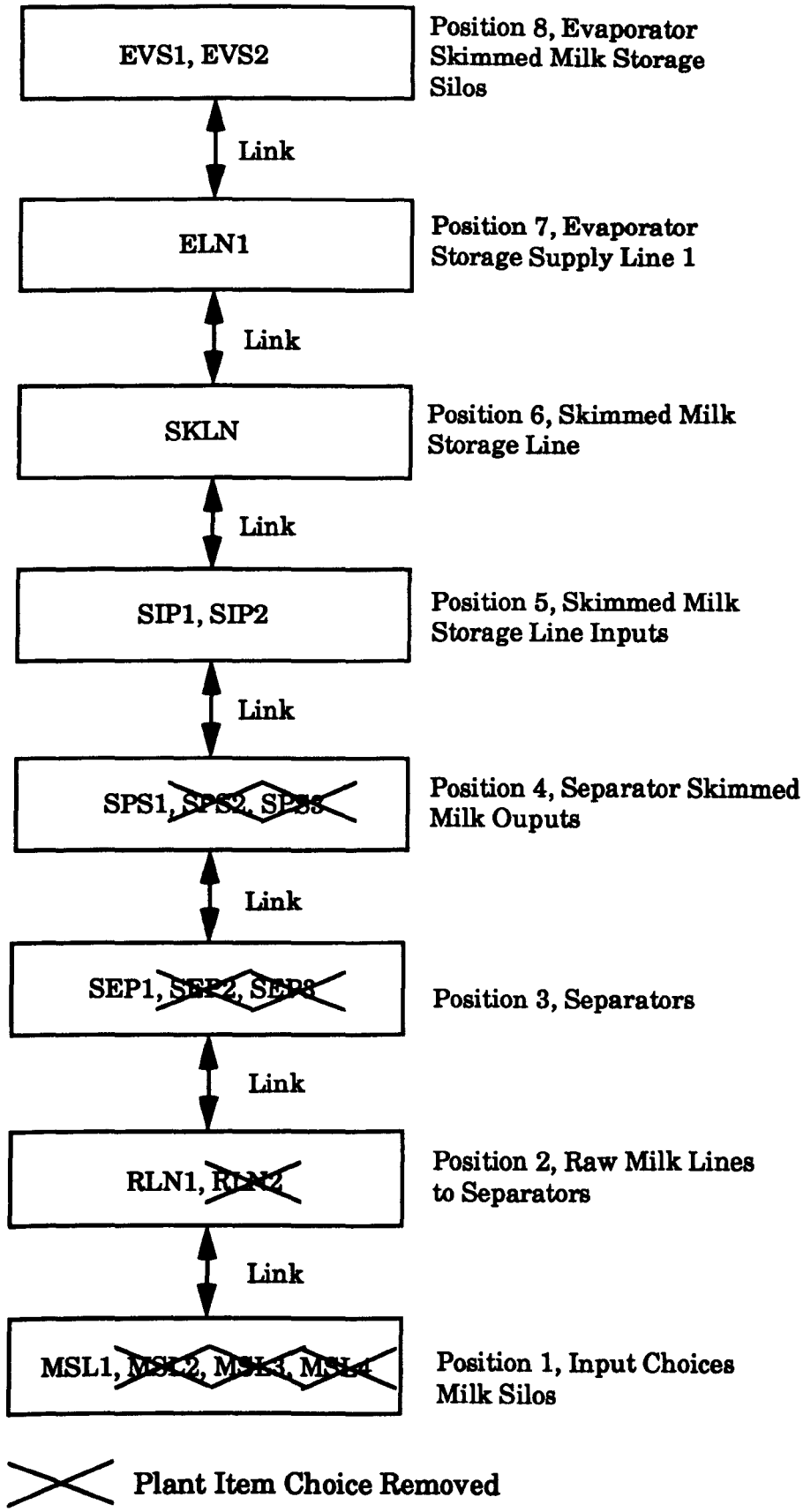



Figure 7.16. Partially Configured Route; Milk Silo 1 and Separator 1 Chosen

need for a rule for the choice of the Skim Line (SKLN) or the Evaporator Silo Input Line (ELN1) as these are both single element choices.

Decision Three: Choose a Skim Line Input

The Skim line input (SIP1 or SIP2) which is to be used cannot be guaranteed to be reduced to a single choice through constraint propagation alone: which could only be ensured if another route was already running into the skim line using one of these inputs. Therefore, a rule which selects the plant item on the arbitrary basis of "first plant item in the choice list" is used in this case which results in SIP1 being selected for the route. The constraint propagation procedure is carried out again, and the remaining choice which requires preferential consideration is made active; that is the choice of an evaporator silo.

Decision Four: Choose the Evaporator Silo

In the case of the evaporator silos Evaporator Silo 2 (EVS2) is chosen on the basis of 'silo last filled', resulting in a final route configuration as in Figure 7.17.

Having configured this route for separating milk to produce skimmed milk for the evaporator department the second activity/batch is considered; **sep-milk-cc-skim/ cc01**. In comparing the route choices for this activity with **sep-milk-ev-skim** it can be seen that there is a potential conflict for resources which is taken care of by the procedures for determining the availability of plant items. After the initial constraint checking and choice of an input silo, it can be seen from Figure 7.18. that the potential choices remaining for other route positions have been considerably reduced due to the connections in the plant that now exist for the **sep-milk-ev-skim** route which has just been configured and the choices so far made for this route. This also shows that it is important

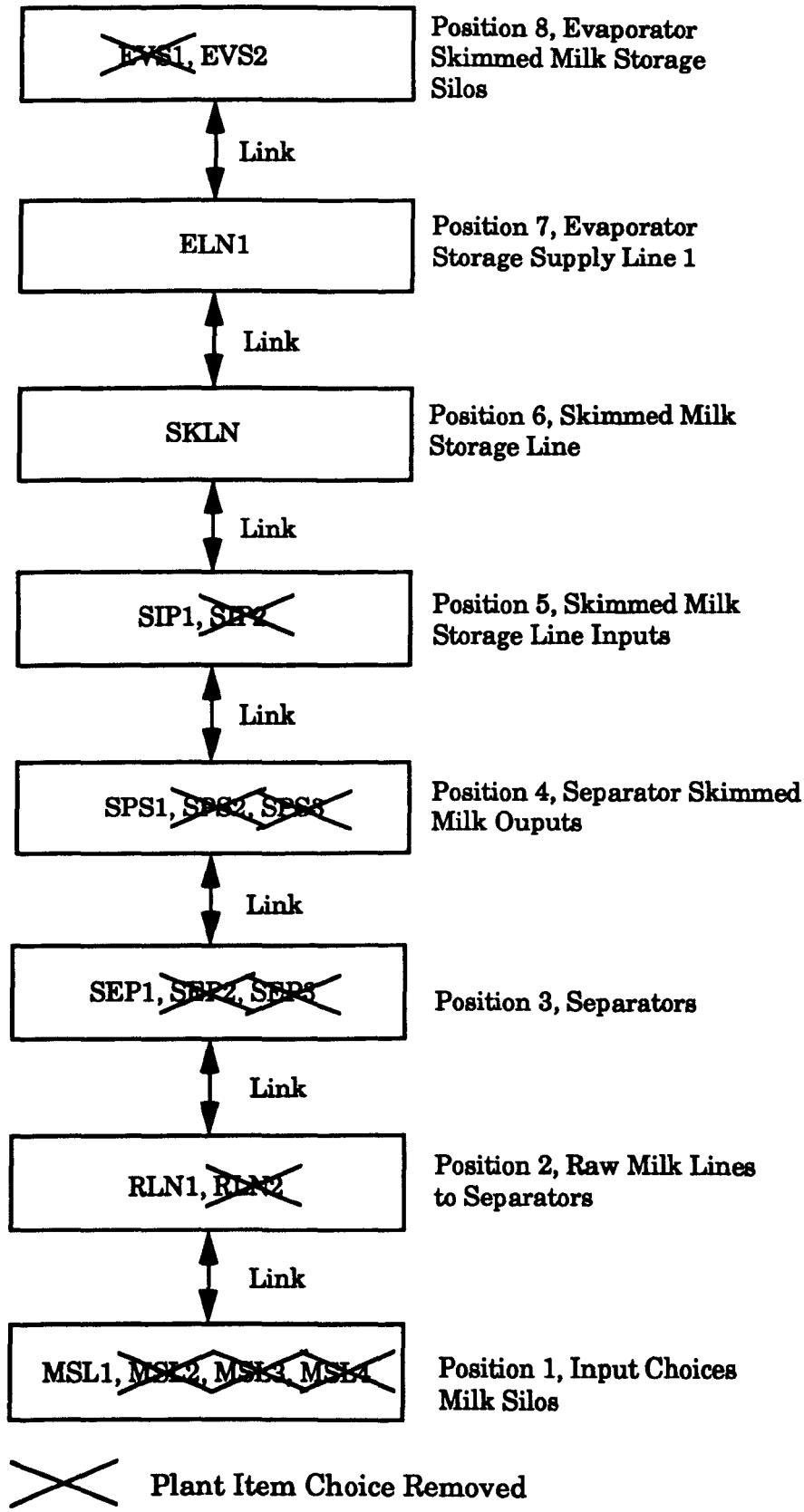



Figure 7.17. Final Configuration of Route 1a for Sep-milk-ev-skim

to be able to define a good decision order because this will have a considerable effect on the choices that remain in each position as the configuration proceeds. For example, the first decision for **sep-milk-cc-skim** was 'choose an input silo', and the silo was chosen on the basis of milk grade 1 being preferred for cottage cheese production leading to the remaining choices as in Figure 7.18. after constraint propagation. However if a different decision order was used for this route so that the first decision taken was 'choose a separator', and SEP2 was chosen on the basis of a particular cream output requirement then constraint propagation would take out all silos except MSL1 because of the requirement for SEP2 to connect to RLN1 which already has a current input connection from MSL1. Thus the only grade of milk available to the route would be 2 which is not ideal for cottage cheese production.

Therefore the development of the rulebase for plant configuration requires a careful consideration of the consequences of each choice on the remaining choices for a route in the light of the overall operation of the plant. However, because of the network consistency and contraint propagation procedure, whatever choice is made for a route position will always leave at least one valid connection left in immediate route positions.

## 7.9.DISCUSSION

This chapter has described a specification of necessary procedures for a generic Batch Process Scheduler (BPS) for the configuration of production activities and simulation of batch process operations at the plant unit/ batch level. The knowledge representation scheme put forward solves the problem of representing the connectivity constraints in a plant network at a suitable level of detail. The routing representation scheme and the network consistency and constraint propagation procedure enables the configuration of production activities to be carried out subject to the current status of the plant. This basic procedure ensures that if at least one plant item exists as a choice for each

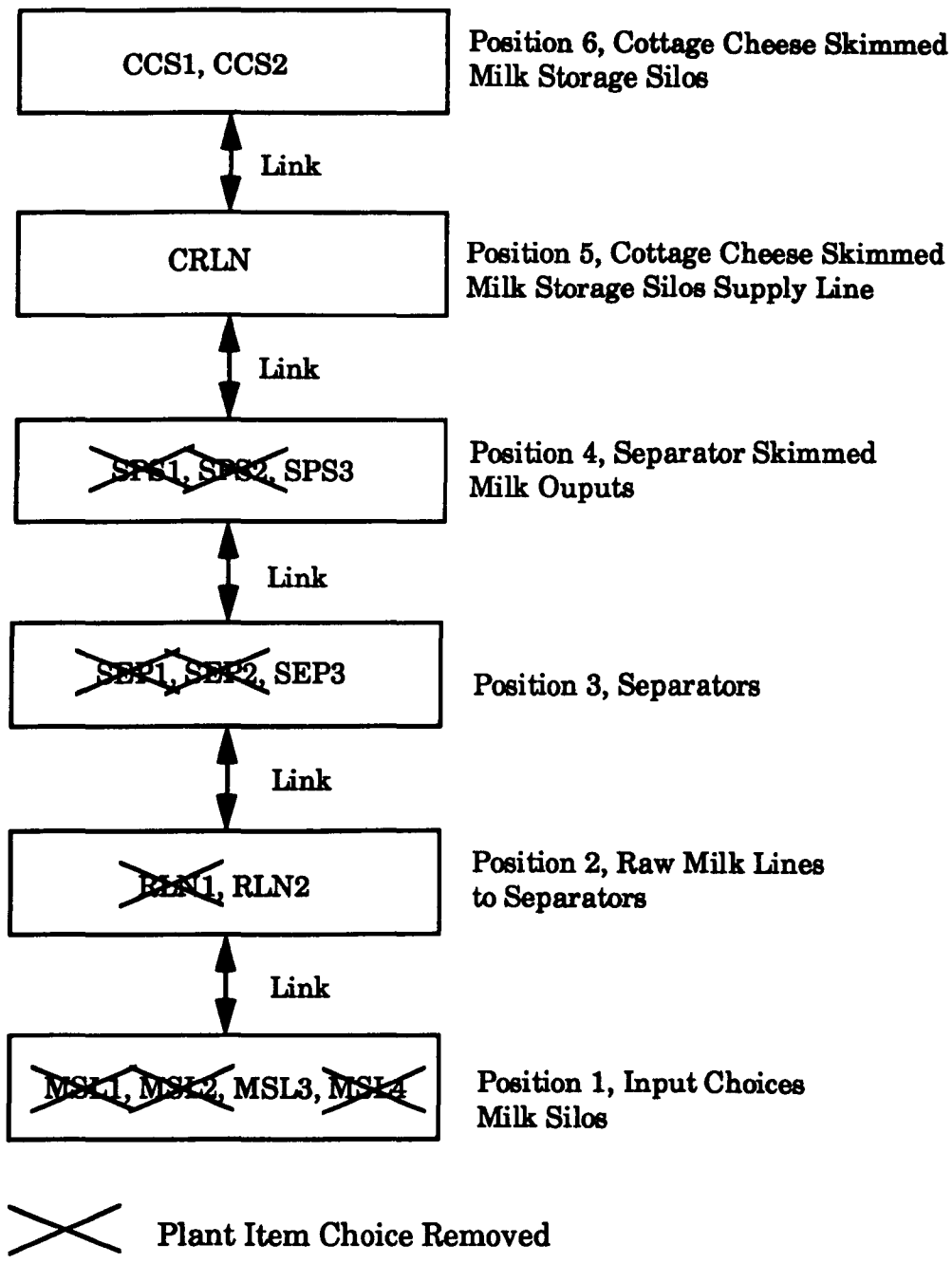



Figure 7.18. Partial Configuration of Sep-milk-cc-skim Route After Choice of Milk Silo 3, and Resulting Constraint Propagation

position in a route when it is initially checked before actual configuration, it will be possible to configure the route using a procedure of selecting an element and propagating the constraints for each position in the route. By incorporating this procedure with a rule-based system for the representation of preferential considerations in allocating resources to activities and a decision order to guide the configuration process the BPS can be used to derive the best configuration for an activity under a given set of circumstances.

A purely rule-based approach could have been used to represent the connectivity constraints and their effects on routing within a batch plant network. However, this type of approach is subject to a number of potential problems such as gaps in the rule-base so that not all circumstances are covered, poor performance due to a large rule-base, and difficulties with updating and maintenance of the rules. The approach which has been put forward overcomes these potential problems. The knowledge representation scheme and network consistency procedure ensures that the BPS can infer the constraints on a route configuration at any point in the model's execution rather than rely on their representation through rules. The BPS is robust in that it can cope with gaps in the rule-base and still produce a feasible configuration for a route through the use of general default rules, and the consistency and propagation procedure. This also allows incremental system development rather than having to specify a large rule-base at the outset. The procedure also ensures that the size of the rule-base required is considerably reduced making its maintenance and updating easier.

One of the reported problems with a network representation of a problem and the use of constraint propagation techniques is that they can require a system to do a lot of work when changes occur in the range of a node value. However, in the procedure suggested, because the problem can be decomposed into small network representations of routes bounded by input and output storage vessels the amount of work that the procedure has to do at any one time is

kept to a reasonable level for acceptable response times. This decomposition approach is a natural one to adopt, because it analogous to the real situation. In a real plant storage vessels are used to decouple semi-continuous processes and allow them to operate with a degree of independence from each other.

The activity configuration module could function as an on-line real-time plant configurer, or as part of an off-line forward scheduling system in conjunction with a batch plant simulation model. The simulation model developed contains some specific features to enable it to be used in this way. It co-ordinates events on chains of plant items making up process routes. It allows process routes to share storage so that generic intermediate products can be used to supply a number of different processes at the same time. It accounts for process related changes to product so that the material balance of the plant can be correctly maintained.

BRITISH LIBRARY HSIRB

IMAGING SERVICES NORTH
Boston Spa, Wetherby
West Yorkshire, LS23 7BQ
www.bl.uk

# PAGE MISSING IN ORIGINAL

# CHAPTER 8 ACTIVITY SCHEDULING USING THE CONFIGURATION MODEL

## 8.0.INTRODUCTION

The main aim of this research has been to develop modelling constructs for scheduling of batch process plants so that the allocation of resources to activities and operations over time will properly take into account the physical constraints to such allocation that exist, as well as preferential considerations. The preceding chapter has described the features required for the dynamic configuration of the model at the level of batch/ unit activity. These features ensure that the assignment of resources to any production activities scheduled using this model will be feasible and could be implemented on a real plant. Unless a schedule developed by any approach is actually feasible, of course it cannot be implemented. Having developed a model which will produce feasible assignments of resources to production activities over time, it can form the basis of a scheduling system which can be progressively developed to produce more acceptable schedules which meet scheduling goals. The ability to develop good feasible schedules requires the ability to represent and reason about the relations between the production activities required to produce a product. The system has therefore been developed with a basic framework for this, which has enabled the dynamic plant configuration and time based modelling to be tested with simple production requirements, and forms the basis of a more comprehensive system. This framework is based on a knowledge representation scheme for defining production recipes and representing the status of the product in the plant, and uses procedures for reasoning about the temporal precedence relations between activities coupled with the use of production rules for representing plant specific control information and heuristics to guide the decision making process.

## 8.1.REQUIREMENTS FOR KNOWLEDGE REPRESENTATION FOR ACTIVITY SCHEDULING

In order to develop a schedule to meet the production requirements of a plant requires knowledge of what activities need to be carried out to manufacture a particular product, what the constraints on them being carried out are, how to determine which activities can be carried out next, and situation specific knowledge about which of these activities should be carried out. The constraints on activities can be classified as scheduling restrictions and scheduling preferences as described by Fox [67]. Scheduling restrictions include causal restrictions such as precedence and resource requirements, and physical restrictions such as plant unit capability and connectivity.

Preferential constraints may be based on the production goals for the system such as the requirement to meet due dates or meet product quality specifications in terms of **No Wait (NW)** operations. These constraints can be broken in a feasible schedule but must be represented and taken into account in conjunction with the scheduling restrictions when determining which activities to attempt to assign resources to. The degree to which they are met or relaxed/ broken determines the acceptability of the schedule. A schedule will be feasible if scheduling restrictions are not broken, and acceptable if preferential constraints are met to some degree determined by the users of the system. For example, a schedule in which product has to wait for an hour after a **No Wait** operation can still be feasible in terms of the assignment of resources to activities, but it will probably be totally unacceptable to the users of the system.

The knowledge about the relationships between activities which must be carried out to manufacture a particular product is normally represented in the form of a production recipe. In addition, in order to determine which products

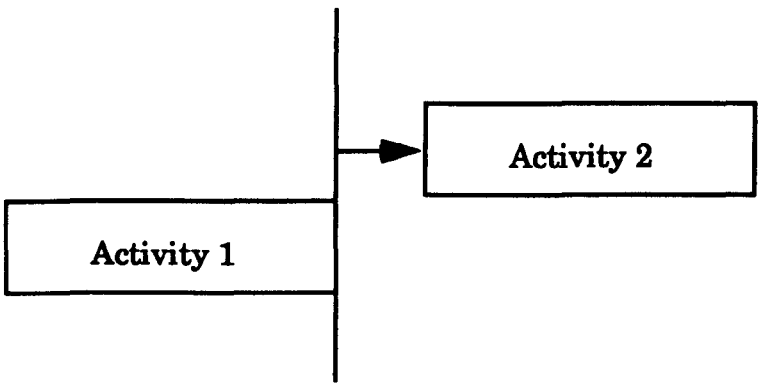
to progress through the plant at any particular stage of schedule development requires that their production status is represented.

## 8.2.PRODUCTION RECIPES

A recipe shows the various stages of the process for a product as one or more activities to be carried out and gives details of the requirements for processing when carrying out these activities. The recipe can be used to indicate scheduling restrictions and preferential considerations. The key scheduling restriction information required about the production activities at the batch scheduling and management level are the precedence relations between activities and any requirements for activities to be carried out together. Preferential information about activities can include alternative activities which can be used to carry out the same stage of a process. The level of process detail in the information represented in the recipe must be sufficient for scheduling batch/ unit operations. At this level we are concerned primarily with the volumes of product moving through the plant and the material balance of the system. Therefore, information such as material requirements, process reaction timings and material balance information must be represented to enable decisions to be made about the scheduling of activities.

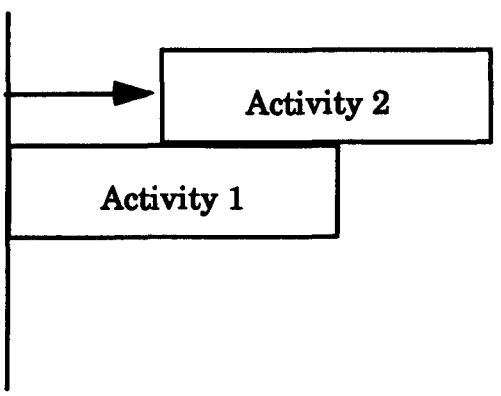
Precedence relations between activities can be represented using a temporal interval notation as described by Allen [82] and used in heuristic search based approaches such as ISIS [67]. The necessity for representing these relations in the reactive approach developed here, as opposed to a search based approach, is to simply determine whether an an activity is able to start or not rather than to infer relations about activities along the whole scheduling horizon time line. Therefore, only a small sub-set of the relations described by Allen need to be represented. These relations are **BEFORE** and **OVERLAPS** as illustrated in Figure 8.1. **BEFORE** is used to indicate that an activity must be fully completed before a related activity at the next stage can start.

Activity 2 can-start at any time after Activity 1 finishes




Activity 1 BEFORE Activity 2

Activity 2 can-start once Activity 1 is in-progress



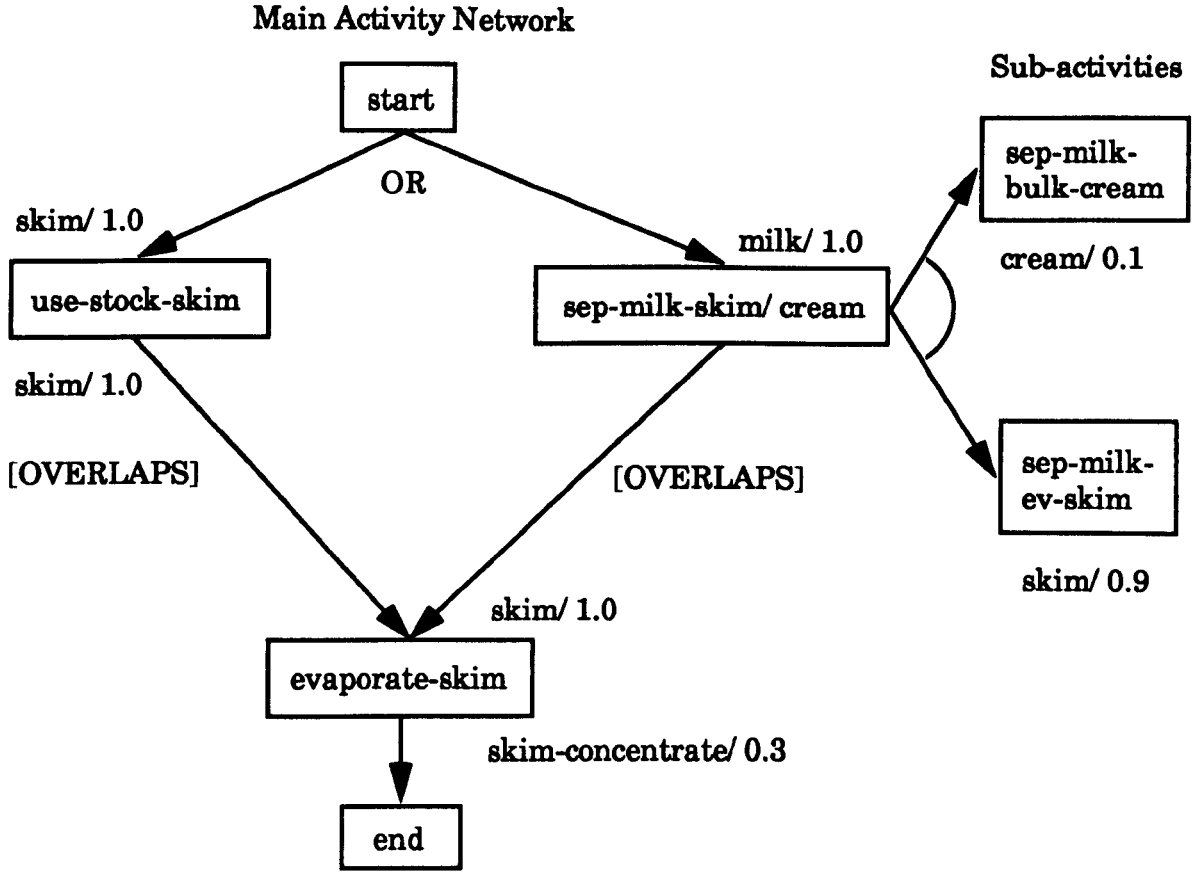

Activity 1 OVERLAPS Activity 2

Figure 8.1. Temporal Relations Between Activities

**OVERLAPS** is used to indicate that for two or more activities in two different stages of the process, the activities at the second stage are able to start before the activities at the preceding stage have finished. However, this information does not indicate how much of the activities in the first stage should be completed before the second stage activities can commence. This is an important consideration and will be discussed later. The other scheduling restriction that some activities must be carried out together, and the preferential information about alternative activities for carrying out a process stage are represented using the AND/OR notation as described by researchers such as Fox [67].

The representation of process and CIP routing for carrying out semi-continuous processing, product transfer and cleaning activities, and the complexity in routing which is introduced by separation and blending activities, has already been described in Chapter 7. Some activities relating to a specific final product may be decomposed into a composite structure of sub-activities in order to represent "split" routes based on the products involved. For example, a milk separation activity can be decomposed into a composite of two sub-activities relating to the cream component and skimmed milk component of the activity. These two sub-activities can then each have alternative potential routes associated with them, which can be combined to form the overall route associated with the activity. Decomposing an activity on the basis of component products enables the status of product batches with respect to the activities associated with their production to be maintained as will be described in the section 8.3. on product status representation.

A recipe for the production of batches of skimmed milk concentrate at the Minsterley plant is shown in Figure 8.2 using the notations described above. The activity **sep-milk-evaporators** is decomposed into two sub-activities **sep-milk-ev-skim** and **sep-milk-bulk-cream**. Each of these sub-activities has routing associated with it relating to the skimmed milk or cream




The recipe representation uses an AND/ OR representation for the activities and sub-activities involved in the production of a product. It also shows the temporal relations for determining whether an activity can start or not, and indicates the product Bill of Process for each activity.

It does not show specific routes for activities. These are associated with the activity representations.

It does not show product batch-sizes for activities. Appropriate batch-sizes for each activity to meet specific final product requirements are calculated using the Bill of Process information.

Figure 8.2. Example Recipe for Activities involved in Skimmed milk Concentrate Production

components, which can be combined to form the complex route associated with the activity.

## 8.3.PRODUCT REPRESENTATION

As discussed before, production may progress from large volume batches of general intermediate products to smaller volume specific final product batches derived from these general intermediate products. For example, as illustrated in Figure 8.3., in the manufacture of cottage cheese at the Minsterley plant a large number of small specific final product batches are derived from a single source of skimmed milk as the generic raw material. It would be totally inappropriate in this case to represent all these final products as discrete batches which move through all stages of the plant because this is not a realistic representation of what happens. In this type of situation the larger batches of intermediate raw material simply feed the requirements of the smaller batches at the final product stage. In addition, in certain environments such as dairy processing, the scheduling function may be driven by raw material input to a certain extent, and it may not be possible initially to assign all this raw material to final products over the immediate scheduling period, so the system must contain some concept of "stock" intermediates. Therefore, an intermediate or final product should be able to be represented appropriately at any given stage of the production process, indicating how its production was arrived at through instantiation of activities concerned with its production. A recipe gives a more suitable representation for a batch of product than a discrete entity moving through the plant.

Intermediates and final products can be represented as an instance or partial instance of the recipe showing the activities which must be carried out for the manufacture of that product. The activities associated with a recipe are carried out through the allocation of plant resources, either as single plant items, or as linked routes of items. Thus as the activities required for a

Production Activities at Three Stages of Cottage Cheese Manufacture and Associated Intermediate and Final Product Requirements

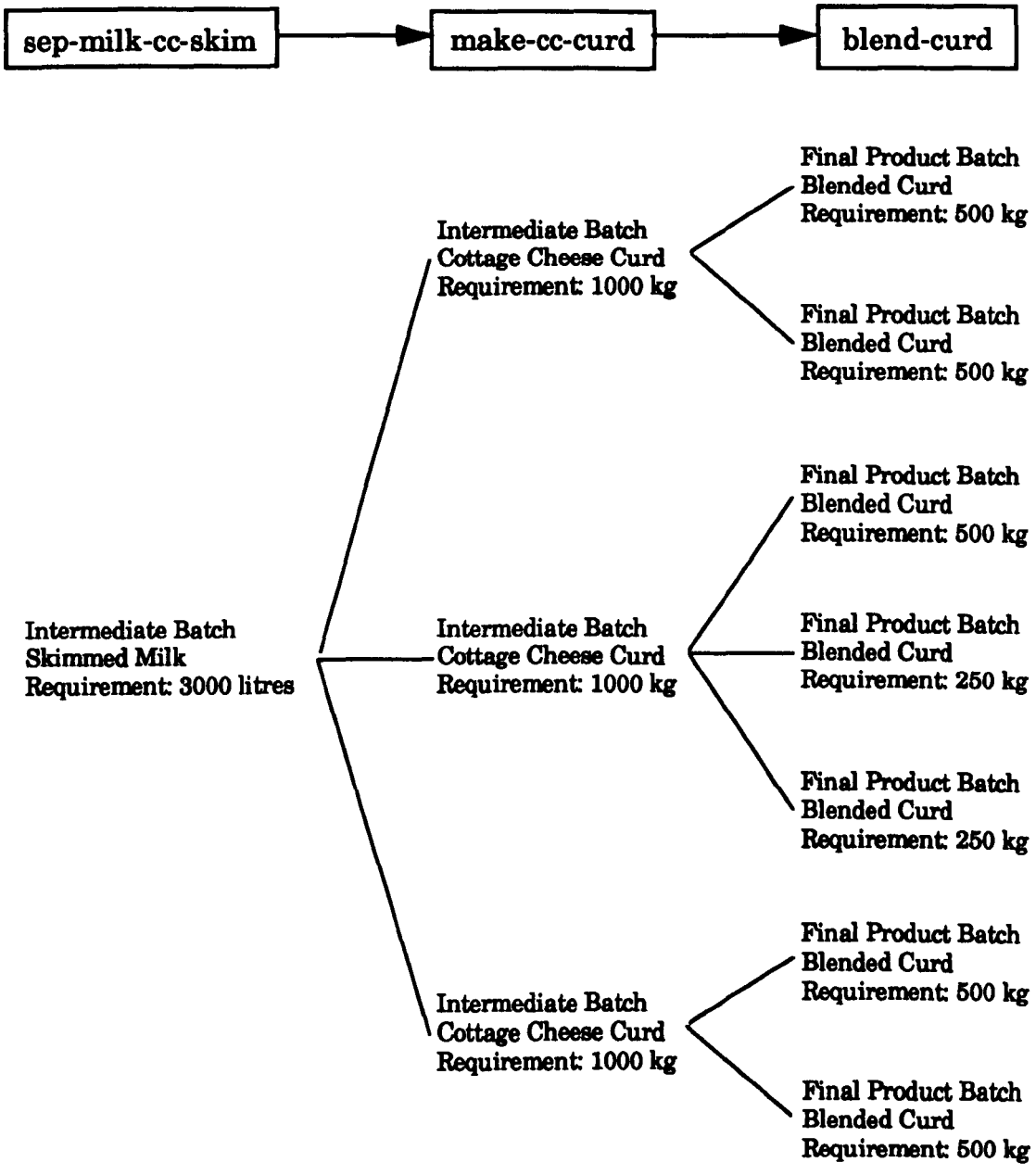



Figure 8.3. Variation in Intermediate and Final Product Batch Size Based on Final Product Requirements

particular batch of final or intermediate product are carried out, their representation within the batch can be linked to the resources used. For example, in the specific model of the Minsterley plant developed with the system, the primary goal was to look at the scheduling of skimmed milk intermediates to the various production areas of the factory. The volume requirement for the batches of these intermediates can be determined from aggregating the material requirements for specific final products produced in these areas. As illustrated in Figure 8.4. a batch of evaporator skimmed milk (**ev01**) can then be represented as an instantiation of the alternative activities by which it can be produced from the overall production recipe for skimmed milk concentrate (Figure 8.2). The batches of skimmed milk concentrate final product (**ev01a** and **ev01b**) which draw their raw material from this batch are represented as instantiations of the activity **evap-skim**, which is the final activity in the production recipe. It can be seen that there are two alternative activities by which the batch of evaporator skimmed milk (**ev01**) can be produced, using two alternative sources of raw material; either milk which is then separated, or skimmed milk "stock" already separated. Both of these activities are shown in the batch representation in the slot 'ACTIVITY STATUS' along with the amount of the final product requirement that has been produced by each activity and the status of each activity. The format of the representation is **[activity/ volume completed/ status]**. It is not desirable to split a specific volume requirement in advance between these activities since scheduling is a dynamic function based on current system state. The actual split between the alternatives for the way that the product is derived should be determined during the execution of the system rather than by a priori specification. At any stage of schedule development the amount of product left to be produced by a stage in the manufacture of one or more final product batches can be determined from the status of the intermediate products associated with that stage, and decisions can be made about how to progress the production considering alternative activities and potential

BATCH CODE: ev01

REQUIREMENT: 180,000 Litres skimmed milk

ACTIVITY STATUS: [ sep-milk-ev-skim/ 10000 / in-progress]

[pump-skim-stock-ev-silos/ 0 / can-start]

Batch of skimmed milk intermediate for two batches of skim concentrate final product ev01a, and ev01b

BATCH CODE: ev01a

REQUIREMENT: 21,000 Litres skimmed milk concentrate

ACTIVITY STATUS: [ evap-skim/ 5000 / in-progress]

BATCH CODE: ev01b

REQUIREMENT: 21,000 Litres skimmed milk concentrate

ACTIVITY STATUS: [ evap-skim/ 0 / can-start]

Figure 8.4. Intermediate and Final Product Batch Representation

process routes. This requires that procedures be incorporated in the model to update batch status with respect to the progress of associated activities.

### 8.3.1.Updating the Status of a Batch

At any stage of model execution there will be routes running that have been set up to process a particular batchsize of product to carry out a particular production activity. It is important to be able to correctly update how much of a given batch/ activity has been completed when a route shuts down after running for a certain time due to finite capacity constraints in the plant. The remaining amount of product to be processed can be converted to a **Remaining Batchsize** for the activity as follows:

**Remaining Batchsize = Batchsize/(Processing Rate * Time)**

The actual amount of **Product Output** from a semi-continuous or product transfer stage is given by the following calculation for a route:

**Product Output = Processing Rate*Time*output yield**

The actual amount of input material (**Product Used**) by a given route is given by the following calculation:

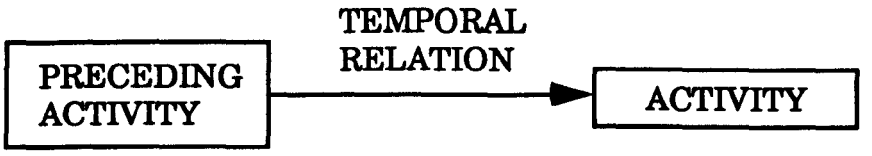
**Product Used = Processing Rate*Time*input factor**

Whenever a process route has to shutdown, either because the net batchsize on the route has been processed, or the limits of some finite capacity have been reached, the status of the activities associated with this stage of the batch's production can be updated on the basis of these calculations.

## 8.4.REASONING ABOUT SCHEDULING RESTRICTIONS ON ACTIVITIES

By representing batches of final products and intermediates within the system as partial or complete instantiations of a production recipe which indicate the status of the associated activities it can be determined at what stage of production a particular product is, which activities could be started, and how critical it is that the batch is progressed.

Using the precedence relations contained in the recipe for a product, and the status of the activities associated with a batch of a particular intermediate or final product it can be determined whether an activity can start or not. This is done by carrying out a search of an activity precedence network in a product recipe and using a "look-up" table which relates the status of linked activities to the temporal precedence relation defined between them. The look-up table indicates how the precedence relations between activities at two stages and the status of activities at the first stage affects the status of any activities at the second stage that they lead into. An activity have one of four different statuses; **cannot-start**, **can-start**, **in-progress**, and **finished**. The look-up table which describes the effects of the relations **BEFORE** and **OVERLAPS** on the status of two activities with respect to whether they can start or not is shown in Figure 8.5. A production activity can have more than one preceding activity leading into it, so the status of all these activities must be taken into account before any change in status from **cannot-start** to **can-start** is made. For example, an activity C is preceded by two activities A and B and the relation between A and C is **BEFORE**, and the relation between B and C is **OVERLAPS**. Even though the relation **OVERLAPS** simply requires B to be **in-progress** for the status of C to be set to **can-start**, the relation between A and C is more constraining and requires that this activity has **finished** before C could be set to **can-start**. Therefore, checks of precedence relations between two activities which indicate a status of **cannot-start** for the second activity




PRECEDENCE-RELATION TABLE:

| PRECEDING ACTIVITY STATUS | CURRENT ACTIVITY STATUS | TEMPORAL RELATION | NEW ACTIVITY STATUS |
|---|---|---|---|
| cannot-start | Any-status | Any-relation | cannot-start |
| can-start | Any-status | Any-relation | cannot-start |
| in-progress | Any-status | BEFORE | cannot-start |
| in-progress | cannot-start | OVERLAPS | can-start |
| in-progress | in-progress | OVERLAPS | in-progress |
| finished | cannot-start | Any-relation | can-start |
| finished | can-start | Any-relation | can-start |
| finished | in-progress | Any-relation | in-progress |
| finished | finished | Any-relation | finished |

Figure 8.5. Precedence Relation Table and Potential Changes to Activity Status

take priority to checks which involve the second activity and indicate a status of **can-start**.

## 8.5.PREFERENTIAL CONSIDERATIONS

Having determined which activities can start, then preferential considerations can be taken into account in deciding whether to actually attempt to start the activity at all, and which activities should be considered in priority to others. This part of decision making is based on a plant specific strategy, which can be expressed through production rules. For example, in the Minsterley plant a number of scheduling goals for the supply of the skimmed milk intermediate to the factory production areas were derived through interviewing the factory management, and these goals were used to develop a small set of strategy rules for the plant to drive the activity scheduling process.

Most scheduling systems use some form of heuristics to guide the direction of the schedule generation and these can be accessed by the rules used for activity scheduling. One measure used to guide the scheduling system is to calculate how critical batches are with respect to meeting due dates. There are a number of ways in which an estimate of the criticality of a particular batch of product can be assessed, for example with the use of critical path analysis, by determining how much total float is left in the time required against the time available to carry out the activities for the production of a particular product. The determination of the total float available has been incorporated into the BPS to give a measure of the criticality of a particular batch when assessing its priority

Other plant specific heuristics could also be used, for example, in the case of overlapping activities. The status of overlapping activities is determined from the precedence relations as described before. However, for any two stages there is a need to determine how far the first stage activities should have

progressed before the second can start and be carried out without interruption because the first stage has not produced enough input material for the second stage. This could be accommodated through a heuristic which relates the rates of the two stages, and the production volumes that are required.

A plant specific rule-base can be used to aid in making decisions about the preferential constraints which affect the scheduling of activities in a batch process plant. One constraint on activity scheduling which is a particular feature of batch processing is product stability. Activities which produce unstable intermediates are known as **Zero Wait (ZW)** or **No Wait (NW)** activities; process reaction and quality considerations dictate that such products should be processed immediately or within some known time span at the next stage. However, this constraint must be treated as preferential, because it is essentially a time constraint which can be broken without affecting the physical feasibility of the schedule in terms of the allocation of resources to activities. In this case, a heuristic rule could be included to incorporate a delay factor between numbers of batches of the product which started the activity. Such delay factors could be based on local knowledge or by carrying out experimentation with the simulation model.

## 8.6.RESOURCE ALLOCATION TO ACTIVITIES

Having determined that an activity should be scheduled then the procedures described for dynamic plant configuration must be applied to ensure that the allocation of resources to the activity takes into account the particular constraints of batch process plants and the preferential constraints at this level.

8.6.1.Production Activity Batch Size

After allocation of resources to a production activity, the final requirement for scheduling is to set an appropriate batch size on a product transfer or semi-continuous process route so that the times associated with the use of the resources and physical movement of product through the plant can be generated by the simulation model. As described in Chapter 7 the determination of this batchsize can be based on a simple Bill of Process (BOP) for the products involved in the activity where each activity is assigned an **input factor** and an **output yield** for calculating the amounts of input material used and product output. The input factors and output yields for the activities involved in the production of skimmed milk concentrate are illustrated in Figure 8.2.

The batchsize set on the route must be defined in equivalent units to the rate of the process unit which is being used in the route in order to correctly schedule events based on this rate. If the output product is the result of some sort of processing rather than a simple product transfer activity, then it is likely that the volume of product required from the process will have to be converted to a volume of product in the equivalent units used to describe the rate at which the process unit runs. For a simple serial route involving a single input and output product this is simply a conversion of the required output product volume by any output yield factor to convert it into the units used by the process unit which sets the rate on the route. For an activity involving separation or combination of products then a batch size which will be processed by the process entity setting the process rate needs to be determined for all the routes involved based on the Bill of Process and this is best illustrated by an example. For a separation activity using a single input product milk and producing two output products skimmed milk and cream it would be possible to describe different batchsizes for the activity based on the different product input factors and output yields. However, a batchsize for the

activity based on milk volume should be determined in this case because the rate of the separator which is used for the activity is described in terms of the rate at which it processes milk. On the basis of which output product requirement is "driving" the activity a net **Batchsize** can be calculated as follows:

**Batchsize = Remaining Demand/**process **output yield**

For example, if the main requirement from carrying out the separation activity is to produce a specific amount of skimmed milk for a production department, then the batchsize set for the activity would be skimmed milk requirement/0.9 (where 0.9 is the **output yield** for the process as shown in Figure 8.2.) to give the batchsize in terms of milk. This batchsize can then be set on both the skimmed milk production route and the cream production route configured for the activity, so that they will both be scheduled to complete this activity at the same time. The output yields set on the skimmed milk route and the cream route will ensure that the correct amount of skimmed milk and cream is produced for the net input batchsize determined on the basis of the skimmed milk requirement. The input factor in this case will be 1 and will ensure that the correct volume of milk is used by this activity.

It is possible to run more than one process route in parallel for an activity relating to a particular batch of product. There is no problem in configuring all the available routes assuming that constraints on plant item allocation are not broken and it may be desired that the batch requirement is run through two or more of these routes as quickly as possible. However, the batchsize for the product associated with this activity cannot simply be split equally between the routes because they may not necessarily run at the same rate. Therefore, in order to synchronise the routes so that the maximum amount of the batch is transferred or processed in the minimum amount of time, all the

routes involved in the activity should be scheduled based on the net rate of the processing units involved in the routes. The overall batch size for the activity can then be proportionally assigned to the routes based on this calculated running time and each route's individual rate as follows:

**Individual Route Batchsize=Calculated running time*individual rate.**

## 8.7.DISCUSSION

The control framework developed for the Minsterley plant milk movement scheduling problem used as an example in this thesis is based on a small strategy rule-base which was set up for the purpose of enabling the activity configuration procedures to be tested under dynamically changing plant conditions. A simple scheduling strategy has been implemented so far which does not ensure that the schedule developed would be as yet of an acceptable quality. It does not attempt to process batches of product until they become critical on the basis of total float. Schedules developed using this strategy are likely to contain production which fails to meet due date by a considerable margin and are therefore unlikely to be unacceptable to the plant management. However, the difference between schedules produced by this system and other systems described in the literature is that because the model of the plant contains procedures so that the true constraints on resource allocation imposed by the plant network structure are taken into account, and preferential considerations are accomodated through the rule-based dynamic configuration approach, the schedules produced by the system are feasible and could therefore be implemented on the plant. Thus the strategy for reasoning about the activities concerned with meeting production requirements can be improved and the improvements will be translated into better feasible schedules. This approach is therefore more likely to succeed in a real environment than the use of an approach which contains a simplified model of

the plant which allows physical constraints to be broken. In the latter case schedules which appear to be good, for example all production requirements are met, may in practise turn out to be infeasible on the plant, so the users are unlikely to have confidence in the system. If the system can be seen to produce schedules that are reproducible, then users can have confidence that it can be implemented on the plant.

# CHAPTER 9 IMPLEMENTATION OF A GENERIC HYBRID BATCH PROCESS SCHEDULER

## 9.0.INTRODUCTION

In Chapter 5 the reasons for developing a hybrid simulation for short term scheduling and control were described. The use of declarative languages for simulation has been put forward as a good way to overcome difficulties in modelling the intelligent and adaptive behaviour which is desirable in order to use simulation for short term scheduling and control applications. This is because declarative languages are particularly designed for knowledge representation purposes which enables a system developer to concentrate on representing the problem and ways to solve it. In a procedural language a system developer may have to develop a suitable knowledge representation scheme first before the problem and solution methods can be represented. Therefore more complex problems can be tackled in a given development time using a declarative language.

The two main representation schemes put forward in the literature on knowledge representation using declarative languages are frames (or objects or schema) and production rules. Frames are important because they enable a compact representation of an entity which can encapsulate all its attributes in a single structure making data manipulation easier than if it is spread through a number of arrays as is common in a procedural langauge representation. Production rules are a natural and flexible way in which to encapsulate the user defined preferences and complex control logic. In addition both these structures give a system inherent modularity because they are independent of the program structure. This is an important consideration when developing a generic modelling tool which is going to be sufficiently flexible to enable a range of specific models to be built.

On this basis the development of simulation systems in declarative languages has primarily been driven by the desire to improve the flexibility and ease of knowledge representation rather than to significantly change the way in which a simulation executes. Therefore the use of a declarative language for the procedural parts of a simulation generally only involves the translation of the procedural parts of the simulation into the declarative language format. This brings no great benefits in terms of simulation execution, but does bring some drawbacks such as the use of awkward programming constructs, and slow model execution. The use of a mixed language hybrid architecture has been put forward by a number of researchers as a way to overcome these difficulties by developing the individual parts of a system using the language best suited to each system part and then integrating them together into a complete system. This approach has been shown to be feasible on a small scale by Flitman [123] who commented that the approach needed to be applied to a full scale industrial problem to test its true worth. It was decided that the hybrid approach was a practical approach to take to the development of a generic system to carry out the procedures described in Chapters 7 and 8 for the configuration and activity scheduling of batch process plants, as it would enable knowledge representation at the required level of detail while still retaining the use of a procedural simulation model for maintenance of the dynamic plant status during model execution. Implementation of the approach involved addressing a number of issues:

1. The development tools to use for the declarative and procedural parts of the BPS.
2. The hybrid structure and the "split" between the declarative and procedural parts of the BPS.
3. The integration and co-ordination of the different parts of the BPS.

This chapter will discuss how these issues have been addressed, and also the testing of the BPS through the development of a model of the Eden Vale

Minsterley plant described in Chapter 7. Finally issues related to the use of the BPS in a "live" scheduling and control environment will be discussed.

## 9.1.TOOLS FOR SYSTEM IMPLEMENTATION

The aim of this project was to build a generic system applicable to the batch process industry. Therefore, as discussed by Bhattacharyya, Roy and Huang [42], relatively low level development tools were required to implement the two parts of the BPS to give as much control as possible during development over the structure and execution of each system part and the BPS as a whole. However, it was not desired to have to write the data manipulation procedures for a frame-based system from scratch, or to have to write an inference engine for the application of rules, or to write a simulation executive from scratch The development tools used for the BPS were chosen on this basis.

It was decided to use a general PROLOG development system to implement the declarative part of the BPS to give the desired flexibility for knowledge representation without being restricted during system development by requirements for model structure which might be imposed by the use of a higher level tool-kit such as Knowledge-craft or OPS5. A general PROLOG system can be relatively easily augmented to include both frames and production rules. In order to do this with the PROLOG system used it was decided to use the source code for a frame and rule-based tool-kit called FOOPS (Forward Chaining Object Oriented Production System) by Merritt [149]. This meant that a set of basic frame manipulation procedures and inference engine did not have to be developed from scratch, but they were "transparent" and the overall generic model structure could be developed as required and without compromise.

The low-level simulation tool-kit SEE-WHY was chosen for the procedural part of the BPS in preference to a higher-level tool such as SIMAN or even

WITNESS. It gives all the facilities required to build a simulation in terms of pre-written FORTRAN routines; for example, for scheduling events on entities, updating entity attributes, and displaying simulation execution. In particular it includes as part of the tool-kit the entity structure developed by Fisher [52] for representing finite capacity vessels and the procedures for scheduling psuedo-continuous events on these vessels on the basis of fixed input and output rates ensuring that these routines did not have to be written from scratch. However it still requires the system developer to write model specific event handling routines and the structure of the simulation in FORTRAN, so it was flexible enough to allow the generic model structure to be developed as desired

## 9.2.THE HYBRID STRUCTURE AND SPLIT BETWEEN THE DECLARATIVE AND PROCEDURAL PARTS OF THE BPS

### 9.2.1.System Execution Structure

The whole BPS structure is based on discrete event simulation because, as described by Roy [8], it is an efficient way in which to carry out forward scheduling taking into account finite capacity and the complex interactions between the entities in a manufacturing system. One of the key issues of the development of the BPS was therefore to determine which "world view" to use for the simulation execution, and where to "split" the BPS into its declarative and procedural parts.

#### A Three Phase Simulation Structure

Paul [150] discusses the importance of the executive world view with respect to model building and the incorporation of decision logic. He says that in the case of the process view, which is the basic executive of many object-oriented approaches, the model must be structured very carefully to '..get the

interrupts and delays and correctly registered' and the model may have to be forced to fit the structure. In the case of the event view, because the system is not globally updated at the end of every event a problem can occur if several events occur at the same time; at the end of a particular event the system will reallocate the resources that are released before it knows '..what other resources are going to be released by the other events that occur at that time.' [150] so it is working with an incomplete knowledge base. In both these cases, the structure of the model is complex and it is difficult to alter one part without causing undesired effects in another part of the model. The common objection to the use of the three phase approach has been lack of efficiency. However, with significant recent increases in computing power, and the use of techniques such as cellular simulation as described by Spinhelli de Carvalho [151] to structure the model this objection becomes less powerful than the reasons for using the approach. In the case of the three phase approach, the conditional phase always has the full global state available to it for decision making which is particularly important in modelling a batch process plant where so much of the system is interlinked. The model itself is a lot more modular and amenable to modification than the other approaches. Therefore, the three phase structure was determined to be an appropriate one with which to develop a rule-based simulation model for scheduling and control with global decision rules, rather than using the object-oriented approach as more commonly used for knowledge based simulation in which decision making is essentially local to objects and not based on a fully global update of the system state. As described by Doukidis [96] if the application of rules in the 'C' phase is carried out through an inference engine rather than a traditional three phase executive, then the order in which the rules are applied is no longer governed by the order in which they are written but determined through the use of a conflict set. In the case of route configuration where it is important that a route specific decision order for plant item selection can be applied which is different from route to route this was an important factor in deciding to structure the BPS in this way. The use of a three phase approach in the

simulation structure facilitated the mixed language hybrid structure by providing a natural "boundary" between the procedural and declarative portions of the BPS. The 'C' phase of the simulation which contains the complex control logic could be transferred to the declarative portion of the BPS where the application of scheduling and configuration rules could be controlled by an inference engine. The basic behavioural part of the model controlled through the 'A' and 'B' phases of the simulation could still be retained in the procedural language with a simpler but more efficient simulation executive to maintain the state of the plant correctly. This basic execution structure as developed for the BPS is shown in Figure 9.1.

BPS Data Structure

The other important issue to address was how to split the plant data between the two parts of the BPS. Each module contains some representation of the entities which are detailed to the appropriate level through a number of attributes.

Simulation Data Representation

It was decided to restrict the data representation of the plant entities in the Simulation Module to a representation of their individual attributes such as finite capacity, activity state, and other individual performance related data such as process rate. No data about the connectivity of plant items is included in the Simulation Module because PROLOG is much better suited to representing their structure and procedures for reasoning about the constraints associated with the plant network structure. However, the simulation does need to co-ordinate chains of plant items linked together to process a given batchsize of product taking into account finite storage capacity which may be able to be shared between routes. Also it needs to be able to co-ordinate plant items linked together in a cleaning route to carry out a cleaning

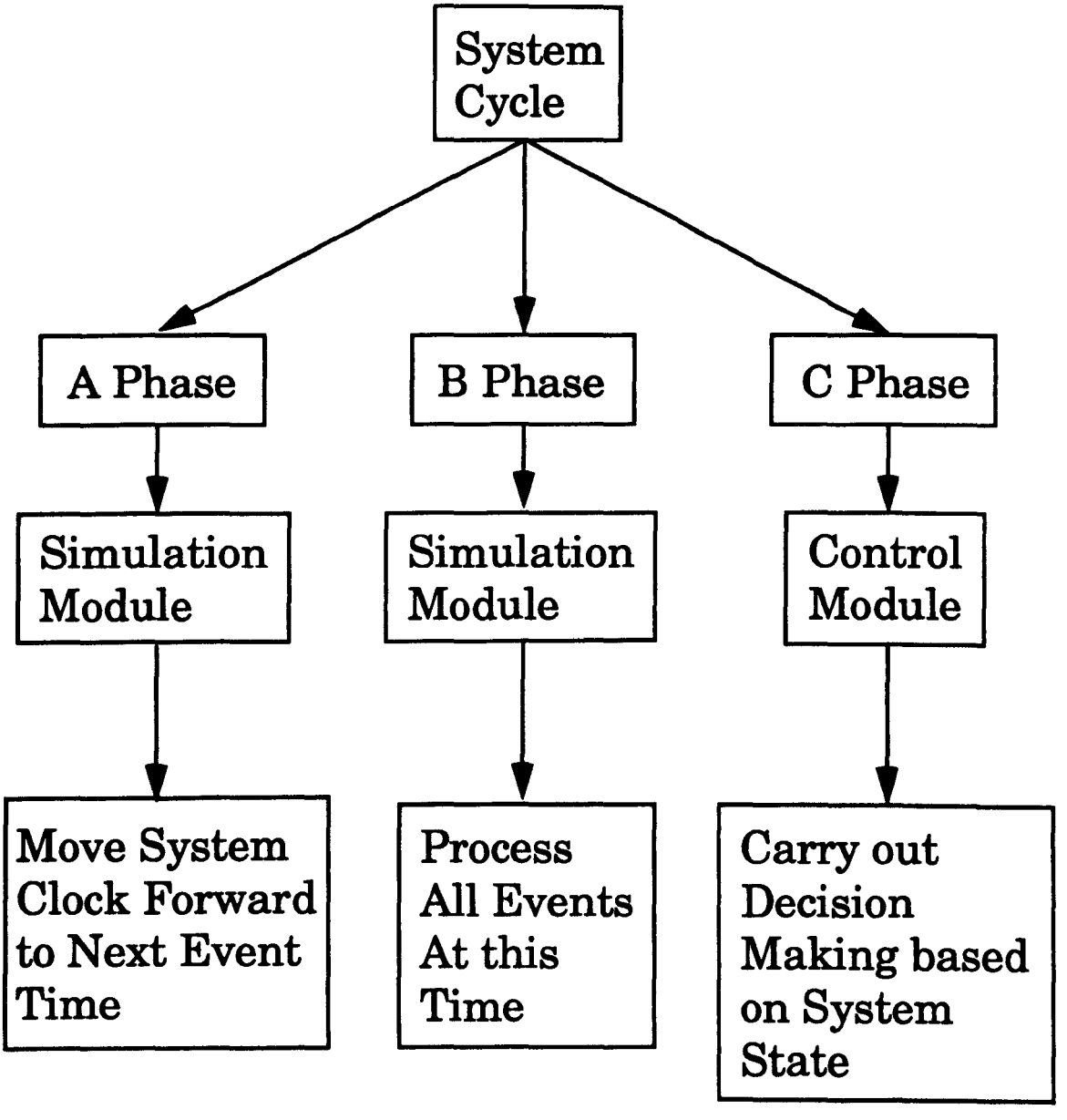



Figure 9.1. Basic Hybrid Three-Phase Execution Structure

activity. Therefore, a simple representation of the process and CIP sub-routes was also included in the Simulation Module, together with the link entities to connect them together to build up the routes configured by the Control Module. The process route/ CIP route representation used consists of three position sets corresponding to the sub-routes defined for a specific model in the Control Module. There is no information defined about what entities could make up the positions in a given sub-route. These sub-routes have an attribute to hold the batchsize to be processed through whatever route they are currently configured for, and an attribute to hold an input factor and an output yield in order to correctly maintain the material balance of the plant. The information about sub-routes which are to be set up in the model at any given point in the BPS execution is supplied from the Control Module in the form of a data set which may list one or more sub-routes, the entity which is to go into each position in the sub-route, and the batchsize which is to be processed by the route overall in the case of a process route or the time for which the route is to run in the case of a CIP route. Once a process or CIP route has been set up in the simulation according to the configuration derived in the Control Module the items in each adjacent position in a set of linked sub-routes are treated by the Simulation Module as if they are directly connected together for the duration of simulated time for which the process route runs. The product batches within the plant are not physically represented as entities in the Simulation Module. The SEE-WHY vessel entity maintains a record of how much product it contains at any point in model execution based on the procedures developed by Fisher [52]. The amount of product which is processed through a plant item is simply a function of the amount of time that the item is in use for, its rate of operation and associated input factors and output yields on routes. The Simulation Module also maintains the schedule data in terms of a process route schedule and a CIP route schedule. These are based on linked arrays which allow recording of data about the sub-routes within the model and the items which are set up in them as the simulation progresses. They can be examined at any time during

simulation execution and output as a schedule of plant operations at the end of the simulation run.

Control Module Data Representation

There is a frame hierarchy of plant item classes, although the possibilities for inheritance are relatively limited in this case as it is a flat rather than deep structure. The entity frame representations include slots for the static feasible connections as an AND/ OR structure and current input and output connections to allow the BPS to infer the existing constraints from the dynamic connectivity of plant items. The attributes of plant items relating to their individual dynamic status such as activity state and contents at last event are also included as slots on entity frame instances and are updated from the Simulation Module as BPS execution proceeds. In addition all the control structures for activity configuration and scheduling are fully represented as frames in the Control Module. These control structures are represented as a hierarchy of product recipes, process activities and potential process routes made up of sets of sub-routes. The control hierarchy for CIP routing consists of CIP circuits and associated sets of sub-routes making up the cleaning routes supplied by the circuits. There is a batch-list of final product requirements and batches of final product and intermediates whose production is being scheduled or controlled by the BPS are also represented as frame instances. In the representations of the **process_route** sub-route structures the key slots are **input_choices**, **processor_choices**, and **output_choices** which show what plant items are defined as being potential choices for a route. The **current_inputs**, **current_processor**, and **current_outputs** slots are used during the route configuration process to hold the results of each constraint check through the route and any preferential element selection using the configuration rules. The configuration procedure terminates when each of these slots only contains a single plant item and the route is now configured. Finally there are a number of other

frame classes which are important to the operation of the BPS. There is a frame to represent the current sequence of configured process sub-routes to be passed to the Simulation Module, and a similar one to represent the current sequence of CIP sub-routes. There is also a frame to represent the time of the last time update carried out by the simulation executive. Examples of specific frame instances as PROLOG data structures compatible with the FOOPS frame manipulation are shown in Figure 9.2. It can be seen from these examples that the use of frames to represent entities and their attributes gives a much more coherent and compact representation than the use of separate arrays to represent the different data types concerned with all entities of all classes, which is the approach which must be used in setting up the SEE-WHY model data representation. Also the flexibility of the PROLOG frame representation in allowing slot/ facet values to be lists of items (which can themselves be lists) has enabled a generic format for the representation of the plant network structure and routing to be developed without the necessity to set predetermined arbitrary limits as required by the language. For example, if arrays were used to represent routes and the choices for each position it is unlikely that they could be sized so that all routes in all batch plants could be represented with all potential choices represented for each position unless it was accepted that there would be a considerable amount of "wasted" space in most models defined with the system.

The user defined plant configuration and scheduling rules are also represented in the Control Module in PROLOG in the format specified for the FOOPS forward chaining inference engine. Again this format is flexible enabling complex decision rules involving any number of conditions and actions to be represented. The applicability of rules in the BPS is governed by goal contexts which are added and removed from the working memory as execution proceeds. Thus most rules have at least one goal as one of their conditions, and some have more than one to make the focus of the rule selection more specific. Examples of configuration rules in this format are

```
/* frame instance representing process route sub-route 'E1ST' holding the route
positions for the route input in the slot "input_choices", and the separator supply
lines in "processor_choices" */

        frinst(process_route,'E1ST',
                [ako - [val process_route],
                description - [val 'milk storage to seps '],
                links_up - [ val ['E1PR']],
                links_down - [val []],
                output_choices - [val [links_up,'LK10']],
                processor_choices - [val ['RLN1','RLN2']],
                input_choices - [val ['MSL1','MSL2',
                                      'MSL3','MSL4']],
                current_output - [val []],
                current_processor - [val []],
                current_input - [val []],
                current_yield - [val 1.0],
                input_factor - [val 1.0],
                remaining_batchsize - [val 0],
                current_batch - [],
                current_activity - [val sep_milk_ev_silos],
                activity_state_attribute4 - [val 1]],1).

/* frame instance representing storage vessel Milk Silo 1 'MSL1' */

        frinst(storage,'MSL1',
                [ako - [val storage],
                description - [val 'milk silo 1'],
                code - [val 'V01'],
                capacity - [val 227.3],
                contents_at_last_update - [val 227.3],
                current_contents - [],
                time_contents_last_updated - [val 0],
                current_rate_in - [val 0],
                current_rate_out - [val 0],
                time_product_out - [val 0],
                time_product_in - [val (-350)],
                time_last_stop_fill - [val (-200)],
                time_last_stop_empty - [val 0],
                contents_status - [val 1],
                system_event - [val maxcon],
                configuration - [val ['MSL1' ---> and: [i,o],
                                                    i ---> and:['OLN1','OLN2'],
                                                    o ---> or:['RLN1','RLN2']]],
                work_area - [val milk_reception],
                current_inputs - [val []],
                current_outputs - [val []],
                activity_state_attribute4 - [val 2],
                current_product_grade - [val 2],
                product - [val milk]],1).
```

Figure 9.2.a. Frame instances in FOOPS Format

```
/* frame instance representing the semi-continuous process element separator 2
'SEP2' */

        frinst(continuous,'SEP2',
                [ako - [val continuous],
                descriptor - [val 'SEP2'],
                code - [val 'P08'],
                configuration - [val ['SEP2' ---> and: [i,o],
                                                i ---> and: ['RLN1'],
                                                o ---> and: ['SPS2','SPC2']]],
                work_area - [val dairy],
                current_inputs - [val []],
                current_outputs - [val []],
                current_process_route - [val []],
                current_rate - [val 0.334/klitre_min],
                activity_state_attribute4 - [val 1]],0).

/* frame instance representing the process line 'RLN1' which provides input from
the milk silos to separators 1 and 2 */

        frinst(process_line,'RLN1',
                [ako - [val process_line],
                descriptor - [val 'RLN1'],
                code - [val 'P18'],
                configuration - [val ['RLN1' ---> and: [i,o],
                                        i ---> or: ['MSL1','MSL2','MSL3','MSL4'],
                                        o ---> and: ['SEP1','SEP2']]],
                work_area - [val dairy],
                current_inputs - [val []],
                current_outputs - [val []],
                current_rate - [val 0],
                current_process_route - [val []],
                activity_state_attribute4 - [val 1]],0).
```

Figure 9.2.b. Frame Instances in FOOPS Format

shown in Figure 9.3. It can be seen that there are a number of **call(predicate)** conditions and actions in these rules. These are direct calls to PROLOG and enable the application of the rules to be integrated with the procedures for making the route consistent during configuration and also integrated with other procedures which have been written to aid in "filtering" route position choices and ordering them according to particular attribute values during preferential element selection.

The overall basis of this flexibility for knowledge representation is the fact that an argument to a PROLOG predicate can be a list of items, which can themselves be lists and so on. The basic structure of both the frame and the rule under FOOPS is simply a predicate with a list as an argument containing either the frame structure or the rule structure. The procedures for frame manipulation and the forward chaining inference engine provided by FOOPS simply act on the data represented by these list structures identified by the predicate name as either a frame, frame instance, or a rule. One other advantage of using PROLOG for a system database is that it can be treated similarly to a relational database. This is due to its inherent procedural operating mechanism of depth first search with backtracking and the fact that a query will generate all solutions unless the process is terminated by the user. It is possible to examine classes of plant items or specific plant items and attribute values through specifying a simple query at the PROLOG intepreter command line, or a simple procedure written in the program which makes it easy to extract data about the status of plant items for decision making purposes.

## 9.3.BPS OPERATION

The following sections will describe the building of a specific plant model and important features in the execution of each BPS module. The integration of

```
/* strategy rule imm1: updates total float and activity state of product batches */

strategy imm1:
[goal - immediate_requirements with [status - active]]
==>
[call(immediate_requirements(factory)),
 call(immediate_requirements(ex_factory)),
 call(delf(goal,immediate_requirements)),
 assert(goal - setup_routes with [status - active]),
 assert(goal - system_input with [status - active])].

/* strategy rule fact1: whose actions call procedure "configure_critical_batches" to
instigate production route configuration */

strategy fact1:
[goal - setup_routes with [status - active],
 goal - factory_product with [status - active]]
==>
[call(configure_critical_batches(skim_concentrate)),
 call(configure_critical_batches(cottage_cheese)),
 call(delf(goal,factory_product)),
 call(delf(goal,setup_routes))].

/* configuration rule evs2: for the selection of a separator for configuration of
process route 1 for supply of skimmed milk to the evaporators. It contains the
procedure "select_element" as it's main action to select a separator which instigates
constraint propagation through the rest of the route. It also sets up the next
configuration decision to choose the skim line input */

rule evs2:
[goal - choose_separator with [status - active],
goal - sep_milk_ev_silos with [status - active],
activity - sep_milk_ev_silos with
                              [current_goals - Subroutes],
is_on('E1PR',Subroutes),
process_route - 'E1PR' with
                              [current_processor - Choices],
call(filter_choices_NOT(Choices,
                    [descriptor - 'SEP3'],Remaining1)),
Remaining1 \= []]
==>
[call(select_element(process_route ,'E1PR',
                    current_processor  - C,Remaining1)),
 call(delf(goal,choose_separator)),
 assert(goal - skim_line_input with [status - active])].
```

Figure 9.3. Strategy and Configuration Rules in FOOPS Format

the modules into the three phase structure will then be described in Section 9.4..

### 9.3.1.Building a Specific Plant Model

In order to build a model of a specific plant with the BPS, a user simply specifies a database for the Control Module of frame instances of plant entities, a control hierarchy of recipes, activities, process sub-routes, CIP circuits and CIP sub-routes, and the configuration and scheduling rules in the system format. An appropriate data structure for the Simulation Module must also be specified as a set of files for simulation sets, process entities, vessel entities, transport entities, transport entity routing, and link entities. These files are read in by the Simulation Module to create the specific SEE-WHY model of the plant. The hybrid model can now be executed to develop a short term schedule.

### 9.3.2.Control Module Execution

The Control Module drives the execution through the activity scheduling and route consistency and configuration procedures described in Chapters 7 and 8. The top level operation of the Control Module is a procedure **dialogue8** which calls the major components of the BPS as follows:

```
dialogue8:-
repeat,
pickup_data(Control),
release_resources,
(Control == 'STOP';
strategy,
send_link,
fail),
write('link terminated'),nl.
```

The main procedure calls are all contained within a repeat....fail loop which will continue execution until it receives a control command to **'STOP'** when it

jumps out of the loop and terminates the current execution of the BPS. The procedures **pickup_data(Control)**, **release_resources**, and **send_link** are concerned with communications, updating the status of the Control Module database and transferring the results of Control Module execution to a waiting Simulation Module. The main procedure which instigates the scheduling and configuration functions is **strategy** which constitutes the 'C' phase of the Control Module operation. The basic operation of this part of the BPS is shown in Figure 9.4.

The Control Module 'C' Phase: The Application of Strategy Rules

The procedure **strategy** initially adds to the working memory an active strategy goal which acts as a context to allow the strategy rules in the rule-base to fire. These are user defined rules which direct the BPS to attempt to configure particular batch/ activity combinations subject to the precedence constraints imposed by the production recipes defined in the Control Module database. The strategy rule-base for the BPS has so far been developed for the purpose of testing the ability of the activity route configuration procedures to cope with the real constraints on resource allocation to process routes. The rule-base therefore currently consists of a few rules to select batch/activity combinations on the single basis of criticality determined through calculating the total float on a batch. The first rules which are activated by the strategy context call PROLOG procedures to update the status of batches with respect to precedence and total float available for their production. The procedure **pc_check** determines whether a batch/ activity combination **can-start** by relating the status of the activities in the specific batch frame instance to the precedence relations between activities defined in the recipe frame instance. The procedures to calculate the total float on a batch were included to give a measure of the criticality of a batch of product with respect to meeting its due date. Total float was considered a useful measure because as described by Elmaraghby [152] it has been shown to produce good results in the scheduling

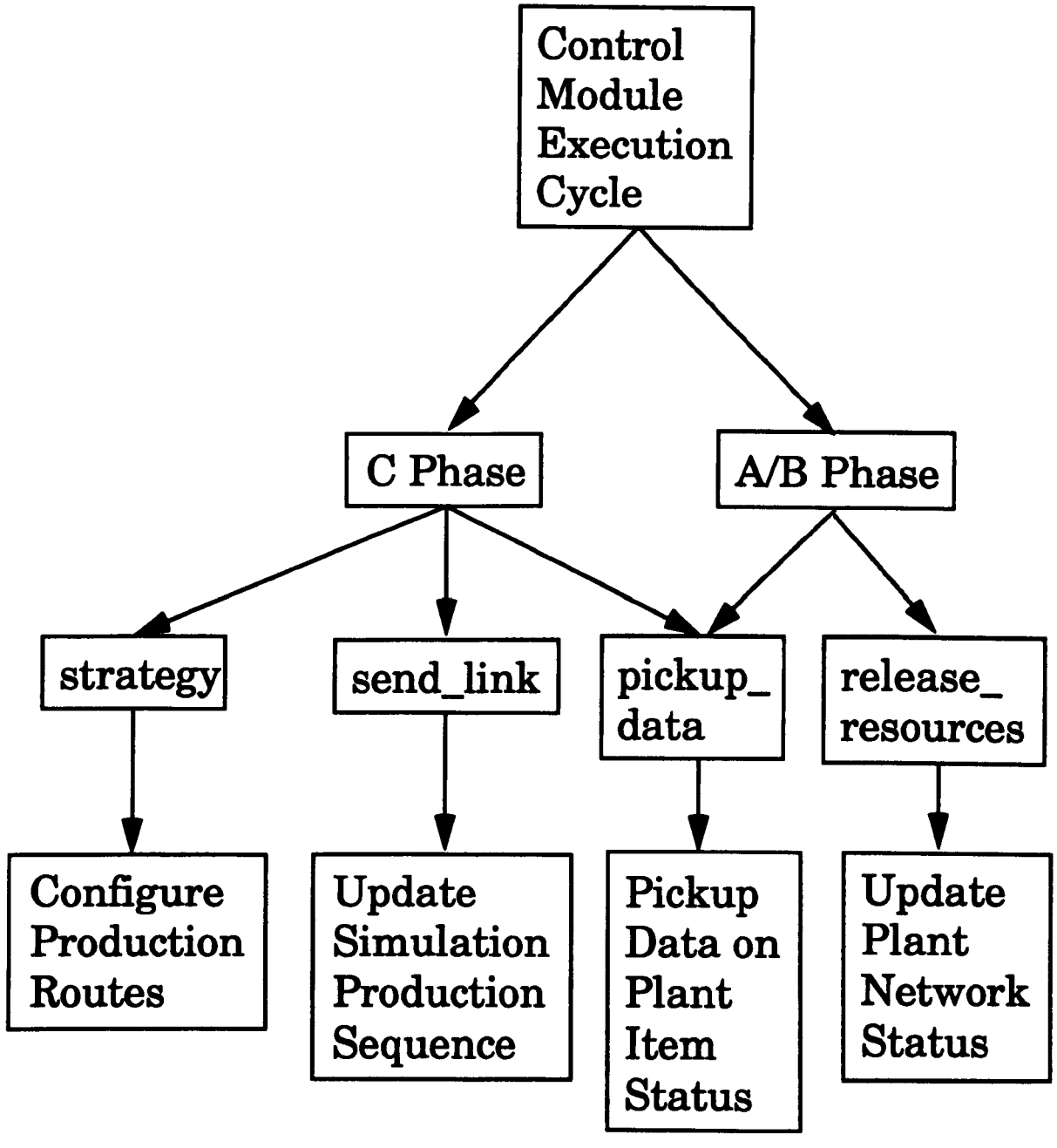



Figure 9.4. Control Module Cycle of Operation

of activities in resource constrained activity networks. The remainder of the rules in the current strategy rule-base determine which of the critical batch/ activity combinations to configure next and invoke the batch/ activity configuration level of the BPS through a call to the procedure **configure_activity(Activity,Batch_code)**.

Batch/ Activity Configuration

The call to the procedure **configure_activity** instigates the most important process in determining a feasible schedule with the BPS; it calls the procedure **find_routes3** which chooses a route for configuration and ensures that it is consistent with respect to the relation **CAN-CONNECT-TO** between route position choices throughout the configuration process. The main clause of **configure_activity** contains the following procedure calls to configure a serial process route:

```
configure_activity(Activity,Batch_code):-
find_routes3(Activity,Subroutes,Choices),
Subroutes \= [],
update_subroutes3(Subroutes,Choices)
addf(goal,Activity,[status - active]),
uptf(activity,Activity,[current_goals - Subroutes]),
go(rule),
batch_code(activity,Activity,current_goals - Subroutes,Code),
rem_batch(activity,Activity,current_goals - Subroutes,Code),
update_links_up(activity,Activity,current_goals - Subroutes),
update_links_down(activity,Activity,current_goals - Subroutes),
update_sequence(activity,Activity,current_goals - Subroutes),
.
```

The purpose and basic mechanism of each of the procedures in **configure_activity** will be described in the remainder of this section to show how procedures described in Chapter 7 have been implemented.

The main clause of **find_routes3** calls the procedures concerned with finding a set of subroutes to configure and making the route positions in them consistent with respect to the unary constraints and the binary **CAN-**

**CONNECT-TO** constraint on physical plant item choices. The procedures are called as follows:

```
find_routes3(Activity,Ordered_goals,Final_choices):-
activity_search(Activity,G,Goallist),
setup_initial_choices3(Goallist,
                                Ordered_goals,Initial_choices),
constraint_check3(Ordered_goals,
                                Initial_choices,Final_choices),
Ordered_goals \= [].
```

The procedure **activity_search** does a search of the activity to be configured to find a set of sub-routes which can make up a process route. The routes are found in the order in which they are written in the activity frame instance slot **alternative_routes** so any prioritisation for routing is determined by that sequence in this way. If a set of sub-routes is found which is currently available it is output from the procedure as **Goallist**. Then the procedure **setup_initial_choices3** carries out two functions to set up the route positions and their listed plant item choices for configuration. The procedure orders the list of sub-routes in **Goallist** on the basis of the links between them as specified in slots of their frame instances and outputs this as the list **Ordered_goals**; this ensures that a set of sub-routes to be configured can be presented in any order in the Control Module database and the configuration procedures will be able to function as required. It then finds the feasible plant item choices contained in the sub-route plant item choice slots and sets them up in a list of linked positions with the format:

[[Position 1 Choices],[Position 2 Choices],....,[Position N Choices]].

This list forms the input argument **Initial_choices** to the procedure **constraint_check3**, which makes the route consistent using variations of Mackworths [87] nc1, REVISE, and ac1 procedures. The REVISE procedure uses the plant item configurations and current connections to infer whether the relation **CAN-CONNECT-TO** holds between two plant items. This can be illustrated with an example of one of the tests for a potential connection that

was described in Chapter 7. The procedure **test_static** is used when the plant item being tested is a static item such as a vessel, or semi-continuous process item. The procedure contains a number of clauses representing the test cases for connectivity. The following clause from **test_static** checks whether a plant item **Element** with **OR** outputs in its **Configuration** can make a connection to a **Choice** in an adjacent output route position when the **Choice** has **OR** inputs:

```
test_static(Element,Choice,Configuration):-
is_on(o ---> or: Elements,Configuration),
is_on(Choice,Elements),
getf(Class,Element,[current_outputs - CO]),
CO == [],
getf(Class2,Choice,[configuration - CF,current_inputs - CI]),
is_on(i ---> or: Elements2,CF),
CI == [],!.
```

The first procedure call to **is_on** checks whether the **Element** being checked has an OR output. One of the advantages of the declarative nature of PROLOG can be seen here because the first argument to the procedure can simply be specified as the structure of the output list on the plant item frame instance configuration slot as **o ---> or: Elements**. If there is a list with this structure on the **Configuration** of the plant item the clause makes the next call to **is_on**, which checks to see if the potential connection **Choice** is on the the list of outputs **Elements** for **Element**. The clause now tests that the **Element** does not already have a **current_connection** through the test for an empty list "[]" as the value of the **current_outputs** slot of the **Element** frame instance. The clause then makes the same checks from the perspective of **Choice** to see that the test case holds in both directions.

A now consistent route is output from the procedure **find_routes3** in a list with the same format as the input argument:

[[Revised Position 1 Choices],[Revised Position 2 Choices],....,[Revised Position N Choices]].

It is quite possible that the **find_routes3** procedure will exit with an empty **Ordered_goals** list because there were no sub-routes available to configure or there were no sub-routes which could be made consistent under the current constraints on plant item availability. If the **constraint_check3** procedure exits with a consistent route then it is guaranteed that a route for the activity can be configured whatever decision order and plant item selection criteria are now adopted as long as the effects of each selection made are propagated through the route during the selection process. Therefore, preferential considerations can be taken into account, and a route can be configured and set up in the process route sequence to be passed to the simulation module when the Control module finishes it's current cycle of execution. The rest of the procedure calls in **configure_activity** are concerned with this process.

The procedure **update_subroutes3** updates the current item slots of the sub-routes which will make up the route to be configured with the currently available plant item choices which were returned by the **find_routes3** procedure.

The next two procedures **addf** and **uptf** are procedures defined in FOOPS which are used to add an activity configuration goal as the first context for the configuration rules, and to update the **current_goals** slot of the activity to be configured with the list of sub-routes which have been found with a set of currently available plant item choices.

The next procedure call is to **go(rule)**. This is the call to the top level of the FOOPS inference engine, and the BPS enters a forward chaining route configuration cycle applying relevant rules in the rule-base to configure the route based on a situation specific decision order as described in Chapter 7. Figure 9.3. shows one of the configuration rules for selection of a plant item for a route position in the format required by FOOPS. The key action carried

out by this rule is the call to **select_element**. The main clause in this procedure is coded as follows:

```
select_element(C,N,S - E,[Head | Tail]):-
getf(C,N,[S - E]),
delf_sv(C,N,[S - E]),
uptf(C,N,[S - [Head]),
create_goallist(C,N,Goallist),
setup_remaining_choices3(Goallist,
                                    Ordered_goals,Intermediates),
ac3(Ordered_goals,Intermediates,Final),
update_subroutes3(Ordered_goals,Final).
```

The first three procedure calls of **getf**, **delf_sv**, and **uptf** in this clause are all FOOPS defined predicates which carry out the updating of the sub-route current plant item position slot with the element selected from the currently available choices. The next two procedures of **create_goallist**, **setup_remaining_choices3** setup a list of the current state of the route at this stage of the configuration process in the format:

[[Route Position 1 Remaining Choices],....,[Route Position N Remaining Choices]].

The call to the procedure **ac3** invokes the binary constraint checking process to carry out constraint propagation of the effects of the selection of a plant item choice throughout the route. The final procedure call of **update_subroutes3** updates the sub-route position slots with the choices that now remain as a result of the constraint checking procedure. This **select_element** procedure is very important to the operation of the BPS because it ensures that throughout the dynamic configuration process a route is always consistent with respect to the remaining available choices in the route positions. It is thus the only way that an element should be selected for inclusion in a route. It is also the basis of a default route configuration procedure, because plant items can be selected for a route in any order and under any arbitrary criterion such as "first item on the list of choices", and the constraint checking procedures will ensure that the route derived is still

feasible under the current constraints on plant item availability that exist in the plant.

Having configured the batch/ activity the inference cycle of the **go(rule)** procedure exits, and the remaining procedures in the **configure_activity** clause are called. The procedures **batch_code** and **rem_batch** update the configured sub-routes with batch code and batchsize that will be processed when the route is set up in the simulation. The procedures **update_links_up** and **update_links_down** simply set up link elements in current route position slots to link the sub-routes together into the full route. The procedure **update_sequence** adds the data about the configured sub-routes to the **current_sequence** slot of the process sequence frame instance of the BPS. The data about the current sequence of sub-routes is set up in the slot with the following format which is required by the Simulation Module:

[[Sub-route-1,Batchsize-1,Position-11,Position-12,Position-13],
.
.
[[Sub-route-N,Batchsize-N,Position-N1,Position-12,Position-13]].

This enables a sequence of any length to be sent to the Simulation Module. In addition the sub-routes can be in any order, the simulation will co-ordinate the activities of the plant items in the route by following the link elements which indicate how the sub-routes are connected together. As each sub-route is added to the sequence its activity state slot value is set to "waiting set up" to indicate that it has been configured but not yet passed to the Simulation Module. On updating this slot on a process route a "daemon" is fired which updates the current connection slots of the plant items in each position in the sub-route with their new connections from the route configuration procedure. This ensures that the representation of the connections made in the plant network is always correctly maintained. It is important that the connections in the network resulting from the configuration of a route are updated before

any attempt to configure another route is made otherwise the choices for the next route may not be properly constrained.

At this point the **configure_activity** procedure exits, and the BPS returns to the strategy level of operation and any remaining actions to be carried out in the rule which just called **configure_activity**. Any further actions (which might include further batch/ activity configuration calls) are carried out, and the strategy level inference cycle continues until no more strategy rules can fire. At this point the **send_link** procedure is called by the top level **dialogue8** procedure of the Control Module, and the current process and CIP route sequences are sent across to the Simulation Module, and the appropriate events scheduled for the processing and cleaning activities which are to take place. The Control Module now becomes the passive part of the system and simply picks up data about the current plant status through the procedure **pickup_data(Control)** from the Simulation Module until it is reactivated at the start of another 'C' phase.

### 9.3.3.Simulation Model Execution

The main features of the SEE-WHY based Simulation Module are concerned with the translation of the process and CIP sequence supplied by the Control Module into system behaviour over time. It provides plant status information to the Control Module and the production schedule of resource use over time when the BPS is used in an off-line mode. The Simulation Module does not contain any of the configuration or scheduling decision logic; but only translates the scheduling decisions made by the Control Module into an actual schedule. The key routines concerned with setting up and co-ordinating the activities of the plant items represented in the process and CIP sequences will be described here.

The Simulation Module carries out the 'A' and 'B' phases of the three-phase operation structure and the purely procedural part of the 'C' phase. Although the SEE-WHY executive is based on the event scheduling world view it can easily be set up to operate as a three phase model using a function **STIME(IMODE)** supplied by SEE-WHY. If the function is called with IMODE set to 1 it returns the current simulation time. If the function is called with **IMODE** set to 2 it returns the time of the next event on the event list. This function can be used as part of an IF..THEN construct so that the call to any consequential events only occurs after all the scheduled events at the current simulation time have been processed. This is incorporated in a routine called **DECIDE** which forms the basis for the simulation to carry out the 'B' phase of the overall three-phase operation, as illustrated in Figure 9.5. **DECIDE** is called after every scheduled event has been processed but only calls the consequential events when the time of the next event on the event list is not equal to the current simulation time. After every scheduled event that occurs the Control Module database is updated with the results of the event on plant item status until there are no more scheduled events at the current time. Active control of the system operation is then passed back to the Control Module which carries out its activity scheduling and configuration as described before. The simulation module simply waits at this point and picks up all sequence data which is passed to it from the Control module. The route data in the current sequence to be set up in the model is recieved from the Control Module is the format described previously, and entered in one of two sets of linked arrays depending on whether it is process route or CIP route data. Each sub-route which is entered into one of the sequences is set to "active" so that the simulation will recognise that it is to be set up during the current 'C' phase of the system execution.

Having received all the current sequence information for the current simulation time from the Control Module the simulation model once again becomes active and enters it's own part of the 'C' phase. At this point the

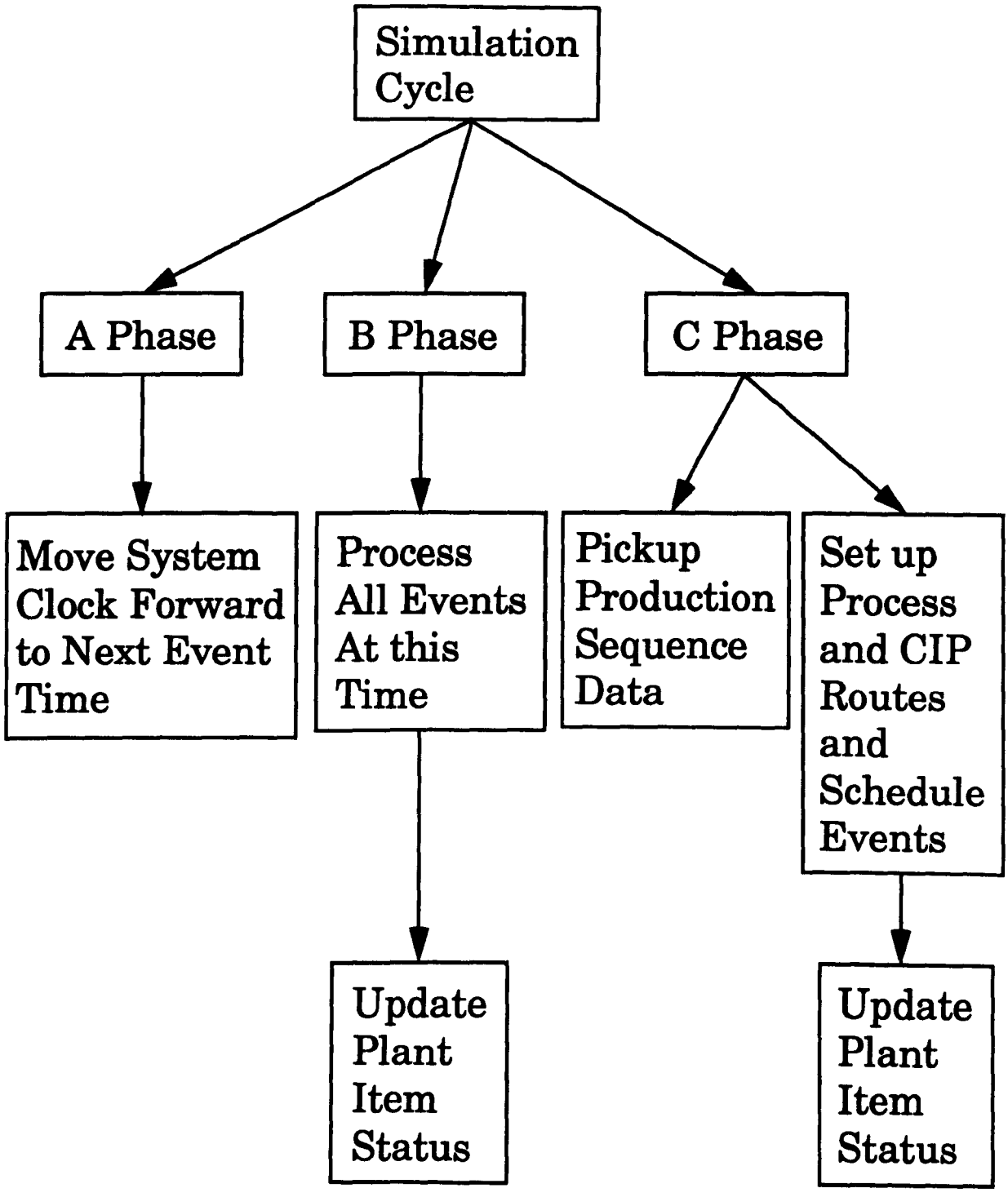



Figure 9.5. SimulationModule Cycle of Operation

Consequential events listed in the **DECIDE** routine are called to set up the routes in the model. There are a number of routines which find the plant items specified in the sequence data and set them up in the routes. For example, "Check Process Element Route Placement" **CKPRRT** finds and places semi-continuous process entities in routes, and "Enable Fill and Empty of Vessels" **ENFILL** finds and places vessel entities in routes. As each route is setup with its current plant item configuration it is set to "read" in the current sequence. Some plant items such as vessels and process lines can be shared across more than one activity process route, and because each route will probably be set up to process a different batchsize at a different process rate this means that these entities could be required to have a number of events scheduled on them. This is handled in the simulation by representing these entities in all the routes in which they are placed as a "shadow entity" as described by O'Keefe and Davies [33]. Each of these "shadow entitities" is a different entity from the "real" plant item entity, and holds a pointer to this real entity. This means that route specific events can be scheduled on the "shadow entities" and item specific events on the real entitities irrespective of the number of routes which are running.

The **LKxx** entities which occupy input and output positions in sub-routes are used to link the sub-routes making up a route for an activity into a "chain" which can be followed by the simulation when scheduling the relevant process or clean activity finish times on the entities in the route. Any entities which are in adjacent positions in the route, or separated only by a **LKxx** entity are directly connected in the route as configured by the Control Module. The first position in a linked chain of sub-routes represents the source of the route, and the last position in a chain represents the sink of the route. There are two other classes of entity used to set up routes which do not represent "real" plant items. These are **INPT** and **OUTP** entities which can be placed in the source or sink position of a route and represent one or more open ends to the route. These are used to represent links to parts of the plant which are not

currently represented in the model. For example, there are several parts of the Minsterley plant which are not currently represented in the test model, but take some of the output from the milk processing and skimmed milk routing which is represented. The outputs from the routes which link into these areas are therefore represented by **OUTP** entities. The other use of these entities is in split routing where they are used to avoid the double placement of some entities in routes and possible problems with material balance calculations. In the case where a process element takes input from a single source and separates it into two or more components the sub-route structure means that the process entity must be represented in two or more sub-routes. However, there is only a requirement to represent a single source entity for the activity input. Therefore the **INPT** entity is used as the input to the process entity in all other sub-routes. The **OUTP** entity would be used in the same way where there is combination of a number of inputs through a single process entity into a single sink entity. The uses of these entities are illustrated in Figure 9.6. which gives an example of a route set up for the separation of milk in the model.

The final routine called during the 'C' phase of the Simulation Module is "Start Processing" **STPROC** in which events specific to the "finish processing activity route" are scheduled on the entities which have been set up in routes. The time at which these events are scheduled for corresponds to the length of time that the route should run based on the batchsize passed with the configuration data and the process rate of the controlling process entity. Each sub-route involved in a process route will have been sent a batchsize from the Control Module, but in order to correctly co-ordinate the events in all the routes they must all be set to run at the same overall rate for the route which is provided by the controlling process entity. The simulation uses a function called **PROCESS_RATE** which is called when the relevant events are scheduled on the entities in each active subroute in the process sequence. This function follows the links from a particular sub-route to any other sub-routes

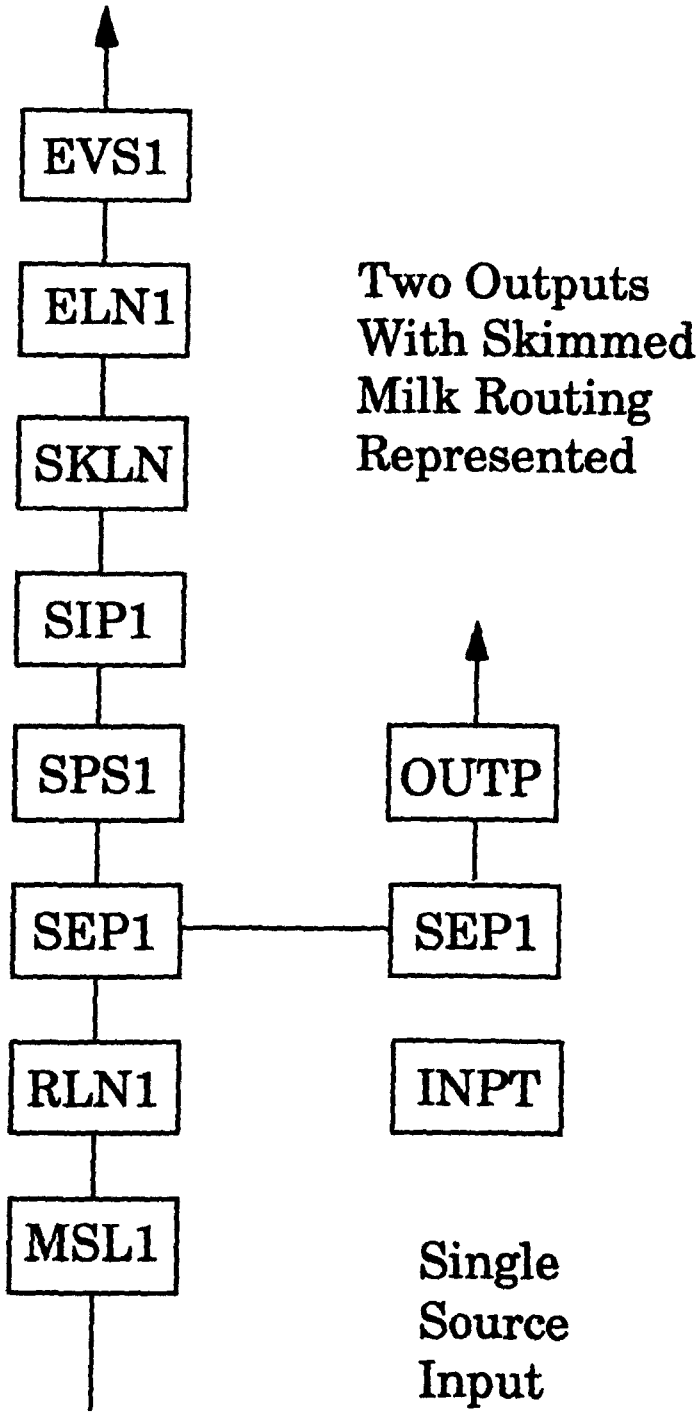



Figure 9.6. Use of INPT and OUTP Entities

to which it is linked and returns the overall process rate of the controlling process entity. This rate can then be used to schedule the events on the particular sub-route entities and set the process rate on shadow entities representing process lines which are passive process entities because they do not have their own process rate. Sub-routes can be presented in the process sequence to the simulation in any order and representing routes of any length (although this is effectively limited by the overall size of the process and CIP sequence array in the simulation). The mechanism to follow the links from a particular sub-route will ensure that the overall route is always correctly co-ordinated. Material balance calculations are also carried out during this routine. A shadow vessel acting as a source or sink takes the rate of the process entity controlling the route, adjusted by the input factor or output yield, and adds this adjusted rate to the net rate on the real vessel that it represents. This causes the system events corresponding to the real vessel to be re-scheduled if necessary in line with this change. This event scheduling process on individual routes assumes that there will be sufficient capacity available for the output product from a route and sufficient input material available. However, the effects of finite capacity are taken into account when a route is running. This is handled through the SEE-WHY vessel entity system events for maximum and minimum contents which are scheduled based on the net rate on a real vessel rather than the route specific events scheduled on the shadow vessels.

Having scheduled all events on all active sub-routes for process and CIP activities and updated the status of the corresponding plant item representations held in the Control Module the simulation enters the 'A' phase of the three phase cycle and moves the simulation time forward to the first scheduled event held in the event file. This could be an event related to an element held in a route or it could be a system event related to a vessel finite capacity limit or threshold level trigger. It could also be some other event which can be scheduled by the simulation such as the arrival of a vehicle, the

end of a batch reaction activity, the end of the simulation scheduling period, or an interrupt event. In a system such as this the simulation could be set to run for a long period of time without any plant item or route specific event occurring. There could be a time missed when it was required to attempt to set up another route in the simulation or dismantle some running routes because of shift patterns for example. Therefore it is important to be able to schedule an interrupt event to account for this type of occurrence.

After the simulation has moved forward in time to the next scheduled event it enters the 'B' phase in which all current events are processed. For each scheduled event at the current clock time the appropriate routine is called which will update the status of the entity affected in the simulation model and also send the relevant status update data to the Control Module. Although the **DECIDE** routine is called after every scheduled event the use of the function **STIME(IMODE)** prevents any of the consequential events being called until there are no more currently scheduled events, thus giving the system its three phase event behaviour. Most scheduled events occur on entities which are in process or CIP routes which are shutting down and being dismantled, and result in the entity being removed from the route. In the case of a shadow entity the real entity must be updated with the effects of the event. Real vessels will have their net rate adjusted by the removal of the route input or output rate held by the shadow entity, and their actual activity state may be altered as well, changing from "filling and emptying" to "filling only" for example. There is effectively no difference in the shutdown of a route because it has met its target batch size or because it has had to shutdown due to a capacity limit being reached through one of the system events occurring. In the second case the route specific process events on the plant items in the affected route are simply re-scheduled for the current time and then processed at the current time. This means that the number of events to be processed at the current simulation time may increase during the 'B' phase as route specific events are re-scheduled. However, because of the way that the

simulation is structured these events will all still be processed before it moves on to any consequential events. Also the remaining batchsize not processed on the route is calculated and passed back to the Control Module along with the other route and entity data. At the end of the Simulation Module 'B' phase when the next event time is different from the current simulation time control of the system execution is passed back to the Control Module.

### 9.3.4.Recording Simulation Execution Information for Scheduling

As a process route or CIP route is set up in the Simulation Module, the data about the configured route is recorded in the schedule arrays so that a production schedule can be output from the model at the end of the system execution. For each process sub-route which is set up the batchsize to be processed and the time at which the route starts processing are recorded. In addition the description and code of the plant item in each route position is also recorded together with the current contents if the item is a vessel. This allows tracing of changes product volumes held in vessels over the duration of the simulation. For each CIP sub-route which is set up the plant item in each position is recorded together with the time the route starts cleaning. The schedule arrays also contain data fields to record information about the time when a route finishes processing or cleaning, and in the case of a process route the product volume in any vessels in the route and any remaining batchsize not processed.

### 9.3.5.The Control Module 'B' Phase

The Control Module picks up data about the plant item events during the Simulation Module 'B' phase and updates the status of its own plant item frame instances. It also adds each sub-route which is dismantled into a **route_shutdown** slot of the process sequence frame instance. The Control Module uses this data to dismantle its representations of the current routing

in the plant, and remove current connections between plant items so that the status of the plant will be correctly updated for the next activity scheduling and configuration phase. When control is returned to the Control Module the top-level procedure **dialogue8** calls the procedure **release_resources** to carry out this updating of plant routing and network status. The procedure **release_resources** makes the following procedure calls for this purpose:

```
release_resources:-
re_build_activities,
getf(process_sequence,M,[activity_shutdown - AS]),
update_batch_data(AS),
remove_connections(AS),
dismantle_subroutes(AS),
delf_sv(process_sequence,M,[activity_shutdown - AS]),!.
```

The procedure **re_build_activities** puts the returned sub-routes in the **route_shutdown** slot of the process sequence frame instance into ordered sets of sub-routes corresponding to each activity which is being closed down. This is held as a list of activities in the slot **activity_shutdown** of the process sequence. It is necessary to carry out this sorting procedure because the events on plant items in sub-routes will not occur in such an order that the dismantling sub-route data will be passed back to the Control Module in the order in which they make up activities.

The procedures **update_batch_data**, **remove_connections**, and **dismantle_subroutes** all act on this list of ordered sub-route sets. The procedure **update_batch_data** updates the temporal activity status and remaining batch size of a product batch which was being processed by a particular route. **remove_connections_act** goes through the sets of sub-routes making up a list of activities being dismantled and removes the current connections between plant items in adjacent route positions in each activity. The procedure **dismantle_subroutes_act** takes the sets of sub-routes from the activity list and resets them as available for use in setting up another route.

Having released the plant resources for use again a single cycle of the three phase structure has been completed and the system operation re-enters the Control Module 'C' phase where decision making concerned with activity scheduling and configuration is carried out as described earlier.

## 9.4.MODULE INTEGRATION

The preceding sections have described how the Control Module and Simulation Module operate together to carry out off-line scheduling using the three phase event structure. In order to co-ordinate the operations of the the two parts of the BPS they had to be fully integrated. As described by Flitman [123] the integration of PROLOG and a FORTRAN based simulation model is feasible and practical through a configuration in which each module is run on one of two separate PCs directly connected via an RS-232 cable. Direct communication of data between the two modules can be achieved in this configuration using a communications interface written in assembly language for each module which gives it control of the serial port on the machine on which it is running. This approach to integration was taken for this project because it was desired to have the Simulation Module and Control Module completely separated from each other. One of the aims of the project has been to develop the Control Module so that it could in principle be disconnected from the simulation and connected to a real plant to act as a batch plant management and configuration system in real time. Thus the development of the communications interface has been part of this process. Also, having the modules operating on separate machines aided system development considerably, because the execution of both modules is always 'visible'. This enabled the co-ordination between the two modules to be seen easily from the point of view of the the effects of Control Module decision making as routes are setup in the simulation, and the effects of events on running routes in terms of plant status data passed back to the Control Module.

Carrying out the integration of the Control Module and the Simulation Module involved two parts. The development of the communications link between the two modules through an assembler interface for each module, and the development of data communications protocols between the two modules so that they would be correctly co-ordinated in the three phase operation and in the transfer of specific collections of data.

### 9.4.1.Development of the Communications Link

It was desired to make the communications link between the two modules as flexible as possible so that it would be easy to set up the appropriate data transfer and co-ordination protocols. Also it was desired to make the BPS amendable by a systems developer without having to resort to coding in assembly language each time an amendment was made. Flitman describes the development of PROLOG predicates and FORTRAN callable routines written in assembly language for the transfer of data in his system, and the same approach was adopted here to give the desired flexibility. For each data type which could be used in the system a piece of assembly code has been written to carry out a data transfer in both directions using a Data Available (DAV)/ Data Acknowledge (DAK) handshake as described by Craine and Martin [153]. For example in the SEE-WHY model interface there is a piece of assembly code for transfer of a 2-byte integer from FORTRAN via the serial port to some receiving device connected to the RS-232 cable, and the code is directly callable from the SEE-WHY program as a subroutine **SENDINT(Integer)**. In the PROLOG model interface there is a corresponding piece of assembly code for receipt of a 2-byte integer from some device connected to serial port of its machine via an RS-232 cable; again the code is directly callable from the PROLOG program as a predicate **recint(Integer)**. The assembly code required for each class of data transfer required for the operation of the system was written and then linked to and compiled with the PROLOG or

FORTRAN module code. For each type of data transfer in one module there is a "mirror" in the other language to carry out the transfer through the correct handshake. Having done this each module now contains a set of data transfer predicates or routines which are directly callable from the respective language as shown in Figure 9.7. With these predicates and routines it was possible to incrementally develop the required co-ordination and data transfer between the two modules.

### 9.4.2.Data Transfer and Module Co-ordination

When the BPS is operating in integrated mode one of the modules is always active while the other is waiting and receiving status updates from the active module. Transfer of sets of data between the two modules is achieved using specific "protocols". Whenever a module is waiting it may receive a protocol number from the active module. In response to this it calls up its appropriate "mirror" protocol to carry out the data exchange required. This data transfer co-ordination is shown in Figure 9.8. which shows the **pickup_data(Control)** procedure. A specific protocol in each module simply consists of a set of calls to the appropriate data type transfer predicates/ routines which are required for the data exchange. For example, when the Simulation Module is waiting one of the data transfers that occurs is an update of the process route sequence with a set of configured sub-routes from the Control Module. This is a fairly complex data transfer involving all classes of data type, and it is not known in advance by the Simulation Module how many lines of sequence data it will be receiving. However, using the basic protocol format each line of the sequence can be correctly received by the simulation embedded in a procedure to take all the lines that are due to be sent. This receiving protocol as it exists in the Simulation Module and the corresponding sending protocol in the Control Module are shown in Figure 9.9.. There are a number of other protocols that have been developed for communications between the two modules including plant item status updates in the Control Module (such as vessel contents

| PROLOG predicates | FORTRAN routines | Purpose |
|---|---|---|
| recint(Integer)<---- | SEND_INT(INTEGER) | send a two byte integer from FORTRAN to PROLOG |
| sendint(Integer)---> | REC_INT(INTEGER) | send a two byte integer from PROLOG to FORTRAN |
| recatom(Atom)<------ | SEND_DATA(CHAR*4) | send a four character atom from FORTRAN to PROLOG |
| sendatm(Atom)------> | REC_CHAR(CHAR*4) | send a four character atom from PROLOG to FORTRAN |
| recreal(Real)<------ | SENDREAL(REAL) | send a four byte real number from FORTRAN to PROLOG |
| sendreal(Real)-----> | RET_REAL(REAL) | send a four byte real number from PROLOG to FORTRAN |

Where ---> indicates the direction of the data transfer.

Figure 9.7. Basic Data Transfer Predicates and Routines

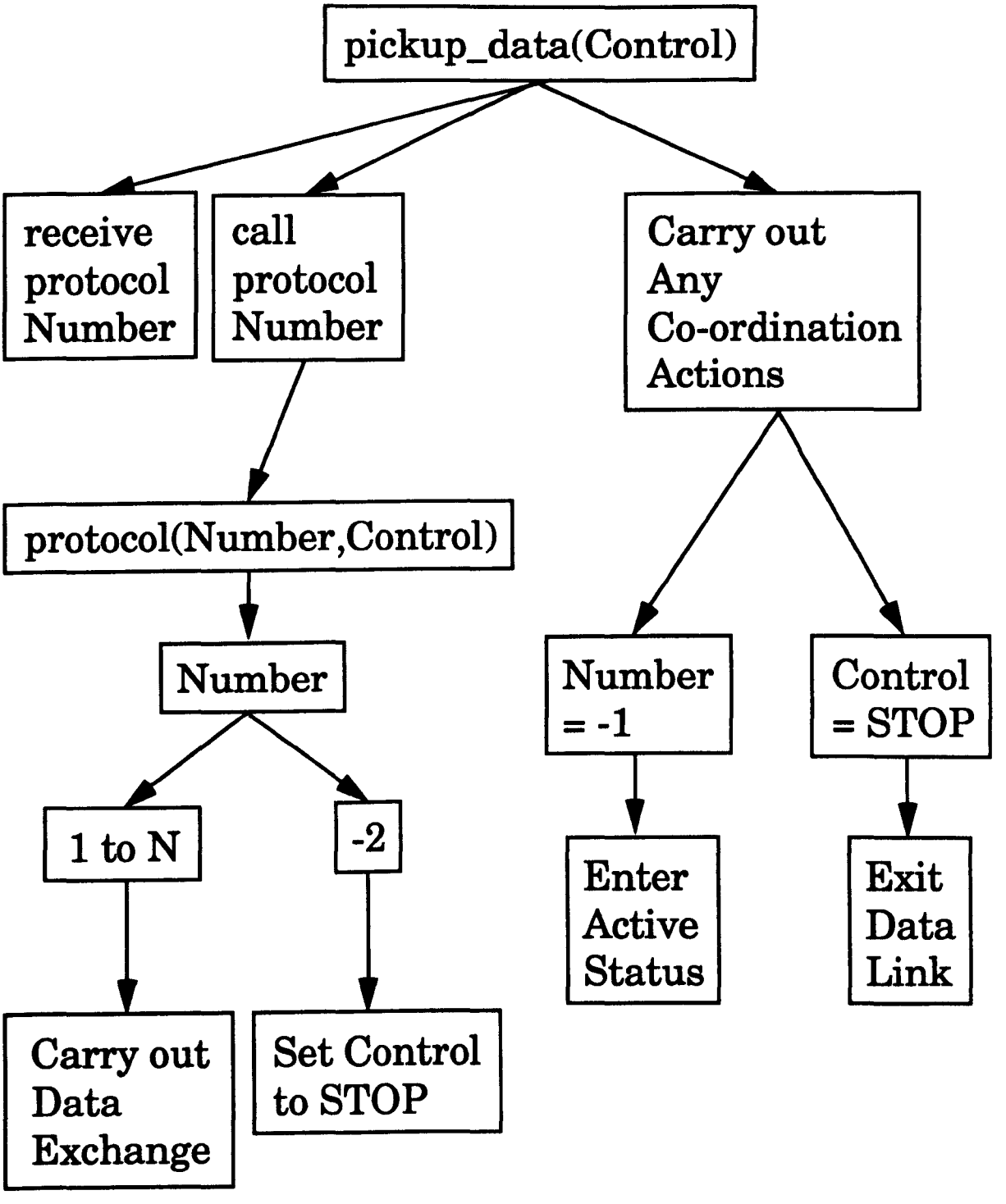



Figure 9.8. Control Module Data Transfer and Status Co-ordination

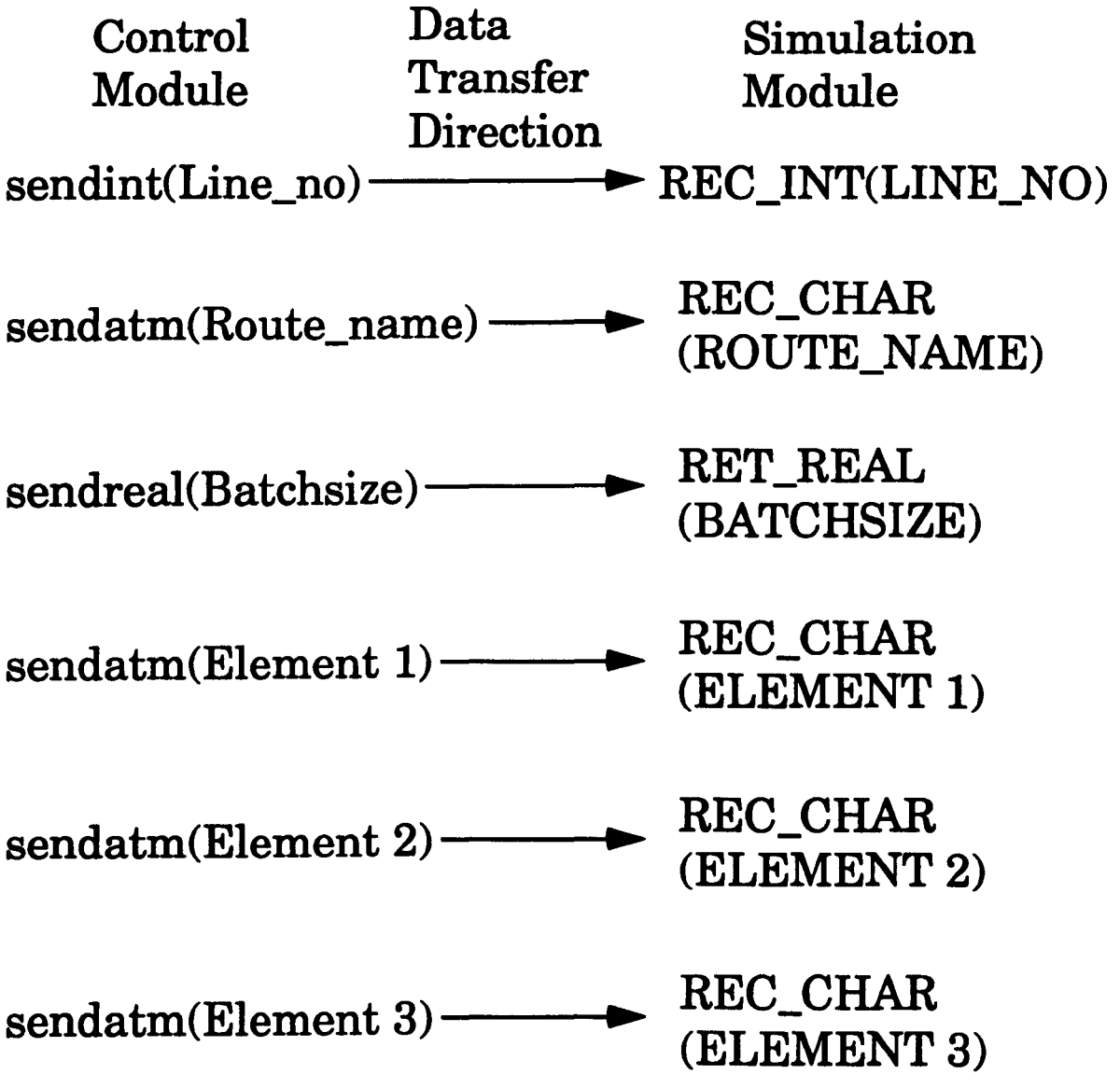



Figure 9.9. Sequence Update Protocol

when a change in state occurs), and a routing status update when routes shutdown in the Simulation Module. Using the basic protocol format the co-ordination between the two modules was also easily acheived. At any point in the BPS operation a waiting module is set as the active module on receipt of a protocol number -1. This co-ordination through these protocols to achieve the system integration with a three phase operating structure is shown in Figure 9.10.

The data link between the two modules is also easily terminated at the end of a scheduling session, or if it is desired to interrupt model execution for any reason. The waiting module receives a protocol number -2 to indicate that the link is terminated, and the two modules then return to their top-level and wait for user interaction.

## 9.5.CURRENT BPS IMPLEMENTATION AND TESTING

As discussed in earlier chapters it was evident from other work in the areas of batch plant scheduling, simulation of batch plants, and AI based techniques for scheduling that there has been a considerable amount of work done at the level of activity scheduling of batch plants and manufacturing systems generally. However, it appeared that the configuration of batch plants to carry out production activities was treated in a simplified manner which would result in infeasible schedules. The aim of the research has been to address this through the modelling approaches described, and to demonstrate the applicability of these approaches through a hybrid implementation. It has not been the intention during this project to build up a comprehensive rule-base for configuration or activity scheduling of the Minsterley plant although these would have to be covered in a full-scale implementaion of the BPS in an industrial environment. The testing of the BPS has therefore been carried out for the following purposes:

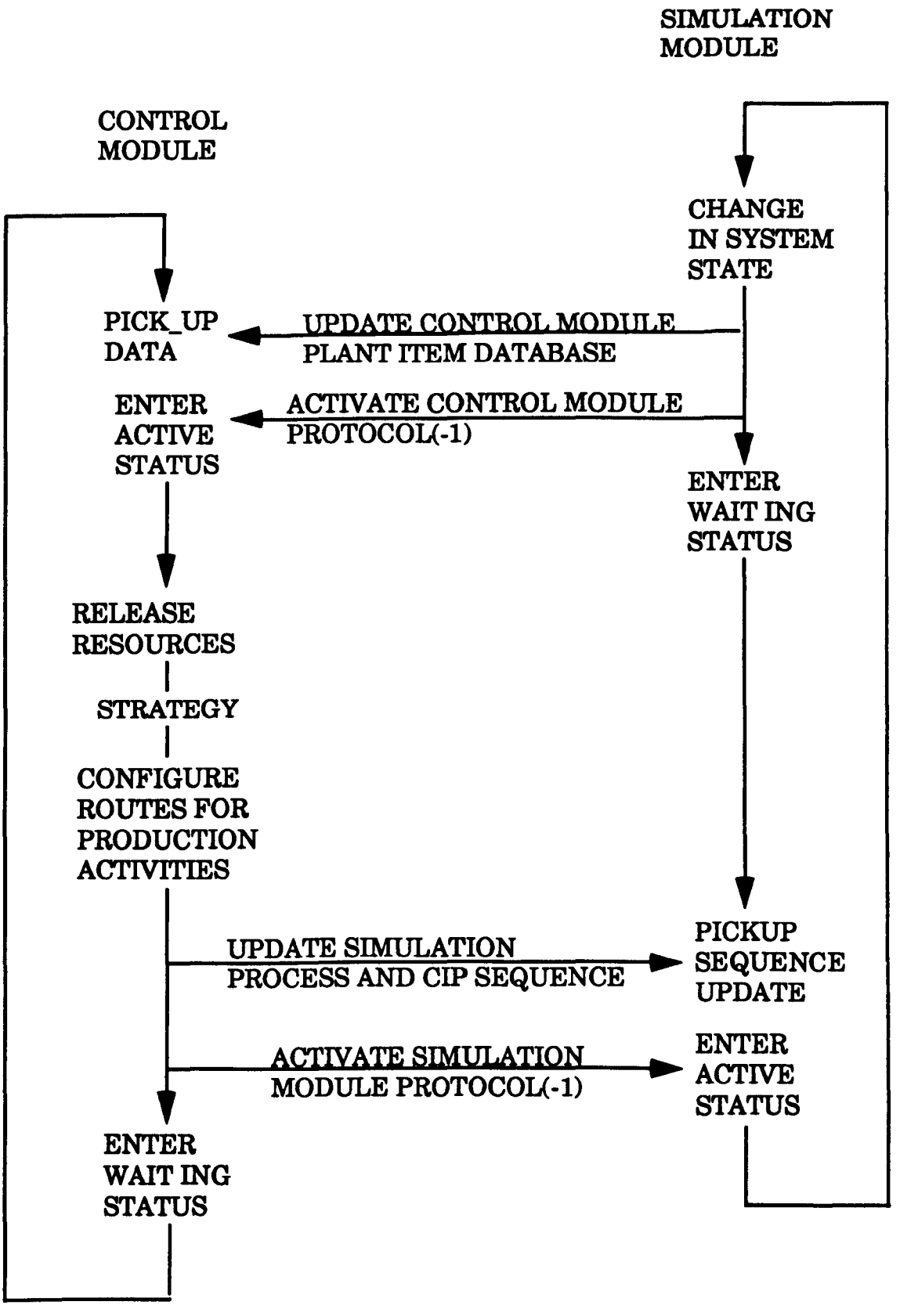



Figure 9.10. System Integration

1.To test the ability of the BPS through its knowledge representation scheme and constraint checking procedures to configure production routes through a plant network in the light of actual dynamically changing constraints on plant item availability.

2.To test the ability of the BPS to cope with the use of resources which are shared across production routes and to cope with the effects of finite capacity in such circumstances.

3.To test the ability of the BPS to co-ordinate correctly the activities of variable numbers of plant items linked together in production routes and correctly maintain the material balance of the plant.

4.To test the synchronisation between the two modules so that the Control Module can correctly react to changes in plant status, and the simulation can correctly respond to control data from the Control Module.

Some examples from the Minsterley plant model database and the results from a trace of system execution which are listed in full in Appendix C will be used to illustrate that the system can successfully carry out the tasks of configuring production routes under changing dynamic conditions so that it can form the basis of a full-scale activity scheduling system which will produce feasible schedules and which can therefore be applied in a real environment.

### 9.5.1.Plant Structure and Routing Representation

Appendix C (9.1.) shows part of the Control Module database for the production routing concerned with the separation of milk for the evaporator and cottage-cheese production departments in the Minsterley plant.

In the plant item representations the number of slots varies according to the information which is considered important for decision making purposes. However, all plant items have the three slots which enable the current availability of a plant item with respect to the **CAN-CONNECT-TO** relation

to be derived. These are **configuration**, **current_outputs**, and **current_inputs**. It can be seen that the varied configurations of the plant items in the Minsterley plant can all be represented through the AND/ OR data structure described in Chapter 7. In addition the vessel plant item representations also have a **contents_status** slot which will contain an indication of whether the item is physically available for filling and emptying or just one of these depending on which of the finite capacity limits or threshold trigger limits have currently been reached. It can be seen that there is flexibility in the numbers of plant items that can be defined for a route position. Also there a number of plant items which are common to a number of different routes so the correct representation of the connectivity of the plant items themselves is vital if they are to be correctly allocated to routes during the execution of the BPS.

### 9.5.2.Route Configuration and Co-ordination of Module Execution

Appendix C (9.1.) shows the status of the plant at a point when there are no routes configured in it. Appendix C (9.2.) shows a single part-cycle of the Control Module through the 'C' phase represented by it's screen output starting from this initial status. The following section will describe the screen output at specific points marked as (Point N) to show the route configuration procedure and the synchronisation of data transfer between the two modules.

(Point 1). The Control Module receives protocol -1 from the Simulation Module so that it becomes the active module. On becoming active the top-level procedure **dialogue8** calls the procedure **strategy** as described earlier which adds the goal strategy with status "active" into working memory and the Control Module attempts to apply strategy rules using the forward chaining inference engine from FOOPS.

(Point 2). The first rule **st1** fires adding the goal **immediate_requirements** with status "active" to working memory.

(Point 3). This goal is required as a context for the rule **imm1:** to fire, which calls a procedure to carry out a Total Float (TF) calculation for all batches which are currently in the working memory. The procedure updates their urgency status to critical if they have a TF of zero or it is negative. The procedure also updates the temporal activity status of each critical batch.

(Point3 - Point 4). On this part of the trace the updating of the cottage-cheese batch **cc01** occurs through the actions of rule **imm1**. Its urgency slot is set to **critical**, and its temporal activity status is returned as **[[sep_milk_cc_silos/ can_start, past_skim_cc_vats/ cannot_start]]**.

(Point 5 - Point 6). The actions of **imm1** are completed with the addition of goals as contexts to focus the direction of the Control Module in configuring production routes for processing batches of product. The specific details of route configuration on the trace are described by comments in the format /* comment */. Initially the focus is on configuring routes for unloading milk (batches **um01**, **um02**, and **um03**) but the calls to the procedure **configure_activity** made by the rule **inpta** for these batches fail to find any route which can be configured because the milk reception area input queue **RECQ** and unloading bays contain no transport vehicles. Thus the rule **inpt3** fires which refocusses the configuration process on activities for critical batches in the factory through the goal **factory_product** with status active.

(Point 7.). The strategy rule **fact1** is put into the working memory conflict set. Its purpose is to trigger the configuration of routes for supplying skimmed milk for internal factory production requirements. The actions of this rule contain several calls to a procedure **configure_activities**, which makes a call to the main configuration procedure **configure_activity** for each critical batch of a particular product type.

(Point 8.) The first call to **configure_activity** is for the activity/ batch combination **sep_milk_ev_silos/ ev02**. Configuration of a process route for this activity now proceeds as described in the example of the dynamic route configuration in Chapter 7. The procedure first finds that the route made up from the set of subroutes **E1ST**, **E1PR**, **E1SS**, **E1SI**, **E1LN**, and **E1SL** is

available and carries out initial unary and binary constraint checking on the route positions.

(Point 9 - Point 10). Having determined that the route is available the Control Module now makes a second call to the FOOPS forward chaining inference engine for the purpose of considering preferential element selection rules. A number of rules concerned with the selection of plant items for positions in this route fire until the configuration of the route is complete with a single plant item in each route position. The key point about all the rules that fire to select a plant item for a route position is that the procedure call **select_element** instigates binary constraint checking to occur to ensure that the route is always consistent with respect to the remaining route choices.

(Point 10 - Point 11). The configuration of the route is now complete, and the sub-routes making it up are added to the current process sequence to be passed to the simulation. This process triggers a daemon **sequence_update** to fire each time a sub-route is added which updates the current connections of the plant items in that sub-route.

(Point 11. - Point 12). Routes for processing a second batch of evaporator skimmed milk and cottage cheese skimmed milk are also configured. The Control Module keeps moving between the strategy level and the configuration level during this process until after the last call to configure_activity, the rule fact1 completes its actions and the strategy level inference procedure completes.

(Point 13). The top-level procedure of the Control Module moves forward to the **send_link** procedure to update the waiting Simulation Module with the current process and CIP sequence.

(Point 14). The procedure **send_current_sequence** is called and it can be seen that the current sequence to be sent consists of a set of configured sub-routes with the appropriate format for the Simulation Module.

(Point 15 - Point 16). The first call to the data transfer protocol number 4 is made to send the first line of the process sequence to the Simulation Module. Calls continue to be made until there are no more lines to be sent and the

**current_sequence** slot is empty. There is now a call to the procedure **send_CIP_sequence.**

(Point 17). All sequence data has now been sent across to the Simulation Module, and the Control Module enters a waiting phase causing the Simulation Module to become active. The format of a sequence which has been received by the simulation can be seen in Photo.9.1. which shows sub-routes associated with the routes configured by the Control Module and the batchsizes which have been set on them. At this point the simulation enters its part of the 'C' phase and sets the plant items up in the routes as specified by the sequence data. As each plant item is set up in a route position its status is updated within the simulation and the required update data is sent back to the Control Module via an appropriate data transfer protocol.

(Point 18). An example of a protocol to update the status of a plant item in the Control Module is shown here. Protocol 5 updates the status of the cottage cheese silo (**CCS2**) with its current net rate of 0.301 klitres $min^{-1}$ and activity state 7 "filling".

(Point 19). Protocol 5 is also called here to update the status of Milk silo 3 (**MSL3**) the source for the route. It can be seen that its net rate is set to -0.334 klitres $min^{-1}$ and it's activity state to 8 "emptying". Apart from being negative because it is is an output rather than an input rate, the value of this rate is different from that set on **CCS2** because the rate on **CCS2** was adjusted by the yield factor of 0.9 for the route to represent the separation of milk into skimmed milk and cream.

(Point 20). At this point updating of Milk silo 1 (**MSL1**) occurs due to its inclusion as the source of route 2 for separation of milk to the evaporator silos. It's net rate is set to -0.334 klitres $min^{-1}$ which is the process rate of the route.

(Point 21). At this point updating of **MSL1** occurs again because it has also been included as the source for route 1 for separation of milk down to the evaporator silos. This makes it a shared resource and it can be seen that its net rate has now altered to -0.668 klitres $min^{-1}$ the combined process rate of both routes.

(Point 22). Updating of plant item statuses from the simulation model finishes. The status of the process routes and individual plant items resulting from this part of the Control Module execution cycle is shown in Appendix C (9.3.) for route 1 to the evaporator skimmed milk storage silos. The batch for which this route was configured was ev02, and it can be seen that the status of the activity **sep_milk_ev_silos** has been set to **in_progress**. Route 1 consisted of the set of sub-routes **E1ST**, **E1PR**, **E1SS**, **E1SI**, **E1LN**, and **E1SL**, and these sub-routes were configured with the plant items shown in the **current_input**, **current_processor**, and **current_output** slots to give the route:

**MSL1->RLN1->SEP1->SPS1->SIP1->SKLN->ELN1->EVS2**

In looking at the status of some of the plant items in this route it can be seen that they have been updated in terms of their **current_connections** to form the current status of the plant network. For example **MSL1** which is the source for this route and the other evaporator supply route shows **RLN1** as its **current_output** connection which actually appears twice as a slot value. Likewise **RLN1** shows **MSL1** twice as an input connection. This is because the connection has been made in two routes. As a route shuts down after completing some or all of its set batch size the resources used by the route must be "released" back to the Control Module. If a connection has been made in more than one route then this representation allows one instance of the connection to be undone by simply removing one item from the current connection slot of each of the participating plant items leaving any other instances of the connection still intact. **RLN1** shows two **current_outputs** to **SEP1** and **SEP2** which is in line with it's **AND** output configuration.

This route configuration can be seen in the Simulation Module in Photo 9.2. The screen shows a simple representation of the process flow in the Minsterley plant on the left hand side in which the configuration of the main

plant items can be seen, and the routing associated with the plant on the right. The sub-routes making up route 1 to the evaporator storage silos can be seen on the far right of the screen. They have been configured with the plant items listed above. The representation also shows **LKxx** elements which link the sub-routes together into the route, and the placement of the separator in a sub-route (**E1CM**) to represent its cream output as well as the skimmed milk output of the main part of the route.

The simulation is still active and moves forward into the 'A' phase of the three phase cycle and moves the system time forward up to the time of the first scheduled event when the Control Module is updated again and becomes the active component in the BPS execution again. Photos 9.3. shows the progression of the simulation forward from this point and some resulting routes which are configured primarily on the basis of arriving milk vehicles for unloading. When the simulation has moved forward in time to 119 minutes a system event occurs when the evaporator skim silo **EVS2** hits high level. This causes the re-scheduling of all events concerned with affected routes and the immediate shutdown of the two evaporator supply routes which are re-configured as in Photo 9.4. with Evaporator silo 1 (**EVS1**) as the sink for both routes. When the simulation has moved forward in time again to time = 271 the required batchsizes for the two evaporator skim supply routes are completed and these routes shutdown. In the resulting Control Module execution, the activity **evap_skim** passes the precedence constraint checks for both the skimmed milk concentrate batches and two routes for these batches are configured as in Photo 9.5. The routes are set up with different Evaporator skim silos as inputs because the silo configurations only allow them to make an output connection to one evaporator. The system then moves forward again continuing with the three phase execution cycle until the simulation is halted at time = 303 after the cottage cheese supply route has been re-configured because **CCS2** hit maximum contents. Appendix C (9.4.) shows the output from the simulation up to this time as recorded in the schedule arrays when

routes were set up and shutdown. A partially developed schedule can be displayed by the Simulation Module as shown in Photo 9.6. For each sub-route that has run or is running in the model it shows the plant items that it was configured with, and information about the amount of product processed and moved through storage vessels. This output contains the relevant information to act as a schedule to be used in a plant for setting up production routes and is feasible to be implemented because no physical constraints on plant item availability have been broken in deriving it. The information from the Control Module database concerning the status of product batches and process routes with respect to product could be used to provide the scheduling information about the product that would be required in a full scale implementation of the BPS.

## 9.6.USE OF THE BPS IN A REAL ENVIRONMENT

There are a number of issues which would have to be addressed to implement the BPS in a real environment, for example the validation of a model, the initialisation of a model with real plant data for scheduling, and the control of the schedule on-line.

### 9.6.1.Model Validation

The work carried out during this project on validation has been concerned with whether the logic of the model is correct and whether the data used to represent plant items and routing is correct so that the output from the BPS would be feasible. This has been assessed by analysing trace output data from both the simulation and the Control Module to see whether physical constraints on plant item availability are broken during the configuration process and whether the two modules interact together correctly. This should be distinguished from experimentation with the control rules to attempt to produce better schedules than are currently being developed by some existing

scheduling system. This would be an iterative process involving two levels of assessment; how good are the route configurations developed by the BPS, and how good are the overall schedules developed by it. On the basis of the results from a particular run under a particular set of conditions decisions would have to be made about whether the control rules should be changed or whether working practise in the plant should be changed.

### 9.6.2.Model Initialisation

Initialisation of a model is currently carried out manually by setting up the status of the plant items, routes, product batches and production requirements in the respective system data files. However, it is certainly desirable that the initialisation of a regularly used scheduling system should be as automated as possible with links to a higher level planning system for initialisation of production requirements and automation of the initial plant status. The initialisation of the plant status is more of a difficult issue than linking into the production planning system. Most literature on using simulation based systems for scheduling assumes that the scheduler will be run at the time that the schedule is required to start. In this case with direct links to the plant control systems, for example where the simulation is used for emulation purposes as described by Harmonosky [56], its status always reflects that of the real plant. Alternatively, initialisation could be based on simply "dumping" the real data to the simulation when required as described by Nurse and Chrystall [154]. However, although on-line data capture gives the assurance that the inital status of the simulation will be accurate, it might be desirable to carry out the scheduling function some time in advance of the start of the actual production period. How the model should be initialised in these circumstances is a matter which has not really been satisfactorily resolved in the literature. The model could be initialised by using the final simulation status from the last scheduling run, although this would require a significant assumption that this schedule is still currently being implemented

and is still valid. Udo Graefe et al. [155] describe the use of a simulation for validating schedules produced by a separate scheduling module, which is initialised using its last saved state, plus the new schedule to test, and a bill of operations. Using some approach based on the results of a simulation run the initialisation can still be automated by integration with a higher level planning system for production requirements and due dates, but it must be recognised that the accuracy of the initial status of the plant will become less sure the further in advance the scheduling is done.

### 9.6.3.Schedule Monitoring and Dynamic Rescheduling

A simulation derived schedule must be used in conjunction with schedule monitoring to ensure that the schedule developed by the simulation is carried out, and to determine what course of action to take when the schedule and reality deviate too far from each other. Bhattacharyya, Roy and Huang [6] propose the use of a simulation based scheduler to initially develop a schedule for a plant off- line, and then to constantly monitor it and make dynamic rescheduling decisions on the basis of the current status. The aim of this dynamic rescheduling is to return back as close as possible to the original plan from any disturbances which occur during its execution. They describe the role of the on- line scheduler in terms of three functions; to assess the extent of any deviation which may have occurred from the original plan; to devise an action plan to get the schedule back to its original schedule if possible; and to modify the original plan if recovery is not possible. Where only minor modifications are required to the original plan, the simulation may be used to update the timing of the plan on- line. If rescheduling of jobs is required, then the simulation is run off- line again, but in this case one of the objectives is to keep changes from the original plan to a minimum, so rescheduling is based on the original work- to- list as far as possible. In order for a system to function in this way it requires rules to assess the extent of any deviation and what course of action to take, knowledge about these different courses of

action, and how the flexibility and responsiveness of the system determines the constraints within which modifications can be made to the original plan.

## 9.7.DISCUSSION

The knowledge representation scheme as implemented as a frame-based system in PROLOG can successfully capture the nature of the connectivity constraints of the configuration of a variety of plant items. It can represent production routing which varies both in length and the numbers of plant items which can be choices for route position without arbitrary restrictions caused by the representation structure.

It can be seen from the trace of the BPS execution and the status of plant items and routes over time that it can successfully configure production routes through a complex batch process plant taking into account the dynamic connectivity constraints that exist on plant item availability. The Simulation Module is able to co-ordinate correctly the activities of variable numbers of plant items linked together in production routes and correctly maintain the material balance of the plant through rate adjustments via input factors and output yields. The communications link between the two modules has enabled the model execution to be set up as a three phase structure such that decision making is based on a full update of plant status and the two modules are fully integrated into a hybrid system. Using any other approach than the three phase structure would be inappropriate in the case of a batch plant model because it is so inter-linked. To make any meaningful control decisions the status of all entities at a particular time in the routes which are setting up or dismantling must be updated completely to correctly take account of plant item connections and plant items which are being shared across routes.

There are some drawbacks to the hybrid approach which have become apparent or been confirmed from this system implementation. Each module

has to have access to appropriate data structures which necessitates some duplication of data. In order to maintain consistency this data should originally come from a single source. However, in the current implementation, for historical development reasons each module's data is set up from a separate set of files. Setting up the communications link between the two modules of the system also confirmed one of the other drawbacks cited for hybrid systems; namely making data types compatible between languages. For example in the version of PROLOG that was used the Real data is held only in double precision format of eight bytes in length, whereas the SEE-WHY Real data is in single precision format of four bytes in length. In order to make Real data transfer possible between the two modules the Assembly routines for the PROLOG communications interface contain code to convert an incoming four byte single precision Real number into an eight byte double precision number and code to convert an outgoing eight byte double precision number into a four byte single precision Real number. It was thought better to have to transfer only four bytes at any time rather than eight to minimise the risk of any data corruption during the transfer. There have been no problems with the operation of the link on this basis. In Flitmans thesis [123] he commented that another drawback was trying to duplicate the list structure of PROLOG in FORTRAN because it could contain varied data types and additional data structures which could not be duplicated in FORTRAN. In the BPS the data transfer is structured so that the receiving module always knows what data type is coming to it next through the use of the protocols for different data transfer requirements so there is no need to try and duplicate complex data structures from one module to another. The PROLOG module is primarily in use because it allows the construction of complex data structures which cannot easily be achieved in FORTRAN and the system is set up so that there should be no requirement to attempt to duplicate them in the FORTRAN module.

The three phase structure provides a natural boundary between the two modules and makes amending the BPS and the way that the two modules are co-ordinated relatively easy. The Control Module can also be operated for a given plant status completely independently of the Simulation Module to configure a set of production routes. Therefore it would appear very feasible that it could act as real-time batch management system in its own right.

One of the charges levelled against rule-based scheduling systems is the difficulty in covering all situations that could be encountered. An aim of developing the BPS has been to account for gaps in a rule-base through the structure of the model which has been achieved through the knowledge representation scheme and constraint reasoning processes developed.

# CHAPTER 10 CONCLUSIONS AND FURTHER WORK

The work in this thesis has been concerned with the development of a generic Batch Process Scheduler (BPS) to produce good feasible short term schedules that can be implemented for production control in a real batch process plant. This will ensure that the batch management and process control levels in a Production Planning and Control hierarchy can function correctly. The specific issues that were addressed in relation to the objectives for the work were:

1. The representation of a batch plant network, and the dynamic connectivity constraints within it.
2. The configuration of routes subject to these constraints and preferential considerations.
3. The scheduling of activities to meet production goals.
4. Accounting for the constraints imposed by the finite capacity of the plant.
5. The development of a hybrid structure to implement the BPS.

## 10.1 CONCLUSIONS

The first issue has been addressed through the development of a new representation scheme for plant items using an AND/ OR structure. This representation enables the feasible connections of plant items to be modelled at an appropriate level of detail for the scheduling of operations without resorting to modelling every valve and process control device which would make the representation too cumbersome for this purpose. The AND/ OR structure is a natural way to view the feasible connections of plant items, and its modularity makes it suitable for a generic tool because it makes updating and maintenance of a plant network representation easy. The development of a procedure for determining whether two plant items can make a connection at any time based on the AND/ OR representation and their current plant connections has effectively dealt with problem of dynamic connectivity. The

development of an accurate model of the constraints on a plant network simply relies on defining the feasible connections for each plant item. This gives considerable advantages over representing these constraints as rules which refer to particular states of the network to determine whether connections are currently feasible. The AND/ OR representation is much more robust than this approach because the potential constraints on connections are directly represented through the data, so the BPS will not suffer from problems such as gaps in the rule-base or contradictory rules.

It has been recognised that the allocation of plant items to production activities involving routing can be viewed as a configuration problem in which resources are limited and their availability changes dynamically. This has been addressed by a new representation scheme for routes as simple networks and the development of a procedure to make any route consistent with respect to the plant items which are currently available as choices for the positions in it. This enables the feasibility of configuring a route to be determined at any point in the BPS execution. In addition it guarantees that a route can be configured if it is made initially consistent and still has at least one plant item choice remaining in each position. Thus a feasible configuration can be derived by a simple procedure of selecting a plant item for each position in any order, and propagating the effects through the rest of the network before making the next selection. By incorporating this procedure with a rule-based approach, to take preferential considerations into account and impose a decision order to implicitly take account of the constraining effects of making a choice, the best feasible allocation of resources to a production activity can be derived. In addition, because a route can be configured with a simple procedure as described above it enables a default general set of rules to be employed. Thus the BPS can have gaps in the configuration rule-base for specific activities and it will still be able to make feasible allocations of plant items to production routes. This makes the BPS more robust than a purely rule-based system and enables a configuration rule-base to be built up incrementally.

The activity scheduling of a plant to meet production goals has been addressed through the development of a framework incorporating control structures, a suitable product representation based on a production recipe, procedures to reason about constraints on activities, and scheduling rules. The product representation developed gives an advantage over other schemes in which batches are discrete entities because it is more natural and flexible, for example, allowing the representation of generic intermediates feeding a number of different processes. Although this part of the system development needs more work in the area of the rules used for scheduling, the incorporation of the dynamic route configuration procedures ensures that even using a simple and incomplete rule-base the BPS will still produce feasible schedules. Thus the scheduling rule-base can be incrementally developed to produce better schedules.

The constraints imposed by the finite capacity of a plant have been addressed by the development of a dynamic simulation model with procedures to co-ordinate the activities of chains of plant items linked together in routes, and account for finite capacity limits in intermediate storage vessels being reached. The model can cope with the use of shared resources across routes and correctly maintain the material balance of system. Used in conjunction with the Control Module to dynamically configure production routes its output represents a feasible and realistic allocation of resources to production activities over time which can be used to control a plant and ensures that the batch management level of the Production Planning and Control hierarchy can function properly.

The issue of a suitable hybrid structure for the implementation of the BPS has been addressed through the development of a three phase structure for execution. This provides a natural split between the two modules, and enables the Control Module to operate on the basis of a global update of plant status

from the Simulation Module. The integration of the two modules has been achieved through the development of a set of flexible data transfer procedures which mean the system can be incrementally developed as required without resorting to low level assembler programming. The structure of the hybrid BPS means that the Control Module could be disconnected from the Simulation Module with the potential to act as an on-line scheduler and batch management system.

## 10.2 FURTHER WORK

There are some issues still to be addressed to meet fully the objectives of the research, and some issues which have arisen as a result.

1. The activity scheduling rule-base requires considerable development in order for the system to produce good schedules. It has been developed in its current form principally to test the production activity configuration procedures and drive the BPS through its exectution cycle to test the hybrid structure of the system.

2. There are some aspects of the BPS as it is currently developed which need to be expanded, for example the ability to handle routes with branches as well as serial routes. This expansion does not involve any major changes to the principles of operation of the system but is required to make it fully applicable as a scheduling tool in a real environment.

3. In order for the BPS to be implemented as a real time scheduler and batch management tool, a number of other issues would need to be addressed, such as the interface of the system to a process control system, and how it should operate in this mode. As discussed by Bhattacharrya, Roy and Huang [6], if it was managing a plant on the basis of a schedule developed off-line, it would need to be able to assess

the extent of any deviations from the schedule and how to react to them. For example, if the deviation was small, then an approach such as the POMA algorithm, developed by Cott and Machietto [5], to adjust the start times of operations in the plant might be sufficient. In the case of a larger deviations such as the breakdown of a plant item or delays in raw material supplies, part or all of schedule might need to be redeveloped using the BPS in its off-line scheduling mode.

REFERENCES

1.T.Simmons, What Makes 'Batch' Work?, p53- 57, Process Engineering, April, 1989.

2.S.F.Bolander, Manufacturing Planning and Control in Process Industries, APICS publications, 1981.

3.B.R.Benesman, Production Planning in the New Zealand Dairy Industry, p747-754, Jnl. Op. Res. Soc., v37, n8, 1986.

4.A.Mauderli, D.Rippin, Production Planning and Scheduling for Multipurpose Batch Chemical Plants, p199-206, Comp. Chem. Eng., v3, 1979.

5.B.J.Cott, S.Machietto, An Integrated Approach to Computer- Aided Operation of Batch Chemical Plants, p1263-1271, Computers Chem. Eng.,v13, n11/12, 1989.

6.S.K.Bhattacharyya, R.Roy, Y.S.Huang, Integrated Intelligent Simulation Environment for Manufacturing Scheduling, p157-167, Proc. Intl. AMSE Conf. "Modelling and Simulation", New Orleans, v3, AMSE Press, 1991.

7.M.A.H.Dempster, M.L.Fisher, L.Jansen, B.J.Lageweg, J.K.Lenstra, A.H.G.Rinnooy Kan, Analytical Evaluation of Hierarchical Planning Systems, p.707-716, Operations Research, v29, n4, 1981.

8.R.Roy, Scheduling and Capacity Management Using Simulation, in Logistics Technology International, 1992, Sterling Publications.

9.G.V.Reklaitis, Review of Scheduling of Process Operations, p119-133, AIChE Symposium Series, n214, v78, 1982.

10.D.W.T.Rippin, Design and Operation of Multiproduct and Multipurpose Batch Chemical Plants- An Analysis of problem Structure, p463-481, Computers and Chemical Engineering, v7, n4, 1983.

11.H.Ku, D.Rajagopalan, I.Karimi, Scheduling in Batch Processes, p35-45, Chemical Engineering Progress, v83, pt8, 1987.

12.S.French, Sequencing and Scheduling An Introduction to the Mathematics of the Job-Shop, Ellis Horwood, 1982.

13.R.F.H.Musier and L.B.Evans, An Approximate Method for the Production Scheduling of Industrial Batch Processes- With Parallel Units, p229-238, Computers and Chemical Engineering, v13, n1/2, 1989.

14.D.B.Birewar and I.E.Grossmann, Efficient Optimisation Algorithms for Zero-Wait Scheduling of Multiproduct Batch Plants, p1333-1345, Industrial Engineering Chemistry Research, v28, n9, 1989.

15.S.H.Rich and G.J.Prokapakis, Scheduling and Sequencing of Batch Operations in a Multipurpose Plant, p979-988, Industrial Engineering Chemistry Process Design and Development, v25, n4, 1986.

16.E.Kondili, C.C.Pantelides, and R.W.H.Sargent, A General Algorithm for Scheduling Batch Operations, p62-75, Third International Synposium on Process Systems Engineering PSE 88, 1988.

17.S.Machietto, Automation Research on a Food processing Pilot Plant, IChemE Symposium Series No 126.

18.K.Kuriyan, and G.V.Reklaitis, Approximate Algorithms for Network Flowshops, p79-90, IChE Symposium Series, v92, 1985.

19.K.Kuriyan, and G.V.Reklaitis, Scheduling Network Flowshops so as to Minimise Makespan, p187-200, Computers in Chemical Engineering, v13, n1/2, 1989.

20.D.Rajagopalan, and I.Karimi, Scheduling in Serial Mixed-Storage Multiproduct Processes with Transfer and Setup Times, p679- 686, International Conference on Foundations of Computer- Aided Process Operation, FOCAPO 87, 1987.

21.D.Rajagopalan, and I.Karimi, Completion Times in Serial Mixed-Storage Multiproduct processes with Transfer and Setup Times, p175-186, Computers and Chemical Engineering, v13, n1/2, 1989.

22.S.Hasebe, I.Hashimoto, T.Takamatsu, Scheduling through Reordering Operations, p76-81, Third International Synposium on Process Systems Engineering PSE 88, 1988.

23.D.P.Daugherty, R.M.Felder, An Expert System for Scheduling in a Multipurpose Speciality Chemicals Plant, p44-49, Plant/ Operations Progress, v9, n1, 1990.

24.F.C.Knopf, Sequencing of a Generalised Two-stage Flowshop with Finite Intermediate Storage, p207-221, Computers and Chemical Engineering, v9, n3, 1985.

25.H.Ku and I.Karimi, Scheduling in Serial Multiproduct Batch Processes with Due-Date Penalties, p580-590, Industrial Engineering Chemistry Research, v29, n4, 1990.

26.R.M.Felder, P.M.Kester and R.F.Moldin, An algorithm for Scheduling Production in a Multipurpose Speciality Chemicals Plant, AIChE Meeting San Francisco, Paper n123A, p1-29, 1984.

27.U.M.Egli and D.W.T.Rippin, Short-term Scheduling for Multiproduct Batch Chemical Plants, p303-325, Computers and Chemical Engineering, v10, n4, 1986.

28.W.Wiede Jr., An Interactive Scheduling System for the Operation of Multiproduct Plants, Ph.D. Dissertation, Purdue University, 1984.

29.R.F.H.Musier and L.B.Evans, Batch Process Management, p66-77, Chemical Engineering Progress, v86, pt6, 1990.

30.J.M.Neville, R.Ventker, T.E.Baker, PROSIT- An Interactive Process Scheduling System, p134-144, AIChE, n214, v78, 1982.

31.R.Beadle, Computerised Scheduling: A Practical Guide, CIM 90 Conference, NEC, Birmingham, 1990.

32.C.D.Pegden, Simulation and Scheduling (Panel), Proceedings 1991 Winter Simulation Conference, p388- 389, SCS, 1991.

33.R.M.Davies, R.O'Keefe, Simulation Modelling with Pascal, Prentice Hall International (UK) Ltd., 1989.

34.A.M.Law, D.W.Kelton, Simulation Modelling and Analysis, 2nd Ed., McGraw- Hill Inc., 1991.

35.R.M.O'Keefe, J.W.Roach, Artificial Intelligence Approaches to Simulation, p713- 722, Jnl. Opl. Res. Soc., v38, n8, 1987.

36.M.Pidd, Computer Simulation in Management Science, 2nd Ed, John Wiley and Sons, 1988.

37.R.M.O'Keefe, J.Haddock, Data Driven Generic Simulators for Flexible Manufacturing Systems, p1795- 1810, Int. Jnl. Prod. Res., v29, n9, 1991.

38.A.Bollino, Study and Realisation of a Manufacturing Scheduler using FACTOR, p9- 20, Proc. 4th. Intl. Conf. Simulation in Manufacturing, IFS, 1988.

39.N.E.Larsen, L.Alting, Requirements to Scheduling Simulation Systems, p231- 236, Proc. 1990 Summer Computer Simulation Conf., SCS, 1990.

40.T.A.Barnes, I.L.Gardner, A Simulation Model for Production Scheduling of a Facility in a Process Industry, p609- 613, Proc 1990 Winter Simulation Conf., SCS, 1990.

41.C.Moreira da Silva, J.M.Bastos, The Use of Decision Mechanisms in Visual Simulation for Manufacturing Systems Modelling, p165- 170, AI Applied to Simulation, v18, n1, SCS, 1986.

42.S.K.Bhattacharyya, R.Roy, Y.S.Huang, Knowledge Based Simulation Techniques for Control of FMS, Proc. Beijing Int. Conf. in Simulation and Scientific Computing, 1989.

43.F.P.Wyman, Common Features of Simulation Based Scheduling, p341-347,Proceedings 1991 Winter Simulation Conference, SCS, 1991.

44.D.T.Sturrock, H.B.Higley, Scheduling with Simulation: A Case Example, World Productivity Forum and 1987 International Industrial Engineering Conf. Proc.

45.H.Grant, Production Scheduling using Simulation Technology, p129- 137, Proceedings 2nd International Conference on Simulation in Manufacturing, IFS, 1986.

46.P.D.Spooner, A Simulation Based Interactive Production Control System, p65- 73, Proc. 1st. Intl. Conf. Simulation in Manufacturing, IFS, 1985.

47.F.Grant, R.G.Lagoni, The User's Role in a Simulation Based Scheduling System, p936- 941, Proc. 1989 Winter Simulation Conf., SCS, 1989.

48.Sun Qi- Zhi, Expert Simulation for On- line Control, p172- 180, Computer Integrated Manufacturing Systems, v2, n3, 1989.

49.R.Roy, P.A.McCarthy, A.D.Klapatyj, A DES Scheduling Tool for Shop Floor Control, p449-453, Proc. 6th Intl. Manuf. Conf with China, IMCC, Hong Kong, v2, 1993.

50.R.D.Hurrion, R.J.R.Secker, Visual Interactive Simulation as an Aid to Decision Making, p419- 426, OMEGA, Int. Jnl. Mgmt. Sci., v6, n5, 1978.

51.R.J.R.Secker, That VIS Offers a Viable Technique for Examining Production Planning and Scheduling Problems, M.Sc. Thesis, University of Warwick, 1977.

52.M.W.Fisher, The Application of Visual Interactive Simulation in the Management of Continuous Process Chemical Plants, Ph.D. Thesis, University of Warwick, 1982.

53.D.T.Sturrock, H.B.Higley, The Use of Simulation for Gross Planning, Scheduling, Standards, and Tracking, p411- 415, Computers and Industrial Engineering, v11, pt1- 4, 1986.

54.J.G.Crookes, Generators, Generic Models and Methodology, p765- 768, Jnl. Opl. Res. Soc, v38, n8, 1987.

55.S.R.Hill, M.A.M.Rogers, Practical Experience Contrasting Conventional Modelling and Data Driven Visual Interactive Simulation Techniques, p207- 220, Proceedings 2nd International Conference on Simulation in Manufacturing, IFS, 1986.

56.C.M.Harmonosky, Implementation Issues Using Simulation for Real- Time Scheduling, Control and Monitoring, p595- 598, Proc. 1990 Winter Simulation Conf, SCS, 1990.

57.F.A.Rodammer, K.P.White, A Recent Survey of Production Scheduling, p841- 851, IEEE Trans. on Sys. Man and Cyb., v18, n5, 1988.

58.M.S.Fox, AI and Expert System Myths, Legends, and facts, p8-20, IEEE Expert, Feb., 1990.

59.P.A.Sachs, A.M.Paterson, M.H.M.Turner, Escort- an expert system for complex operations in real time, p22- 29, Expert Systems, v1., n3., 1986.

60.J.Efstathiou, Expert Systems in Process Control, Longman, 1989.

61.J.J.Kanet, H.H.Adelsburger, Expert Systems in Production Scheduling, p51- 59, Eur. Jnl. Opn. Res., v29, 1987.

62.A.Kusiak, M.Chen, Expert Systems for Planning and Scheduling Manufacturing Systems, p113- 130, Eur. Jnl. Opn. Res., v34, 1988.

63.D.Ben- Ariah, Knowledge Based Control System for Automated Production and Assembly, p347- 368, Modelling and Design of Flexible Manufacturing Systems, Elsevier, 1986.

64.G.Bruno, A.Elia, P.Laface, A Rule- Based System to Schedule Production, p32- 39, IEEE Computer, 19, 1986.

65.C.Copas, J.Browne, A Rules- Based Scheduling System for Flow Type Assembly, p981- 1005, Int. Jnl. Prod. Res., v28, n5, 1990.

66.K.Kempf, C.LePape, S.F.Smith, B.R.Fox, Issues in the Design of AI- Based Schedulers: A Workshop Report, p37- 46, AI Magazine, 1991.

67.M.S.Fox, Constraint Directed Search: A Case Study of Job- Shop Scheduling, Pitman Publishing, 1987.

68.S.F.Smith, P.S.Ow, J.Potvin, N. Muscettola, D.C.Matthys, An Integrated Framework for Generating and Revising Factory Schedules, p539- 552, Jnl. Opn. Res. Soc., 1990.

69.C.LePape, SOJA: A Daily Workshop Scheduling System, p195- 211, Expert Systems '85, University of Warwick, Dec 17- 19, 1985.

70.U.Canzi, G.Guida, W.Poloni, S.Pozzi, CHRONOS II: A Knowledge- based Scheduler for Complex Manufacturing Environments, p76- 83, IEEE 2nd Int. Conf. on Data and Knowledge Systems for Manufacturing and Engineering, IEEE, 1989.

71.C.Badie, G.Bel, E.Bensana, G.Verfaillie, Operations Research and Artificial Intelligence Cooperation to Solve Scheduling Problems: the OPAL and OSCAR systems, p1-5, IEE Ist Int. Conf. Expert Planning Systems, 27- 29th June 1990, IEE, 1990.

72.W.S.Steffen, T.J.Greene, A Prototype System for Scheduling Parallel Processors Using Artificial Intelligence Methods, p156- 164, 1986 Annual International Industrial Engineering Conference Proceedings, 1986.

73.P.Burke, P.Prosser, A Distributed Asysnchronous System for Predictive and Reactive Scheduling, Technical Report AISL- 42, October 1989, University of Strathclyde, Department of Computer Science.

74.P.Elleby, H.E.Fargher, T.R.Addis, A Constraint- Based Scheduling System for VLSI Wafer Fabrication, Knowledge Systems Group, Department of Computer Science, University of Reading, 1988.

75.M.S.Fox, N.Sadeh, C.Baykan, Constrained Heuristic Search, p309- 315, IJCAI- 89, v1, 1989.

76.E.Rich, K.Knight, Artificial Intelligence, 2nd. Ed., McGraw Hill, 1991.

77.S.F.Smith, M.S.Fox, P.S.Ow, Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge- Based Factory Scheduling Systems, p45- 60, AI Magazine, v7, pt4, 1986.

78.M.Minsky, A Framework for Representing Knowledge, p95- 128, Mind Design, MIT Press, 1981.

79.A.Sathi, M.S.Fox, M.Greenberg, Representation of Activity Knowledge for Project Management, p531- 552, IEEE Transactions on Pattern Analysis and Machine Intelligence, v7, n5, 1985.

80.Y.V.Reddy, M.S.Fox, N.Husain, The Knowledge Based Simulation System, p27- 37, IEEE Software, v3, 1986.

81.L.S.Homem de mello, A.C.Sanderson, AND/ OR Graph Representation of Assembly Plans, p.188- 1999, IEEE Transactions on Robotics and Automation, v6, n2, 1990.

82.J.F.Allen, Maintaining Knowledge about Temporal Intervals, p832- 843, Communications of the ACM, v26, n11, 1983.

83.S.Parthasarathy, S.H.Kim, Temporal Reasoning for Manufacturing: A Hybrid Representation and Complexity Management, p67- 81, Robotics and Computer- Integrated manufacturing, v6, n1, 1989.

84.C.LePape, S.F.Smith, Management of Temporal Constraints for Factory Scheduling, p159- 170, Temporal Aspects in Information Systems, Elsevier, 1988.

85.P.J.Sackett, I.S.Fan, The Control of a Flexible Manufacturing System by Short Term Goal Identification, p123- 143, Int. Jnl. Adv. Manf. Tech., v4, 1988.

86.S.D.Wu, R.A.Wysk, An Inference Structure for the Control and Scheduling of Manufacturing Systems, p247- 262, Computers and Industrial Engineering, v18, n3, 1990.

87.A.K.Mackworth, Consistency in Networks of Relations, p99- 118, Artificial Intelligence, 8(1), 1977.

88.S.Mittal, F.Freyman, Towards a Generic Model of Configuration Tasks, p1395- 1401, IJCAI- 89, v2, 1989.

89.J.McDermott, R1: A Rule-Based Configurer of Computer Systems, p39-88, Artificial Intelligence, v19, 1982.

90.W.S.Havens, P.S.Rehfuss, Platypus: a Constraint- Based Reasoning System, p48- 53, IJCAI- 89, v1, 1989.

91.M.S.Crone, P.M.Julich, Dynamic Network Reconfiguration Using Constraint Propagation, p6- 11, AI and Simulation, Theory and Applications, v22, n3, SCS, 1990.

92.N.Sathi, M.S.Fox, R.Goyal, A.S.Kott, Resource Configuration and Allocation A Case Study of Constrained Heuristic Search, p26- 35, IEEE Expert, April 1992.

93.J.G.Vaucher, Views of Modelling: Comparing the Simulation and AI Approaches, p3-7, AI, Graphics and Simulation, SCS, 1985.

94.R.E.Shannon, R.Mayer, H.H.Adelsburger, Expert Systems and Simulation, p275-284, Simulation, v44:6, SCS, 1985.

95.R.O'Keefe, Simulation and Expert Systems- A Taxonomy and Some Examples, p10-16, Simulation, v46:1, SCS, 1986.

96.G.I.Doukidis, An Anthology on the Homology of Simulation with Artificial Intelligence, p701-712, Jnl. Opl. Res. Soc, v38, n8, 1987.

97.R.D.Hurrion, Intelligent Visual Interactive Modelling, p349-356, Eur. Jnl. Opl. Res., v54, 1991.

98.B.D.Marsh, T.M.Williams, G.L.Mathieson, The Use of Mixed Prolog/ Fortran for Battle Simulation, p311-318, Jnl. Opl. Res. Soc, 1990.

99.R.J.Paul, G.I.Doukidis, Further Developments in the Use of Artificial Intelligence which Formulate Simulation Problems, p787-810, Jnl. Opl. Res. Soc., v37, n8, 1986.

100.D.R.Ford, B.J.Schroer, An Expert Manufacturing Simulation System, p193-200, Simulation, v48:5, SCS, 1987.

101.J.Haddock, An Expert System Framework Based on a Simulation Generator, p45-53, Simulation, v48:2, SCS,1987.

102.S.Prakash, R.E.Shannon, Intelligent Back End of a Goal Directed Simulation Environment for Discrete- Part Manufacturing, p883-891, Proc. 1989 Winter Sim. Conf., SCS, 1989.

103.P.Khlar, W.S.Faught, Knowledge Based Simulation, p181-183, Proc. First Conf. AAAI, 1980.

104.A.Round, Chapter XXII: Knowledge Based Simulation, p417-518, The Handbook of Artificial Intelligence Vol. IV, A.Barr, P.Cohen, E.Fiegenbaum (Eds), 1989.

105.W.Vaessen, ProfiSEE: A Workbench for Visual Interactive Simulation Modelling in the Chemical- Pharmaceutical Industry, p239-249, OR Spektrum, v11, 1989.

106.J.Alasuvanto, E.Eloranta, M.Fuyuki, T.Kida, I.Inoues, Object Oriented Programming in Production Management- Two Pilot Systems, p765-776, Int. Jnl. Prod. Res., v26, n5, 1988.

107.T.Thomasma, Y.Mao, O.Ulgen, Defining Behaviour in a Hierarchical Object Oriented Simulation Program Generator, p266-271, Proc. 1990 Summer Comp. Sim. Conf., SCS, 1990.

108.P.Robertson, A Rule Based Expert Simulation Environment, p9-15, Intelligent Simulation Environments, v17, n1, Simulation Series, SCS, 1986.

109.J.Shivnan, J.Browne, AI Based Simulation of Advanced Manufacturing Systems, p23-33, Proc. 2nd Int. Conf. Sim. in Manf., IFS, 1986.

110.I.Futo, T.Gergely, TS-PROLOG A Logic Simulation language, p319-335, Transactions of the Society for Computer Simulation, v3, n4, SCS, 1987.

111.J.Cleary, K.Goh, B.Unger, Discrete Event Simulation in Prolog, p8-13, Artificial Intelligence, Graphics, and Simulation, SCS, 1985.

112.I.S.Fan, P.J.Sackett, A PROLOG Simulator for Interactive Flexible Manufacturing Systems Control, p239-247, Simulation, v50:6, SCS, 1988.

113.J.G.Vaucher, G.Lapalme, Process- Oriented Simulation in Prolog, p41-46, Simulation and Artificial Intelligence, v18, n3, Simulation Series, SCS, 1987.

114.C.R.Asfahl, A.Balagamwala, SIMLOG: A Prolog- based Simulator for Industrial Logic Control Systems, p195-199, Comp. Ind. Eng., v19, n1-4, 1990.

115.J.Haddock, R.M.O'Keefe, Using Artificial Intelligence to Facilitate Manufacturing Systems Simulation, p275-283, Comp. Ind. Eng., v18, n3, 1990.

116.S.Ruiz- Meyer, J.Talavage, A Hybrid Paradigm for Modelling of Complex Systems, p135-141, Simulation, v48:4, SCS, 1987.

117.J.Rothenburg, Knowledge- Based Simulation at the RAND Corporation, p133-161, Advances in Simulation 4, Knowledge- Based Simulation: Methodology and Application, Springer Verlag, 1989.

118.A.Ahmad, Towards a Knowledge- based Discrete Simulation Modelling Environment Using PROLOG, Ph.D. Thesis, University of Warwick, 1989.

119.R.E.Shannon, Knowledge based Simulation Techniques for Manufacturing, p953-973, Int. Jnl. Prod. Res., v26, n5, 1988.

120.A.Radiya, R.G.Sargent, ROBS: Rules and Objects Based Simulation, p241-256, Modelling and Simulation Methodology: Knowledge Systems' Paradigms, Elsevier, 1989.

121.W.Walker, K.Maughan, E.J.Fletcher, P.Smith, Knowledge Based Systems with Embedded Simulation Components, Proc. UK IT 90 Conf., IEE, 1990.

122.G.Nadoli, D.Castillo, J.E.Biegel, HRS: A Structure for Hierarchical Reasoning in Knowledge Based Simulation, p34-40, AI and Simulation, Theory and Applications, v22, n3, Simulation Series, SCS, 1990.

123.A.Flitman, Towards the Application of AI Techniques for Discrete Event Simulation, Ph.D. Thesis, University of Warwick, 1986.

124.G.C.Borchardt, STAR: A Computer Language For Hybrid AI Applications, p169-177, Coupling Symbolic and Numeric Computing in Expert Systems, Elsevier, 1986.

125.P.P.Lin, A.J.Yang, Data Communication Between an Expert System Shell and a Conventional Algorithmic Program with Application to Cam Motion Specification, p113-119, Engineering with Computers, v6, pt2, 1990.

126.P.V.Youle, Simulation of Full Scale Multi-stage Batchwise Chemical Plant, p150-157, Computer Journal, v3, 1960.

127.G.S.Joglekar and G.V.Reklaitis, A Simulator for Batch and Semi-Continuous Processes, p315-327, Computers and Chemical Engineering, v8, n6, 1984.

128.C.M.Ready, W.H.Simmonds, J.C.Taunton, A Knowledge Based Planning and Scheduling Toolkit for the Process Industries, p110-113, IEE 1st Int. Conf. Expert Planning Systems, IEE, 1990.

129.C.A.Roberts, T.G.Beaumariage, Y.Dessouky, M.Ogle, Object Oriented Tools Necessary for a Flexible Batch Process Management Architecture, p323-330, Proc. 1991 Winter Sim. Conf., SCS, 1991.

130.P.F.Roth, Discrete, Continuous and Combined Simulation, p25-29, Proc. 1987 Winter Sim. Conf., SCS, 1987.

131.C.D.Pegden, R.E.Shannon, R.P.Sadowski, Introduction to Simulation using SIMAN, McGraw Hill Inc., 1990.

132.R.M.Felder, Simulation- a tool for Optimising Batch-process production, Chemical Engineering, v90, pt8, p79-84, 1983.

133.R.M.Felder, G.B.McLeod, and R.F.Moldin, Simulation for the Capacity Planning of Speciality Chemicals Production, Chem. Eng. Prog., v81, pt6, p41-46, 1985.

134.C.Leggett, A Case Study of a batch manufacturing plant simulation, European Journal of Operational Research, v2, p1-7, 1978.

135.C.H.White, Application of Operations Research (OR) Methodology to Process Operations, p101- 138, International Conference on Foundations of Computer- Aided Process Operation (FOCAPO '87), July 1987, Park City, Elsevier- New York 1987.

136.K.Kuriyan, G.V.Reklaitis, and G.Joglekar, Multiproduct Plant Scheduling Studies using BOSS, Ind. Eng. Chem. Res., v26, pt8, p1551-1558, 1987.

137.R.A.Young, G.V.Reklaitis, Capacity Expansion Study of a Batch Production Line, Ind. Eng. Chem. Res., v28, pt6, p771-777, 1989.

138.R.C.Morris, Simulating Batch Processes, Chemical Engineering, v90, pt10, p77-81, 1983

139.R.D.Hurrion, An Interactive Simulation System for Industrial Management, p86-93, Eur. Jnl. Op. Res., v5, 1980.

140.M.Hofmeister, L.Halsz, D.W.T.Rippin, Knowledge- Based Tools for Batch Processing Systems, p82-87, Third Int. Sym. on Process Systems Engineering, 1988.

141.M.Walters, A.Terroux, D.Waye, Modelling Made easy: The Synthetic Intelligent (SI) Approach, p78-90, AI Papers 1988, v20, n1, SCS, 1988.

142.E.O'Shima, Sequencing and Scheduling of Plant Operations, p139-167, International Conference on Foundations of Computer- Aided Process Operation (FOCAPO '87), July 1987, Park City, Elsevier- New York 1987.

143.R.Lakshmanan, G.Stephanopoulos, Synthesis of Operating Procedures for Complete Chemical Plants- I. Hierarchical, Structured Modelling for Non-Linear Planning, p985-1002, Comput. Chem. Eng. v12, n9/10, 1988.

144.A.G.Hofmann, G.M.Stanley, L.B.Hawkinson, Object Oriented Models and their Application in Real- Time Expert Systems, p27-32, Simulation and AI, SCS, 1989.

145.D.H.Cherry, M.L.Preston, Batch Process Schedule Generation- ICI's BATCHMASTER, p91-100, ICHEME Symposium Series, n92, 1985.

146.M.Pidd, Simulating Automated Food Plants, p683-693, Jnl Op. Res.Soc, v38,n8, 1987.

147.J.A.Ward, Continuous Process Plant Scheduling Using Simulation, Ph.D. Thesis, University of Warwick, 1979.

148.I.Bratko, PROLOG Programming for Artificial Intelligence, 2nd Ed, Adison Wesley, 1990.

149.D.Merritt, Building Expert Systems in PROLOG, Springer-Verlag,1989.

150.R.J.Paul, Recent Developments in Simulation Modelling, p217-226, Jnl Op. Res. Soc., v42,n3, 1991.

151.R.Spinhelli de Carvalho, Cellular Simulation, p31-40, Opl. Res. Qrt., v27, n1, 1976.

152.S.E.Elmaraghby, Activity Networks: Project Planning and Control by Network Models, Wiley, 1977.

153.J.F.Craine, G.R.Martin, Microcomputers in Engineering and Science, Addison Wesley, 1985.

154.O.A.Nurse, C.N.Chrystall, Using Simulation and Expert Systems for Real Time Control of a Flexible Manufacturing Cell at IBM Havant, p165-176, Proc. 4th Intl. Conf. Simulation in Manufacturing, IFS, 1988.

155.P.W.Udo Graefe, A.W.Chan, M.Levi, A Production Control Aid for Managers of Manufacturing Plants, p55-64, Proc. 1st. Intl. Conf. Simulation in Manufacturing, IFS, 1985.

BIBLIOGRAPHY

1.D.M.Etter, Structured FORTRAN 77 for Engineers and Scientists, 2nd ed., Benjamin Cummings, 1987.

2.R.Wild, Production and Operations Management, 4th ed., Cassell, 1989.

3.S.Bennett, Real-Time Computer Control: An Introduction, Prentice Hall, 1988.

4.T.Dettmann, J.Kyle, M.Johnson, DOS Programmers Reference, 3rd ed., Que, 1992.

5.H.A.Taha, Operations Research An Introduction, 3rd ed, MacMillan, 1982.

6.Dairy Handbook, Alfa-Laval AB, Lund, Sweden.

7.J.Giarratano, G.Riley, Expert Systems Principles and Programming, PWS-KENT, 1989.

8.B.S.Steel, D.Westwood, LPA 386-PROLOG Programming Guide, version 1.2.

9.B.D.Steel, LPA 386-PROLOG 32-bit Assembler Interface, 1992.

10.SEE-WHY Users Guide, AT and T Istel.

11.Lahey Fortran F77l Users Guide, Assembly Language Interface.

ABBREVIATIONS USED

AGV - Automated Guided Vehicle
AI - Artificial Intelligence
APICS - American Production and Inventory Control Society
AS - Activity Scanning
BOM - Bill of Materials
BOP - Bill of Process
BPS - Batch Process Scheduler
CAO - Computer Aided Operation
CHS - Constrained Heuristic Search
CIP - Cleaning In Place
CSP - Constraint Satisfaction Problem
DES - Discrete Event Simulation
ES - Expert System
FIFO - First in First Out
FIS - Finite Intermediate Storage
FMCG - Fast Moving Consumer Goods
FMS - Flexible Manufacturing System
FOOPS - Forward Chaining Object Oriented Production System
IFE - Intelligent Front End
IID - Independent Identically Distributed
MILP - Mixed Integer Linear Programming
MIS - Mixed Intermediate Storage
NIS - No Intermediate Storage
NW - No Wait
OPT - Optimised Production Technology
OR - Operations Research
PC - Personal Computer
POMA - Projected Operation Modification Algorithm
SPL - Simulation Programming Language
SRL - Schema Representation Language
STN - State Task Network
UIS - Unlimited Intermediate Storage
VIS - Visual Interactive Simulation
ZW - Zero Wait

APPENDIX B DOCUMENTATION RELATING TO ELICITATION OF SCHEDULING AND CONFIGURATION RULES

To: Mr.G.Wilkinson
Mr.A.Warden
Mr.D.Potter

From: W.R.Goodall

Date: 25/9/92

MINSTERLEY MILK HANDLING SYSTEM GOALS:

The following list of goals have been derived from the meetings held so far regarding the rules to control the milk handling simulation model.

A set of rules for meeting each goal needs to be developed, and a hierarchy of goals is needed to represent a strategy to drive the application of scheduling and plant configuration rules as the simulation is running.

For example if the top level goal for the system is considered to be:

'Meet service level requirements to user departments and customers'

then rules relating to the satisfaction of this goal should be applied before rules relating to a goal such as:

'Minimise time spent by product in storage'.

Therefore the goals as listed need to be evaluated for completeness and importance, and then put into a hierachy representing the control strategy for the plant. This should result in a strategy which ensures that the key goals for the control of the plant are satisfied first, with goals lower down in the hierarchy 'refining' the schedule with desired characteristics.

When carrying out this evaluation it can be seen that many of the goals in the list are actually sub-goals of other goals or can be broken down into goal/sub-goal structures e.g.

'Meet factory milk requirements with priority over ex-factory requirements'

could be broken down into the following structure:

Goal:Meet factory and ex-factory milk requirements
Subgoal:Meet immediate factory requirements
Subgoal:Meet future factory requirements
Subgoal:Meet immediate ex-factory requirements
Subgoal:Meet future ex-factory requirements

Some of the goals relate to plant configuration rather than overall operating strategy e.g.

Goal:Meet system quality requirements
Subgoal:Minimise the age of milk within the system
Subgoal:Use the 'oldest' milk first

implies that the rules for choice of storage vessels as the source for a route should always choose the oldest milk first (although the exception is cottage cheese), and therefore the structure of a hierarchy for these goals represents a strategy for the use of resources in routes.

The following list of goals was derived from three meetings held at Minsterley over the last nine months.

Top Level General Goals:

Meet service level requirements to user departments and customers.

Meet system quality requirements.

Maximise the utilisation of key assets.

Minimise time spent by product in storage.

Minimise the active period of system input.

General Goals related to milk handling:

100% service level.

Minimise the age of milk within the system.

Control stock to balance the requirement to supply customers against space to accept supplies.

Predict final stock levels in advance.

Control the separation end point.

Meet targets for final stock levels within upper and lower bounds.

Create windows in production for staff release.

Maintain milk reception contractual obligations at a level of 100%.

Specific goals related to the control of the system:

Handle ex-farm milk at milk reception in the time window [0900 -> 1800].

Handle accomodation milk at milk reception in the time windows [0800 -> 0900] and [1900 -> 0300].

Segregate enough Class 'A' milk (today's ex farm delivery) at milk reception to meet cottage cheese requirements.

Ensure that the age of milk for despatch is <= 36 hours old.(?from what point is the age of the milk measured ?).

For other production area requirements:
- Keep the age of the milk to a minimum.
- Use the 'oldest' milk first.

Maintain a fill only policy for milk silos and skim silos.

Maintain an empty/clean policy for milk silos and skim silos.

Meet cottage requirements without supplying excess stock.

Meet cottage cheese requirements with a continuous period of separation. i.e. don't mix requiremnts of raw and pasteurised skim which necessitate a clean when going from raw -> pasteurised.

Keep departments running.

Meet factory milk requirements with priority over ex-factory requirements.

Meet immediate factory requirements with priority over stock for later use within the factory.

Meet requirement for stock within the factory for later use with priority over ex-factory requirements.

Possible Rules relating to goals:

The order of these goals/rules as they are written is not important. That is dependent upon the strategy, to be decided, for attempting to satisfy the goals.

A basic goal/rule structure to form a single strategy to meet service level goals

Goal:Meet immediate factory requirements

(The following rules interpret this goal as referring to the situation of a product requirement becoming critical) (? when can a product be considered critical ?).

IF there is an immediate requirement for a batch of product
AND the resources to supply this product are free
THEN set up and configure the relevant subroutes to supply this batch of product

IF there is an immediate requirement for a batch of product
AND the resources to supply this product are NOT free
AND there is an activity in production of a lower priority product to this one which utilises the resources required
THEN shut down the currently running activity and set up and configure the relevant subroutes to supply this batch of product with the free'd resources.

IF there is an immediate requirement for a batch of product
AND the resources for the relevant activity are NOT free
AND there is an activity in the sub-route sequence which has not yet been started for a product of lower priority which utilises the resources required
THEN remove the lower priority product activity from the sequence
AND set up and configure the relevant subroutes to supply this batch of product with the free'd resources

Goal:Meet immediate ex-factory requirements

IF there is an immediate requirement for ex-factory product
AND the resources for the relevant activity are free
THEN set up and configure the subroutes to achieve this requirement

IF there is an immediate requirement for ex-factory product
AND the resources for the relevant activity are NOT free
AND there is an activity in production of a lower priority product to this one which utilises the resources required
THEN shut down the currently running activity and set up and configure the relevant subroutes to supply this batch of product with the free'd resources.

IF there is an immediate requirement for ex-factory product
AND the resources for the relevant activity are NOT free
AND there is an activity in the sub-route sequence which has not yet been started for a product of lower priority which utilises the resources required
THEN remove the lower priority product activity from the sequence
AND set up and configure the relevant subroutes to supply this batch of product with the free'd resources

Goal:Meet future factory requirements

(The following rules interpret this goal as meaning to recognise where conflict for resources such as plant/storage/milk could occur because of product requirements which may at some stage exceed the capacity of the resources i.e. by doing a capacity analysis of the plant and identifying potential problem areas, the smoothing of resource utilisation can be carried out to prevent later problems occurring)

IF there is a requirement for a product at some time in the future
AND there are other product requirements with activities which require the same resources (i.e. there may be a conflict for resources)
AND these resources are currently free
AND there are some activities for the product which could be carried out early on these critical resources wihout breaching quality constraints
THEN set up and configure the relevant activities now

Goal:Meet future ex-factory requirements

IF there is a requirement for an ex-factory product at some time in the future
AND there are other product requirements with activities which require the same resources (i.e. there may be a conflict for resources)
AND these resources are currently free
AND there are some activities for the product which could be carried out early on these critical resources wihout breaching quality constraints
THEN set up and configure the relevant activities now

Process subroute configuration rules:

These should specify an order for the choice of resources for a particular activity and relate to the goals for choice of resources.

The order of choice is controlled by specifying configuration sub-goals for the route. This is particularly important because each choice made has a bearing on the remaining choices that can be made. Other goals for preferential choice of entities in route positions should also be taken into account.

Are there different cases for these activities when the order of decision making will be different? e.g. source/processor/sink for one case and sink/processor/source for another case.

For routes where there is only one choice possible for a position in the original data then no specific rule is required for configuration of this position.

Each time a choice is made the choices for the remaining positions will be updated automatically to represent the additional constraints imposed on them by the reduced connections now possible. This will simplify the amount of information which needs to be expressed in the rules about each position.

In addition to further simplify the rules, if a silo is empty it must be clean to be a choice for filling/emptying.

If as a result of other choices made for positions in the route only one choice remains for the next position then this will become the choice by default:

Default choice rule:

IF there is only one choice for this position
THEN set up the subroute with this element and set the goal to configure the remainder of the route.

Activity unload milk:

Goal:Configure activity unload_milk

IF the activity is unload milk
THEN the first sub-goal is choose a milk silo

Sub-goal:Choose a milk silo:

IF the vehicle to be unloaded is a bulk farm tanker
AND there is a milk silo currently being filled with Grade 'A' milk
THEN choose this silo to fill into

IF the vehicle to be unloaded is a bulk farm tanker
AND there is NO milk silo currently being filled with Grade 'A' milk
AND there is an empty silo
THEN choose this silo to fill into
AND set subgoal choose silo input line

IF the vehicle to be unloaded is an accomodation vehicle
AND there is currently a silo being filled with Grade 'B' milk
THEN choose this silo to fill into

IF the vehicle to be unloaded is an accomodation vehicle
AND there is NO milk silo currently being filled with Grade 'B' milk
AND there is an empty silo
THEN choose this silo to fill into
AND set subgoal choose silo input line

Subgoal: choose silo input line

IF there is a choice between the two input lines
THEN choose the first one found

Activity separate milk to cottage cheese:

(These rules take into account the preference to use Grade 'A' milk for cottage cheese and will use a silo that is filling if there is no other choice)

Goal: Configure activity separate milk to cottage cheese

IF the activity is separate milk to cottage cheese
THEN the first subgoal is choose a milk silo for cottage cheese supply.

Subgoal:Choose a milk silo for cottage cheese supply.

IF there is a silo choice containing Grade 'A' milk
AND this choice is NOT filling
THEN choose this silo for cottage cheese supply
AND set the second subgoal choose a separator

IF there is a silo choice containing Grade 'A' milk
THEN choose this silo for cottage cheese supply

Subgoal:Choose a separator to separate milk to cottage cheese

IF the factory milk intake is high
AND there is a choice for the separator
including separator 3
THEN choose separator 3 for this route
AND set third subgoal choose a cottage cheese silo for input

IF the factory milk intake is low
AND there is a choice of separator as either separator 1 or separator 2
THEN choose the separator which can run longest
AND set third subgoal choose a cottage cheese silo for input

Subgoal:Choose a cottage cheese silo for input

IF there is a cottage cheese silo choice which is empty
THEN choose this silo for input

IF there is NOT a silo which is empty
AND there is a choice which is NOT currently emptying
AND this choice contains the freshest skim
THEN choose this silo for input.

IF there is NOT a silo which is empty
AND there is a choice which is currently emptying
THEN choose this silo for input.

Activity separate milk to evaporator silos:

Goal:configure activity separate milk to evaporator silos
(Two possible routes which can run in parallel- evaporator line 1, and evaporator line 2)

Subgoal: configure evaporator line 1 route:

The availability of the processor determines whether a choice is possible for the source for this route as milk or skim.
Possible cases:
Case 1.The evaporator line skim pump and at least one separator are available --> The choice for source could be milk or skim
Case 2.Only the evaporator line skim pump is available --> the choice for source can only be skim
(The case that only a separator is available will not arise, because if the skim pump is in use or not available then it must be pumping skim down to the evaporators, or the line is cleaning)
Need to know what the decision making process would be in case 1: i.e. How will the source be decided upon? How will the processor be decided upon?

Therefore first need to set the context for the decisions about configuring this route:

Case 1:

IF the processor could be the evaporator skim pump or a separator
AND conditions best suit the the movement of milk from milk reception
THEN set the subgoal use milk

IF the processor could be the evaporator skim pump or a separator
AND conditions best suit the use of skim from skim storage
THEN set the subgoal use skim from skim storage

Subgoal: use milk

IF the subgoal is use milk
THEN set the subgoal choose a separator

Subgoal: choose a separator

IF the choice is between sep1/2 and sep3
THEN choose from sep1/2 based on given criteria (? how is this choice made ?)
AND set subgoal choose milk silo

IF the choice is between sep1 and 2
THEN choose the separator based on given criteria (? how is this choice made ?)
AND set subgoal choose milk silo

Subgoal: choose milk silo

IF there is a choice which is emptying
THEN choose this silo
AND set the subgoal choose the evaporator skim storage

IF the choices are not emptying
AND there is one which contains the 'oldest' milk
AND this silo is NOT also being filled
THEN choose this silo for the sub-route
AND set the sub-goal choose the evaporator skim storage

Subgoal: choose the evaporator skim storage

IF there is a choice which meets the policy for evaporator skim storage use
THEN choose this silo for the subroute.

Case 2 only the evaporator skim pump is available:

IF the only processor is the evaporator skim pump
THEN set this up in the sub-route and set the subgoal choose skim silo

Subgoal: choose skim silo

(The only case that needs to be considered here is whether there is a choice which is not filling, the case of fill and empty is the default)

IF there is a choice which is NOT filling
AND this choice contains the 'oldest' skim
THEN put this choice into the sub-route
AND set the sub-goal choose the evaporator skim storage

Subgoal: configure evaporator line 2 route:

(In this case the only source is milk, but the processor should be chosen first to reflect the preference for the use of seps 1/2)

IF the subgoal is configure evaporator line 2
THEN set subgoal choose the separator

(The rules for entity choice will then be the same as above for the relevant sub-goals)

Activity: separate milk to the skim silos:

(these rules are based on up to two separators being able to go into one skim silo via the skim silo input line, therefore there are two routes which can operate to one skim silo in parallel)

Goal: configure activity separate milk to skim silos

IF the goal is configure separate milk to skim silos
THEN set the subgoal choose the separator

Subgoal: choose the separator

IF the choice could be sep1/2 or 3
THEN choose sep 1 or 2 depending upon which is currently 'best'
AND set subgoal choose milk silo

IF the choice could be between sep 1 or 2
AND one of them is currently 'best'
THEN choose this one and set the subgoal choose milk silo

Subgoal: choose milk silo

IF there is a choice which is emptying
THEN choose this silo
AND set the subgoal choose skim silo

IF there is a choice which is NOT filling
AND contains the 'oldest' milk
THEN choose this silo
AND set the subgoal choose skim silo

Subgoal: choose skim silo

IF there is a choice which is filling
THEN choose this silo

IF there is a choice which is empty
THEN choose this silo

## APPENDIX B A SAMPLE OF STRATEGY AND CONFIGURATION RULES

```
/* strategy rules */

/* set up a goal to supply immediate factory requirements */

strategy st1:
[goal - strategy with [status - active]]
==>
[assert(goal - immediate_requirements with [status - active]),
 call(delf(goal,strategy))].

/* assess critical batches and assert goal to configure
   those which are criticalprocedures */

strategy imm1:
[goal - immediate_requirements with [status - active]]
==>
[call(immediate_requirements(factory)),
 call(immediate_requirements(ex_factory)),
 call(delf(goal,immediate_requirements)),
 assert(goal - setup_routes with [status - active]),
 assert(goal - system_input with [status - active])].

/* call configuration of milk input routes */

strategy inpta:
[goal - setup_routes with [status - active],
 goal - system_input with [status - active],
 batch - Code with [product - milk,
                    urgency - critical,
                    temporal_activity_status - TS],
 is_on([Activity,can_start],TS)]
==>
[call(configure_activity(Activity,Code))].

strategy inpt3:
[goal - setup_routes with [status - active],
 goal - system_input with [status - active],
 queue - 'RECQ' with [current_members_na - Cmna],
 Cmna == []]
==>
[call(delf(goal,system_input)),
 assert(goal - factory_product with [status - active])].

/* call configuration procedures for all critical batches of
   factory product */

strategy fact1:
[goal - setup_routes with [status - active],
 goal - factory_product with [status - active]]
==>
[call(configure_critical_batches(skim_concentrate)),
 call(configure_critical_batches(cottage_cheese)),
 call(delf(goal,factory_product)),
 call(delf(goal,setup_routes))].
```

```
/* example configuration rules */

/* configuration of route 1 to supply
   evaporator skimmed milk */

/* assert goal to choose milk silo */

rule evs1:
[goal - sep_milk_ev_silos with [status - active],
activity - sep_milk_ev_silos with [current_goals - Subroutes],
is_on('E1ST',Subroutes),
process_route - 'E1ST' with [current_input - Choices],
call(filter_choices(Choices,[product - milk],Choices2)),
Choices2 \= []]
==>
[assert(goal - choose_milk_silo with [status - active])].

/* choose silo with milk grade 2 and assert a goal
   to choose a separator */

rule evs3a:
[goal - choose_milk_silo with [status - active],
 goal - sep_milk_ev_silos with [status - active],
 activity - sep_milk_ev_silos with [current_goals - Subroutes],
 is_on('E1ST',Subroutes),
 process_route - 'E1ST' with [current_input - Choices],
 call(filter_choices(Choices,[product - milk],Choices2)),
 Choices2 \= [],
 call(filter_choices(Choices2,
                     [current_product_grade - 2],Choices2a)),
 Choices2a \= []]
==>
[call(sort_choices(Choices2a,time_product_in,-1,Choices3)),
 call(select_element(process_route,'E1ST',
                                   current_input - CI,Choices3)),
 call(delf(goal,choose_milk_silo)),
 assert(goal - choose_separator with [status - active])].

/* choose a separator other than separator 3 if possible
   and assert a goal to choose a skim line input */

rule evs2:
[goal - choose_separator with [status - active],
goal - sep_milk_ev_silos with [status - active],
activity - sep_milk_ev_silos with [current_goals - Subroutes],
is_on('E1PR',Subroutes),
process_route - 'E1PR' with [current_processor - Choices],
call(filter_choices_NOT(Choices,
                        [descriptor - 'SEP3'],Remaining1)),
Remaining1 \= []]
==>
[call(select_element(process_route ,'E1PR',
                     current_processor  - C,Remaining1)),
 call(delf(goal,choose_separator)),
 assert(goal - skim_line_input with [status - active])].
```

```
/* choose a skim line input and assert a goal to choose
   evaporator silo */

rule evs5a:
[goal - skim_line_input with [status - active],
 goal - sep_milk_ev_silos with [status - active],
 activity - sep_milk_ev_silos with
                                [current_goals - Subroutes],
 is_on('E1SI',Subroutes),
 process_route - 'E1SI' with [current_processor - Choices]]
==>
[call(select_element(process_route,'E1SI',
                                current_processor - CP,Choices)),
 call(delf(goal,skim_line_input)),
 assert(goal - evap_silo with [status - active])].

/* choose an evaporator silo which was last filled and
   exit with configuration complete */

rule evs5:
[goal - evap_silo with [status - active],
 goal - sep_milk_ev_silos with [status - active],
 activity - sep_milk_ev_silos with
                                [current_goals - Subroutes],
 is_on('E1SL',Subroutes),
 process_route - 'E1SL' with [current_output - Choices]]
==>
[call(sort_choices(Choices,time_last_stop_fill,1,Choices2)),
 call(select_element(process_route,'E1SL',current_output - CO,Choices2)),
 call(delf(goal,evap_silo)),
 call(delf(goal,sep_milk_ev_silos))].


/* configuration of evaporator supply route 2 */

/* assert a goal to choose a milk silo */

rule evs7:
[goal - sep_milk_ev_silos with [status - active],
activity - sep_milk_ev_silos with [current_goals - Subroutes],
is_on('E2PR',Subroutes)]
==>
[assert(goal - choose_milk_silo with [status - active])].
```

```
/* choose a milk silo with grade 2 milk and assert a goal
   to choose a separator */

rule evs9a:
[goal - choose_milk_silo with [status - active],
 goal - sep_milk_ev_silos with [status - active],
 activity - sep_milk_ev_silos with
                                    [current_goals - Subroutes],
 is_on('E2ST',Subroutes),
 process_route - 'E2ST' with [current_input - Choices2],
 call(filter_choices(Choices2,
                          [current_product_grade - 2],Choices2a)),
 Choices2a \= []]
==>
[call(sort_choices(Choices2a,time_product_in,-1,Choices3)),
 call(select_element(process_route,'E2ST',
                                        current_input - CI,Choices3)),
 call(delf(goal,choose_milk_silo)),
 assert(goal - choose_separator with [status - active])].

/* choose a separator other than separator 3 if possible
   and assert a goal to choose an evaporator silo */

rule evs8:
[goal - choose_separator with [status - active],
goal - sep_milk_ev_silos with [status - active],
activity - sep_milk_ev_silos with [current_goals - Subroutes],
is_on('E2PR',Subroutes),
process_route - 'E2PR' with [current_processor - Choices],
call(filter_choices_NOT(Choices,
                                    [descriptor - 'SEP3'],Remaining1)),
Remaining1 \= []]
==>
[call(select_element(process_route ,'E2PR',
                                    current_processor  - C,Remaining1)),
 call(delf(goal,choose_separator)),
 assert(goal - evap_silo with [status - active])].

/* choose an evaporator silo on the basis of time last filled
   and exit with the route configured */

rule evs11:
[goal - evap_silo with [status - active],
 goal - sep_milk_ev_silos with [status - active],
 activity - sep_milk_ev_silos with
                                    [current_goals - Subroutes],
 is_on('E2SL',Subroutes),
 process_route - 'E2SL' with [current_output - Choices]]
==>
[call(sort_choices(Choices,time_last_stop_fill,1,Choices2)),
 call(select_element(process_route,'E2SL',
                                        current_output - CO,Choices2)),
 call(delf(goal,evap_silo)),
 call(delf(goal,sep_milk_ev_silos))].


/* configuration rules for skimmed milk supply to the
   cottage cheese department */
```

```
/* assert a goal to choose a milk silo */

rule ccs1:
[goal - sep_milk_cc_silos with [status - active],
 activity - sep_milk_cc_silos with [current_goals - Goals]]
==>
[assert(goal - choose_milk_silo with [status - active])].

/* choose a milk silo with grade 1 milk and assert a goal
   to choose a separator */

rule ccs2a:
[goal - choose_milk_silo with [status - active],
goal - sep_milk_cc_silos with [status - active],
process_route - 'CCST' with [current_input - Choices],
call(filter_choices(Choices,
                    [current_product_grade - 1],Remaining1)),
Remaining1 \= []]
==>
[call(sort_choices(Remaining1,current_contents,1,Choices2)),
call(select_element(process_route,'CCST',
                                    current_input - CI,Choices2)),
call(delf(goal,choose_milk_silo)),
assert(goal - choose_separator with [status - active])].

/* choose any separator available and assert a goal
   to choose a cottage cheese silo */

rule ccs3:
[goal - choose_separator with [status - active],
goal - sep_milk_cc_silos with [status - active],
process_route - 'CSP1' with [current_processor - Choices]]
==>
[call(select_element(process_route,'CSP1',
                                  current_processor - CP,Choices)),
call(delf(goal,choose_separator)),
assert(goal - choose_cc_silo with [status - active])].

/* choose a cottage cheese silo with least contents */

rule ccs4:
[goal - choose_cc_silo with [status - active],
goal - sep_milk_cc_silos with [status - active],
process_route - 'CSLS' with [current_output - Choices]]
==>
[call(delf(goal,choose_cc_silo)),
call(sort_choices(Choices,current_contents,-1,Choices2)),
call(select_element(process_route,'CSLS',
                                  current_output - CO,Choices2))].

/* exit with cottage chees supply route configured */

rule ccs5:
[goal - sep_milk_cc_silos with [status - active]]
==>
[call(delf(goal,sep_milk_cc_silos))].
```

APPENDIX C (9.1). PLANT DATABASE AT TIME ZERO NO ROUTES CONFIGURED

/* active product batches in the system */

Frame: batch Instance: ev01
  Slot: ako - [val batch]
  Slot: batch_code - [val ev01]
  Slot: urgency - [val non_critical]
  Slot: product - [val skim_concentrate]
  Slot: activity_status -
                    [val [sep_milk_ev_silos 0 [0 0 ]]
                    [evap_skim 0 [0 0 ]]]
  Slot: total_float - [val 0]
  Slot: temporal_activity_status -
                    [val [sep_milk_ev_silos can_start ]
                    [evap_skim cannot_start ]]
Time stamp: 0

Frame: batch Instance: ev02
  Slot: ako - [val batch]
  Slot: batch_code - [val ev02]
  Slot: urgency - [val non_critical]
  Slot: product - [val skim_concentrate]
  Slot: activity_status -
                    [val [sep_milk_ev_silos 0 [0 0 ]]
                    [evap_skim 0 [0 0 ]]]
  Slot: total_float - [val 0]
  Slot: temporal_activity_status -
                    [val [sep_milk_ev_silos cannot_start ]
                    [evap_skim cannot_start ]]
Time stamp: 0

Frame: batch Instance: cc01
  Slot: ako - [val batch]
  Slot: urgency - [val non_critical]
  Slot: batch_code - [val cc01]
  Slot: product - [val cottage_cheese]
  Slot: activity_status -
                    [val [sep_milk_cc_silos 0 [0 0 ]]
                    [past_skim_cc_vats 0 [0 0 ]]]
  Slot: total_float - [val 0]
  Slot: temporal_activity_status -
                    [val [sep_milk_cc_silos can_start ]
                    [past_skim_cc_vats cannot_start ]]
Time stamp: 0

```
/* storage vessels */
/* milk silos */

Frame: storage Instance: MSL1
  Slot: ako - [val storage]
  Slot: description - [val milk silo 1]
  Slot: code - [val V01]
  Slot: capacity - [val 227.300]
  Slot: contents_at_last_update - [val 227.300]
  Slot: current_contents - []
  Slot: time_contents_last_updated - [val 0]
  Slot: current_rate_in - [val 0]
  Slot: current_rate_out - [val 0]
  Slot: time_product_out - [val 0]
  Slot: time_product_in - [val -350]
  Slot: time_last_stop_fill - [val -200]
  Slot: time_last_stop_empty - [val 0]
  Slot: contents_status - [val 1]
  Slot: system_event - [val maxcon]
  Slot: configuration - [val MSL1 ---> and : [i,o]
                                       i ---> and : [OLN1,OLN2]
                                       o ---> or : [RLN1,RLN2] ]
  Slot: work_area - [val milk_reception]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: activity_state_attribute4 - [val 2]
  Slot: current_product_grade - [val 2]
  Slot: product - [val milk]
Time stamp: 1

Frame: storage Instance: MSL2
  Slot: ako - [val storage]
  Slot: description - [val milk silo 2]
  Slot: code - [val V02]
  Slot: capacity - [val 227.300]
  Slot: contents_at_last_update - [val 0]
  Slot: current_contents - []
  Slot: time_contents_last_updated - [val 0]
  Slot: current_rate_in - [val 0]
  Slot: current_rate_out - [val 0]
  Slot: time_product_out - [val 0]
  Slot: time_product_in - [val 0]
  Slot: time_last_stop_fill - [val 0]
  Slot: time_last_stop_empty - [val 0]
  Slot: contents_status - [val 2]
  Slot: system_event - [val mincon]
  Slot: configuration - [val MSL2 ---> and : [i,o]
                                       i ---> and : [OLN1,OLN2]
                                       o ---> or : [RLN1,RLN2] ]
  Slot: work_area - [val milk_reception]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: activity_state_attribute4 - [val 1]
  Slot: current_product_grade - [val 0]
  Slot: product - [val milk]
Time stamp: 1
```

Frame: storage Instance: MSL3
Slot: ako - [val storage]
Slot: description - [val milk silo 3]
Slot: code - [val V03]
Slot: capacity - [val 227.300]
Slot: contents_at_last_update - [val 210]
Slot: current_contents - []
Slot: time_contents_last_updated - [val 0]
Slot: current_rate_in - [val 0]
Slot: current_rate_out - [val 0]
Slot: time_product_out - [val 0]
Slot: time_product_in - [val -200]
Slot: time_last_stop_fill - [val 0]
Slot: time_last_stop_empty - [val 0]
Slot: contents_status - [val 0]
Slot: system_event - [val maxcon]
Slot: configuration - [val MSL3 ---> and : [i,o]
i ---> and : [OLN1,OLN2]
o ---> or : [RLN1,RLN2] ]
Slot: work_area - [val milk_reception]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: activity_state_attribute4 - [val 2]
Slot: current_product_grade - [val 1]
Slot: product - [val milk]
Time stamp: 1

Frame: storage Instance: MSL4
Slot: ako - [val storage]
Slot: description - [val milk silo 4]
Slot: code - [val V04]
Slot: capacity - [val 227.300]
Slot: contents_at_last_update - [val 0]
Slot: current_contents - []
Slot: time_contents_last_updated - [val 0]
Slot: current_rate_in - [val 0]
Slot: current_rate_out - [val 0]
Slot: time_product_out - [val 0]
Slot: time_product_in - [val 0]
Slot: time_last_stop_fill - [val 0]
Slot: time_last_stop_empty - [val 0]
Slot: contents_status - [val 2]
Slot: system_event - [val mincon]
Slot: configuration - [val MSL4 ---> and : [i,o]
i ---> and : [OLN1,OLN2]
o ---> or : [RLN1,RLN2] ]
Slot: work_area - [val milk_reception]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: activity_state_attribute4 - [val 1]
Slot: current_product_grade - [val 0]
Slot: product - [val milk]
Time stamp: 1

/* cottage cheese skimmed milk storage silos */

Frame: storage Instance: CCS1
Slot: ako - [val storage]
Slot: description - [val cc skim silo 1]
Slot: code - [val V05]
Slot: capacity - [val 90.920]
Slot: contents_at_last_update - [val 20]
Slot: current_contents - []
Slot: time_contents_last_updated - [val 0]
Slot: current_rate_in - [val 0]
Slot: current_rate_out - [val 0]
Slot: time_product_out - [val 0]
Slot: time_product_in - [val 0]
Slot: time_last_stop_fill - [val 0]
Slot: time_last_stop_empty - [val 0]
Slot: contents_status - [val 0]
Slot: system_event - [val maxcon]
Slot: configuration - [val CCS1 ---> and : [i,o]
i ---> and : [CRLN]
o ---> and : [CPST] ]
Slot: work_area - [val cottage_cheese]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: activity_state_attribute4 - [val 2]
Slot: current_product_grade - [val 1]
Slot: product - [val skim]
Time stamp: 0

Frame: storage Instance: CCS2
Slot: ako - [val storage]
Slot: description - [val cc skim silo 2]
Slot: code - [val V06]
Slot: capacity - [val 90.920]
Slot: contents_at_last_update - [val 0]
Slot: current_contents - []
Slot: time_contents_last_updated - [val 0]
Slot: current_rate_in - [val 0]
Slot: current_rate_out - [val 0]
Slot: time_product_out - [val 0]
Slot: time_product_in - [val 0]
Slot: time_last_stop_fill - [val 0]
Slot: time_last_stop_empty - [val 0]
Slot: contents_status - [val 0]
Slot: system_event - [val mincon]
Slot: configuration - [val CCS2 ---> and : [i,o]
i ---> and : [CRLN]
o ---> and : [CPST] ]
Slot: work_area - [val cottage_cheese]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: activity_state_attribute4 - [val 1]
Slot: current_product_grade - [val 1]
Slot: product - [val skim]
Time stamp: 0

/* evaporator skimmed milk storage silos */

Frame: storage Instance: EVS1
Slot: ako - [val storage]
Slot: description - [val evaporator skim silo 1]
Slot: code - [val V07]
Slot: capacity - [val 181]
Slot: contents_at_last_update - [val 0]
Slot: current_contents - []
Slot: time_contents_last_updated - [val 0]
Slot: current_rate_in - [val 0]
Slot: current_rate_out - [val 0]
Slot: time_product_out - [val 0]
Slot: time_product_in - [val 0]
Slot: time_last_stop_fill - [val -100]
Slot: time_last_stop_empty - [val -10]
Slot: contents_status - [val 2]
Slot: system_event - [val mincon]
Slot: configuration - [val EVS1 ---> and : [i,o]
i ---> and : [ELN1,ELN2]
o ---> or : [EVR1,EVR2] ]
Slot: work_area - [val evaporators]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: activity_state_attribute4 - [val 1]
Slot: current_product_grade - [val 0]
Slot: product - [val skim]
Time stamp: 0

Frame: storage Instance: EVS2
Slot: ako - [val storage]
Slot: description - [val evaporator skim silo 2]
Slot: code - [val V08]
Slot: capacity - [val 90.920]
Slot: contents_at_last_update - [val 20]
Slot: current_contents - []
Slot: time_contents_last_updated - [val 0]
Slot: current_rate_in - [val 0]
Slot: current_rate_out - [val 0]
Slot: time_product_out - [val 0]
Slot: time_product_in - [val -50]
Slot: time_last_stop_fill - [val 0]
Slot: time_last_stop_empty - [val 0]
Slot: contents_status - [val 0]
Slot: system_event - [val mincon]
Slot: configuration - [val EVS2 ---> and : [i,o]
i ---> and : [ELN1,ELN2]
o ---> or : [EVR1,EVR2] ]
Slot: work_area - [val evaporators]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: activity_state_attribute4 - [val 2]
Slot: current_product_grade - [val 2]
Slot: product - [val skim]
Time stamp: 0

```
/* semi-continuous process plant items */
/* separators and skimmed milk outputs */

Frame: continuous Instance: SEP1
  Slot: ako - [val continuous]
  Slot: descriptor - [val SEP1]
  Slot: code - [val P07]
  Slot: configuration - [val SEP1 ---> and : [i,o]
                                       i ---> and : [RLN1]
                                       o ---> and : [SPS1,SPC1] ]
  Slot: work_area - [val dairy]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0.334 / klitres_min]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0

Frame: process_line Instance: SPS1
  Slot: ako - [val process_line]
  Slot: descriptor - [val SPS1]
  Slot: code - [val P34]
  Slot: configuration - [val SPS1 ---> and : [i,o]
                                  i ---> and : [SEP1]
                                  o ---> or : [SIP1,SIP2,CRLN,ELN2] ]
  Slot: work_area - [val dairy]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0

Frame: continuous Instance: SEP2
  Slot: ako - [val continuous]
  Slot: descriptor - [val SEP2]
  Slot: code - [val P08]
  Slot: configuration - [val SEP2 ---> and : [i,o]
                                       i ---> and : [RLN1]
                                       o ---> and : [SPS2,SPC2] ]
  Slot: work_area - [val dairy]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_process_route - [val ]
  Slot: current_rate - [val 0.334 / klitre_min]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0

Frame: process_line Instance: SPS2
  Slot: ako - [val process_line]
  Slot: descriptor - [val SPS2]
  Slot: code - [val P35]
  Slot: configuration - [val SPS2 ---> and : [i,o]
                                  i ---> and : [SEP2]
                                  o ---> or : [SIP1,SIP2,CRLN,ELN2] ]
  Slot: work_area - [val dairy]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0
```

```
Frame: continuous Instance: SEP3
  Slot: ako - [val continuous]
  Slot: descriptor - [val SEP3]
  Slot: code - [val P09]
  Slot: configuration - [val SEP3 ---> and : [i,o]
                                        i ---> and : [RLN2]
                                        o ---> and : [SPS3,SPC3] ]
  Slot: work_area - [val dairy]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0.334 / klitres_min]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0

Frame: process_line Instance: SPS3
  Slot: ako - [val process_line]
  Slot: descriptor - [val SPS3]
  Slot: code - [val P36]
  Slot: configuration - [val SPS3 ---> and : [i,o]
                                  i ---> and : [SEP3]
                                  o ---> or : [SIP1,SIP2,CRLN,ELN2] ]
  Slot: work_area - [val dairy]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0

/* evaporators */

Frame: continuous Instance: EVR1
  Slot: ako - [val continuous]
  Slot: descriptor - [val EVR1]
  Slot: code - [val P10]
  Slot: configuration - [val EVR1 ---> and : [i,o]
                                        i ---> and : [EVS1,EVS2]
                                        o ---> and : [OUTP] ]
  Slot: work_area - [val evaporators]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_process_route - [val ]
  Slot: current_rate - [val 0.252 / klitres_min]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0

Frame: continuous Instance: EVR2
  Slot: ako - [val continuous]
  Slot: descriptor - [val EVR2]
  Slot: code - [val P11]
  Slot: configuration - [val EVR2 ---> and : [i,o]
                                        i ---> and : [EVS1,EVS2]
                                        o ---> and : [OUTP] ]
  Slot: work_area - [val evaporators]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_process_route - [val ]
  Slot: current_rate - [val 0.151 / klitres_min]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0
```

```
/* process lines */
/* evaporator supply lines */

Frame: process_line Instance: ELN1
  Slot: ako - [val process_line]
  Slot: descriptor - [val ELN1]
  Slot: code - [val P21]
  Slot: configuration - [val ELN1 ---> and : [i,o]
                                  i ---> or : [SKM1,SKM2,SKM3,SKLN]
                                  o ---> or : [EVS1,EVS2] ]
  Slot: work_area - [val evaporators]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0

Frame: process_line Instance: ELN2
  Slot: ako - [val process_line]
  Slot: descriptor - [val ELN2]
  Slot: code - [val P22]
  Slot: configuration - [val ELN2 ---> and : [i,o]
                                  i ---> or : [SPS1,SPS2,SPS3]
                                  o ---> or : [EVS1,EVS2] ]
  Slot: work_area - [val evaporators]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0

/* raw milk supply lines to separators from milk silos */

Frame: process_line Instance: RLN1
  Slot: ako - [val process_line]
  Slot: descriptor - [val RLN1]
  Slot: code - [val P18]
  Slot: configuration - [val RLN1 ---> and : [i,o]
                                  i ---> or : [MSL1,MSL2,MSL3,MSL4]
                                  o ---> and : [SEP1,SEP2] ]
  Slot: work_area - [val dairy]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0

Frame: process_line Instance: RLN2
  Slot: ako - [val process_line]
  Slot: descriptor - [val RLN2]
  Slot: code - [val P19]
  Slot: configuration - [val RLN1 ---> and : [i,o]
                                  i ---> or : [MSL1,MSL2,MSL3,MSL4]
                                  o ---> and : [SEP3] ]
  Slot: work_area - [val dairy]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0
```

/* milk reception unloading bay manifold output lines to milk silos */

Frame: process_line Instance: OLN1
Slot: ako - [val process_line]
Slot: descriptor - [val OLN1]
Slot: code - [val P27]
Slot: configuration - [val OLN1 ---> and : [i,o]
i ---> and : [MLN1,MLN2,MLN3]
o ---> or : [MSL1,MSL2,MSL3,MSL4] ]
Slot: work_area - [val milk_reception]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: current_rate - [val 0]
Slot: current_process_route - [val ]
Slot: activity_state_attribute4 - [val 2]
Time stamp: 1

Frame: process_line Instance: OLN2
Slot: ako - [val process_line]
Slot: descriptor - [val OLN2]
Slot: code - [val P28]
Slot: configuration - [val OLN2 ---> and : [i,o]
i ---> and : [MLN1,MLN2,MLN3]
o ---> or : [MSL1,MSL2,MSL3,MSL4] ]
Slot: work_area - [val milk_reception]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: current_rate - [val 0]
Slot: current_process_route - [val ]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

/* milk reception unloading bay lines to milk silo input manifold
output lines */

Frame: process_line Instance: MLN1
Slot: ako - [val process_line]
Slot: descriptor - [val MLN1]
Slot: code - [val P24]
Slot: configuration - [val MLN1 ---> and : [i,o]
i ---> or : [BY1A,BY1B]
o ---> or : [OLN1,OLN2] ]
Slot: work_area - [val milk_reception]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: current_rate - [val 0]
Slot: current_process_route - [val ]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

```
Frame: process_line Instance: MLN2
  Slot: ako - [val process_line]
  Slot: descriptor - [val MLN2]
  Slot: code - [val P25]
  Slot: configuration - [val MLN2 ---> and : [i,o]
                                       i ---> or : [BY2A,BY2B]
                                       o ---> or : [OLN1,OLN2] ]
  Slot: work_area - [val milk_reception]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_line Instance: MLN3
  Slot: ako - [val process_line]
  Slot: descriptor - [val MLN3]
  Slot: code - [val P26]
  Slot: configuration - [val MLN3 ---> and : [i,o]
                                       i ---> or : [BY3A,BY3B]
                                       o ---> or : [OLN1,OLN2] ]
  Slot: work_area - [val milk_reception]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

/* cottage cheese skimmed milk supply line */

Frame: process_line Instance: CRLN
  Slot: ako - [val process_line]
  Slot: descriptor - [val CRLN]
  Slot: code - [val P29]
  Slot: configuration - [val CRLN ---> and : [i,o]
                                       i ---> or : [SPS1,SPS2,SPS3]
                                       o ---> or : [CCS1,CCS2] ]
  Slot: work_area - [val cottage_cheese]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 0

/* general skimmed milk distribution supply line */

Frame: process_line Instance: SKLN
  Slot: ako - [val process_line]
  Slot: descriptor - [val SKLN]
  Slot: code - [val P20]
  Slot: configuration - [val SKLN ---> and : [i,o]
                                  i ---> and : [SIP1,SIP2]
                                  o ---> or : [SKM1,SKM2,SKM3,ELN1] ]
  Slot: work_area - [val dairy]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1
```

/* skimmed milk supply line manifold input lines */

Frame: process_line Instance: SIP1
Slot: ako - [val process_line]
Slot: descriptor - [val SIP1]
Slot: code - [val P30]
Slot: configuration - [val SIP1 ---> and : [i,o]
i ---> or : [SPS1,SPS2,SPS3]
o ---> and : [SKLN] ]
Slot: work_area - [val dairy]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: current_rate - [val 0]
Slot: current_process_route - [val ]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_line Instance: SIP2
Slot: ako - [val process_line]
Slot: descriptor - [val SIP2]
Slot: code - [val P31]
Slot: configuration - [val SIP2 ---> and : [i,o]
i ---> or : [SPS1,SPS2,SPS3]
o ---> and : [SKLN] ]
Slot: work_area - [val dairy]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: current_rate - [val 0]
Slot: current_process_route - [val ]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

/* milk reception area unloading bays */

Frame: transport_bay Instance: BY1A
Slot: ako - [val transport_bay]
Slot: descriptor - [val BY1A]
Slot: code - [val P01]
Slot: configuration - [val BY1A ---> and : [i,o]
i ---> or : [BLKR,ACCM]
o ---> or : [MLN1] ]
Slot: work_area - [val milk_reception]
Slot: source_queues - [val RECQ ]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: current_rate - [val 0.455]
Slot: current_process_route - [val ]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: transport_bay Instance: BY1B
Slot: ako - [val transport_bay]
Slot: descriptor - [val BY1B]
Slot: code - [val P04]
Slot: configuration - [val BY1B ---> and : [i,o]
i ---> or : [BLKR,ACCM]
o ---> or : [MLN1] ]
Slot: work_area - [val milk_reception]
Slot: source_queues - [val RECQ ]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: current_rate - [val 0.455]
Slot: current_process_route - [val ]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: transport_bay Instance: BY2A
Slot: ako - [val transport_bay]
Slot: descriptor - [val BY2A]
Slot: code - [val P02]
Slot: configuration - [val BY2A ---> and : [i,o]
i ---> or : [BLKR,ACCM]
o ---> or : [MLN2] ]
Slot: work_area - [val milk_reception]
Slot: source_queues - [val RECQ ]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: current_rate - [val 0.455]
Slot: current_process_route - [val ]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: transport_bay Instance: BY2B
Slot: ako - [val transport_bay]
Slot: descriptor - [val BY2B]
Slot: code - [val P05]
Slot: configuration - [val BY2B ---> and : [i,o]
i ---> or : [BLKR,ACCM]
o ---> or : [MLN2] ]
Slot: work_area - [val milk_reception]
Slot: source_queues - [val RECQ ]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: current_rate - [val 0.455]
Slot: current_process_route - [val ]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: transport_bay Instance: BY3A
Slot: ako - [val transport_bay]
Slot: descriptor - [val BY3A]
Slot: code - [val P03]
Slot: configuration - [val BY3A ---> and : [i,o]
i ---> or : [BLKR,ACCM]
o ---> or : [MLN3] ]
Slot: work_area - [val milk_reception]
Slot: source_queues - [val RECQ ]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: current_rate - [val 0.455]
Slot: current_process_route - [val ]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

```
Frame: transport_bay Instance: BY3B
  Slot: ako - [val transport_bay]
  Slot: descriptor - [val BY3B]
  Slot: code - [val P06]
  Slot: configuration - [val BY3B ---> and : [i,o]
                                       i ---> or : [BLKR,ACCM]
                                       o ---> or : [MLN3] ]
  Slot: work_area - [val milk_reception]
  Slot: source_queues - [val RECQ ]
  Slot: current_inputs - [val ]
  Slot: current_outputs - [val ]
  Slot: current_rate - [val 0.455]
  Slot: current_process_route - [val ]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

/* process route sub-routes making up production activity routes */
/* sub-routes for evaporator skimmed milk supply route 1 */

Frame: process_route Instance: E1SL
  Slot: ako - [val process_route]
  Slot: description -
                    [val skim to evap silos via evap line 1]
  Slot: links_up - [val ]
  Slot: links_down - [val E1LN ]
  Slot: output_choices - [val EVS1 EVS2 ]
  Slot: processor_choices - [val ELN1 ]
  Slot: input_choices - [val links_down LK09 ]
  Slot: current_output - [val ]
  Slot: current_processor - [val ]
  Slot: current_input - [val ]
  Slot: current_yield - [val 0.900]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val sep_milk_ev_silos]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: E1LN
  Slot: ako - [val process_route]
  Slot: description - [val skim line routing]
  Slot: links_up - [val E1SL ]
  Slot: links_down - [val E1SI ]
  Slot: output_choices - [val links_up LK09 ]
  Slot: processor_choices - [val SKLN ]
  Slot: input_choices - [val links_down LK31 ]
  Slot: current_output - [val ]
  Slot: current_processor - [val ]
  Slot: current_input - [val ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val sep_milk_ev_silos]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1
```

Frame: process_route Instance: E1SI
Slot: ako - [val process_route]
Slot: description - [val skim line inputs]
Slot: links_up - [val E1LN ]
Slot: links_down - [val E1SS ]
Slot: output_choices - [val links_up LK31 ]
Slot: processor_choices - [val SIP1 SIP2 ]
Slot: input_choices - [val links_down LK32 ]
Slot: current_output - [val ]
Slot: current_processor - [val ]
Slot: current_input - [val ]
Slot: current_yield - [val 1]
Slot: input_factor - [val 1]
Slot: remaining_batchsize - [val 0]
Slot: current_batch - []
Slot: current_activity - [val sep_milk_ev_silos]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: E1SS
Slot: ako - [val process_route]
Slot: description - [val skim line routing]
Slot: links_up - [val E1SI ]
Slot: links_down - [val E1PR ]
Slot: output_choices - [val links_up LK32 ]
Slot: processor_choices - [val SPS1 SPS2 SPS3 ]
Slot: input_choices - [val links_down LK33 ]
Slot: current_output - [val ]
Slot: current_processor - [val ]
Slot: current_input - [val ]
Slot: current_yield - [val 1]
Slot: input_factor - [val 1]
Slot: remaining_batchsize - [val 0]
Slot: current_batch - []
Slot: current_activity - [val sep_milk_ev_silos]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: E1PR
Slot: ako - [val process_route]
Slot: description - [val separators to skim outputs]
Slot: links_up - [val E1SS ]
Slot: links_down - [val E1ST ]
Slot: output_choices - [val links_up LK33 ]
Slot: processor_choices - [val SEP1 SEP2 SEP3 ]
Slot: input_choices - [val links_down LK10 ]
Slot: current_output - [val ]
Slot: current_processor - [val ]
Slot: current_input - [val ]
Slot: current_yield - [val 0.900]
Slot: input_factor - [val 1]
Slot: remaining_batchsize - [val 0]
Slot: current_batch - []
Slot: current_activity - [val sep_milk_ev_silos]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

```
Frame: process_route Instance: E1ST
  Slot: ako - [val process_route]
  Slot: description -
                    [val skim or milk storage to seps or pump]
  Slot: links_up - [val E1PR ]
  Slot: links_down - [val ]
  Slot: output_choices - [val links_up LK10 ]
  Slot: processor_choices - [val RLN1 RLN2 ]
  Slot: input_choices - [val MSL1 MSL2 MSL3 MLS4 ]
  Slot: current_output - [val ]
  Slot: current_processor - [val ]
  Slot: current_input - [val ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val sep_milk_ev_silos]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

/* sub-routes for evaporator skimmed milk supply route 2 */

Frame: process_route Instance: E2SL
  Slot: ako - [val process_route]
  Slot: description -
                    [val skim to evap silos via evap line 2]
  Slot: links_up - [val ]
  Slot: links_down - [val E2SS ]
  Slot: output_choices - [val EVS1 EVS2 ]
  Slot: processor_choices - [val ELN2 ]
  Slot: input_choices - [val links_down LK11 ]
  Slot: current_output - [val FREE ]
  Slot: current_processor - [val FREE ]
  Slot: current_input - [val FREE ]
  Slot: current_yield - [val 0.900]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val sep_milk_ev_silos]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: E2SS
  Slot: ako - [val process_route]
  Slot: description - [val sep skim outputs]
  Slot: links_up - [val E2SL ]
  Slot: links_down - [val E2PR ]
  Slot: output_choices - [val links_up LK11 ]
  Slot: processor_choices - [val SPS1 SPS2 SPS3 ]
  Slot: input_choices - [val links_down LK34 ]
  Slot: current_output - [val ]
  Slot: current_processor - [val ]
  Slot: current_input - [val ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val sep_milk_ev_silos]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1
```

Frame: process_route Instance: E2PR
Slot: ako - [val process_route]
Slot: description - [val separators to evap line 2]
Slot: links_up - [val E2SS ]
Slot: links_down - [val E2ST ]
Slot: output_choices - [val links_up LK34 ]
Slot: processor_choices - [val SEP1 SEP2 SEP3 ]
Slot: input_choices - [val links_down LK12 ]
Slot: current_output - [val ]
Slot: current_processor - [val ]
Slot: current_input - [val ]
Slot: current_yield - [val 0.900]
Slot: input_factor - [val 1]
Slot: remaining_batchsize - [val 0]
Slot: current_batch - []
Slot: current_activity - [val sep_milk_ev_silos]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: E2ST
Slot: ako - [val process_route]
Slot: description - [val milk storage to seps]
Slot: links_up - [val E2PR ]
Slot: links_down - [val ]
Slot: output_choices - [val links_up LK12 ]
Slot: processor_choices - [val RLN1 RLN2 ]
Slot: input_choices - [val MSL1 MSL2 MSL3 MLS4 ]
Slot: current_output - [val ]
Slot: current_processor - [val ]
Slot: current_input - [val ]
Slot: current_yield - [val 1]
Slot: input_factor - [val 1]
Slot: remaining_batchsize - [val 0]
Slot: current_batch - []
Slot: current_activity - [val sep_milk_ev_silos]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

/* sub-routes for cottage cheese skimmed milk supply route */

Frame: process_route Instance: CSLS
Slot: ako - [val process_route]
Slot: description - [val cc line to silos]
Slot: links_up - [val ]
Slot: links_down - [val CSSS ]
Slot: output_choices - [val CCS1 CCS2 ]
Slot: processor_choices - [val CRLN ]
Slot: input_choices - [val links_down LK07 ]
Slot: current_output - [val ]
Slot: current_processor - [val ]
Slot: current_input - [val ]
Slot: current_yield - [val 0.900]
Slot: input_factor - [val 1]
Slot: remaining_batchsize - [val 0]
Slot: current_batch - []
Slot: current_activity - [val sep_milk_cc_silos]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: CSSS
Slot: ako - [val process_route]
Slot: description - [val sep skim outputs]
Slot: links_up - [val CSLS ]
Slot: links_down - [val CSP1 ]
Slot: output_choices - [val links_up LK07 ]
Slot: processor_choices - [val SPS1 SPS2 SPS3 ]
Slot: input_choices - [val links_down LK35 ]
Slot: current_output - [val ]
Slot: current_processor - [val ]
Slot: current_input - [val ]
Slot: current_yield - [val 1]
Slot: input_factor - [val 1]
Slot: remaining_batchsize - [val 0]
Slot: current_batch - []
Slot: current_activity - [val sep_milk_cc_silos]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: CSP1
Slot: ako - [val process_route]
Slot: description - [val separators to sep line]
Slot: links_up - [val CSSS ]
Slot: links_down - [val CCST ]
Slot: output_choices - [val links_up LK35 ]
Slot: processor_choices - [val SEP1 SEP2 SEP3 ]
Slot: input_choices - [val links_down LK08 ]
Slot: current_output - [val ]
Slot: current_processor - [val ]
Slot: current_input - [val ]
Slot: current_yield - [val 0.900]
Slot: input_factor - [val 1]
Slot: remaining_batchsize - [val 0]
Slot: current_batch - []
Slot: current_activity - [val sep_milk_cc_silos]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: CCST
Slot: ako - [val process_route]
Slot: description -
[val milk silos to separators for cc skim]
Slot: links_up - [val CSP1 ]
Slot: links_down - [val ]
Slot: output_choices - [val links_up LK08 ]
Slot: processor_choices - [val RLN1 RLN2 ]
Slot: input_choices - [val MSL1 MSL2 MSL3 MSL4 ]
Slot: current_output - [val ]
Slot: current_processor - [val ]
Slot: current_input - [val ]
Slot: current_yield - [val 1]
Slot: input_factor - [val 1]
Slot: remaining_batchsize - [val 0]
Slot: current_batch - []
Slot: current_activity - [val sep_milk_cc_silos]
Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

/* sub-routes for milk unloading route 1 */

Frame: process_route Instance: B1SL
  Slot: ako - [val process_route]
  Slot: description - [val put bay 1 milk to a silo]
  Slot: links_up - [val ]
  Slot: links_down - [val B1ML ]
  Slot: output_choices - [val MSL1 MSL2 MSL3 MSL4 ]
  Slot: processor_choices - [val OLN1 OLN2 ]
  Slot: input_choices - [val links_down LK02 ]
  Slot: current_output - [val MSL4 MSL3 MSL2 MSL1 ]
  Slot: current_processor - [val OLN1 OLN2 ]
  Slot: current_input - [val links_down LK02 ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val unload_milk]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: B1ML
  Slot: ako - [val process_route]
  Slot: description - [val milk line]
  Slot: links_up - [val B1SL ]
  Slot: links_down - [val B1VH ]
  Slot: output_choices - [val links_up LK02 ]
  Slot: processor_choices - [val MLN1 ]
  Slot: input_choices - [val links_down LK01 ]
  Slot: current_output - [val links_up LK02 ]
  Slot: current_processor - [val MLN1 ]
  Slot: current_input - [val links_down LK01 ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val unload_milk]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: B1VH
  Slot: ako - [val process_route]
  Slot: description - [val unload milk from vehicle at bay 1]
  Slot: links_up - [val B1ML ]
  Slot: links_down - [val ]
  Slot: output_choices - [val links_up LK01 ]
  Slot: processor_choices - [val BY1A BY1B ]
  Slot: input_choices - [val RECQ RC1A RC1B ]
  Slot: current_output - [val links_up LK01 ]
  Slot: current_processor - [val BY1A BY1B ]
  Slot: current_input - [val RECQ RC1A RC1B ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val unload_milk]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

/* sub-routes for milk unloading route 2 */

Frame: process_route Instance: B2SL
  Slot: ako - [val process_route]
  Slot: description - [val put bay 2 milk to a silo]
  Slot: links_up - [val ]
  Slot: links_down - [val B2ML ]
  Slot: output_choices - [val MSL1 MSL2 MSL3 MSL4 ]
  Slot: processor_choices - [val OLN1 OLN2 ]
  Slot: input_choices - [val links_down LK04 ]
  Slot: current_output - [val ]
  Slot: current_processor - [val ]
  Slot: current_input - [val ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val unload_milk]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: B2ML
  Slot: ako - [val process_route]
  Slot: description - [val milk line]
  Slot: links_up - [val B2SL ]
  Slot: links_down - [val B2VH ]
  Slot: output_choices - [val links_up LK04 ]
  Slot: processor_choices - [val MLN2 ]
  Slot: input_choices - [val links_down LK03 ]
  Slot: current_output - [val ]
  Slot: current_processor - [val ]
  Slot: current_input - [val ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val unload_milk]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: B2VH
  Slot: ako - [val process_route]
  Slot: description - [val unload milk from vehicle at bay 2]
  Slot: links_up - [val B2ML ]
  Slot: links_down - [val ]
  Slot: output_choices - [val links_up LK03 ]
  Slot: processor_choices - [val BY2A BY2B ]
  Slot: input_choices - [val RECQ RC2A RC2B ]
  Slot: current_output - [val ]
  Slot: current_processor - [val ]
  Slot: current_input - [val ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val unload_milk]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

/* sub-routes for milk unloading route 3 */

Frame: process_route Instance: B3SL
  Slot: ako - [val process_route]
  Slot: description - [val put bay 3 milk to a silo]
  Slot: links_up - [val ]
  Slot: links_down - [val B3ML ]
  Slot: output_choices - [val MSL1 MSL2 MSL3 MSL4 ]
  Slot: processor_choices - [val OLN1 OLN2 ]
  Slot: input_choices - [val links_down LK06 ]
  Slot: current_output - [val ]
  Slot: current_processor - [val ]
  Slot: current_input - [val ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val unload_milk]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: B3ML
  Slot: ako - [val process_route]
  Slot: description - [val milk line from bay to manifold]
  Slot: links_up - [val B3SL ]
  Slot: links_down - [val B3VH ]
  Slot: output_choices - [val links_up LK06 ]
  Slot: processor_choices - [val MLN3 ]
  Slot: input_choices - [val links_down LK05 ]
  Slot: current_output - [val ]
  Slot: current_processor - [val ]
  Slot: current_input - [val ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val unload_milk]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

Frame: process_route Instance: B3VH
  Slot: ako - [val process_route]
  Slot: description - [val unload milk from vehicle at bay 3]
  Slot: links_up - [val B3ML ]
  Slot: links_down - [val ]
  Slot: output_choices - [val links_up LK05 ]
  Slot: processor_choices - [val BY3A BY3B ]
  Slot: input_choices - [val RECQ RC3A RC3B ]
  Slot: current_output - [val ]
  Slot: current_processor - [val ]
  Slot: current_input - [val ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: remaining_batchsize - [val 0]
  Slot: current_batch - []
  Slot: current_activity - [val unload_milk]
  Slot: activity_state_attribute4 - [val 1]
Time stamp: 1

```
/* (Point 1) */
protocol: -1

/* (Point 2) */
conflict set: st1
rule selected: st1
/* rule conditions matched */
(goal - strategy with [status - active]) / 1

adding - (goal - immediate_requirements with [status - active])
rule fired: st1

conflict set: imm1
rule selected: imm1
/* rule conditions matched */
(goal - immediate_requirements with [status - active]) / 2
/* (Point 3) */
/* the procedure immediate_requirements is called by the rule */
batch : cottage_cheese  cc01  is critical
solving branch :sep_milk_cc_silos ---> and : [past_skim_cc_vats]
end node : past_skim_cc_vats ---> end
result of precedence check :
[[sep_milk_cc_silos / can_start,past_skim_cc_vats / cannot_start]]
/* (Point 4) */
batch : cottage_cheese  cc02  is not yet critical
batch : skim_concentrate  ev01  is critical
solving branch :sep_milk_ev_silos ---> and : [evap_skim]
end node : evap_skim ---> end
result of precedence check :
[[sep_milk_ev_silos / can_start,evap_skim / cannot_start]]
batch : skim_concentrate  ev02  is critical
solving branch :sep_milk_ev_silos ---> and : [evap_skim]
end node : evap_skim ---> end
result of precedence check :
[[sep_milk_ev_silos / can_start,evap_skim / cannot_start]]
batch : milk  um01  is critical
end node : unload_milk ---> end
result of precedence check :
[unload_milk / can_start]
batch : milk  um02  is critical
end node : unload_milk ---> end
result of precedence check :
[unload_milk / can_start]
batch : milk  um03  is critical
end node : unload_milk ---> end
result of precedence check :
[unload_milk / can_start]
batch : skim_for_bottling  hc01  is not yet critical
batch : skim_for_bottling  hc02  is not yet critical
/* (Point 5) */
adding - (goal - setup_routes with [status - active])
adding - (goal - system_input with [status - active])
rule fired: imm1

conflict set: inpt3 inpta inpta inpta
rule selected: inpta
/* rule conditions matched */
(goal - setup_routes with [status - active]) / 102
(goal - system_input with [status - active]) / 103
(batch - um03 with [product - milk,urgency - critical,
temporal_activity_status - [[unload_milk,can_start]]]) / 83
is_on([unload_milk,can_start],[[unload_milk,can_start]]) / 0
```

```
/* call to configure_activity */
Configuration of activity: unload_milk
Goallist:
[B1VH,B1ML,B1SL] /* a set of sub-routes for the activity */
Ordered_goals:
[B1VH,B1ML,B1SL] /* put into order */
Initial_choices: /* the initial route position choices */
[[MSL1,MSL2,MSL3,MSL4], /* Position 5 */
[OLN1,OLN2], /* Position 4 */
[MLN1], /* Position 3 */
[BY1A,BY1B], /* Position 2 */
[RECQ,RC1A,RC1B]] /* Position 1 */
Intermediate_choices: /* the results of the unary
[[MSL2,MSL3,MSL4],          constraint check */
[OLN1,OLN2],
[MLN1],
[BY1A,BY1B],
[]]                         /* There are no vehicles in the system so
                               RECQ, RC1A, and RC1B are removed and this
                               route fails as a candidate for this
                               activity */
/* Two other routes are now tested but also fail for the same reasons */
Goallist:
[B2VH,B2ML,B2SL]
Ordered_goals:
[B2VH,B2ML,B2SL]
Initial_choices:
[[MSL1,MSL2,MSL3,MSL4],
[OLN1,OLN2],
[MLN2],
[BY2A,BY2B],
[RECQ,RC2A,RC2B]]
Intermediate_choices:
[[MSL2,MSL3,MSL4],
[OLN1,OLN2],
[MLN2],
[BY2A,BY2B],
[]]
Goallist:
[B3VH,B3ML,B3SL]
Ordered_goals:
[B3VH,B3ML,B3SL]
Initial_choices:
[[MSL1,MSL2,MSL3,MSL4],
[OLN1,OLN2],
[MLN3],
[BY3A,BY3B],
[RECQ,RC3A,RC3B]]
Intermediate_choices:
[[MSL2,MSL3,MSL4],
[OLN1,OLN2],
[MLN3],
[BY3A,BY3B],
[]]
Goallist:
[]
/* configure_activity exits having failed to find a route which can be configured */
configuration of activity: unload_milk  on batch: um03  has failed
rule fired: inpta
```

```
conflict set: inpt3 inpta inpta inpta
rule selected: inpt3
/* rule conditions matched */
(goal - setup_routes with [status - active]) / 102
(goal - system_input with [status - active]) / 103
(queue - RECQ with [current_members_na - []]) / 0
([] == []) / 0
/* (Point 6) */
adding - (goal - factory_product with [status - active])
rule fired: inpt3
/* (Point 7) */
conflict set: fact1
rule selected: fact1
/* rule conditions matched */
(goal - setup_routes with [status - active]) / 102
(goal - factory_product with [status - active]) / 104
/* configure_activity is now called again */
Configuration of activity: sep_milk_ev_silos
Goallist:
[E1SL,E1LN,E1SI,E1SS,E1PR,E1ST]
Ordered_goals:                          /* subroutes are put in order */
[E1ST,E1PR,E1SS,E1SI,E1LN,E1SL]
Initial_choices:
[[EVS1,EVS2],
[ELN1],
[SKLN],
[SIP1,SIP2],
[SPS1,SPS2,SPS3],                  /* route choices as defined in
[SEP1,SEP2,SEP3],                        route data */
[RLN1,RLN2],
[MSL1,MSL2,MSL3,MSL4]]
Intermediate_choices: /* the results of unary constraint check */
[[EVS1,EVS2],
[ELN1],
[SKLN],
[SIP1,SIP2],
[SPS1,SPS2,SPS3],
[SEP1,SEP2,SEP3],
[RLN1,RLN2],
[MSL1,MSL3]]              /* MSL2 and MSL4 are both lost because they
                             are empty */
Remaining_choices: /* results of first binary constraint check */
[[EVS1,EVS2],
[ELN1],
[SKLN],
[SIP1,SIP2],
[SPS1,SPS2,SPS3],
[SEP1,SEP2,SEP3],
[RLN1,RLN2],
[MSL1,MSL3]]
Changes: _00055448        /* Changes remains uninstantiated so no change
                             has been made to the route */
Final_choices: /* these are the choices now currently available
[[EVS1,EVS2],          to the route */
[ELN1],
[SKLN],
[SIP1,SIP2],
[SPS1,SPS2,SPS3],
[SEP1,SEP2,SEP3],
[RLN1,RLN2],
[MSL1,MSL3]]
/* configure_activity now calls rule-based element selection to be carried out */
```

```
/* (Point 8) */
conflict set: evs1
rule selected: evs1
/* rule conditions matched */
(goal - sep_milk_ev_silos with [status - active]) / 117
(activity - sep_milk_ev_silos with [current_goals -
                                    [E1ST,E1PR,E1SS,E1SI,E1LN,E1SL]]) / 118
is_on(E1ST,[E1ST,E1PR,E1SS,E1SI,E1LN,E1SL]) / 0
(process_route - E1ST with [current_input - [MSL1,MSL3]]) / 106
call(filter_choices([MSL1,MSL3],[product - milk],[MSL1,MSL3])) / 0
([MSL1,MSL3] \= []) / 0
/* this rule adds the first decision directing context */
adding - (goal - choose_milk_silo with [status - active])
rule fired: evs1

conflict set: evs3b evs3a evs1
rule selected: evs3a
/* rule conditions matched */
(goal - choose_milk_silo with [status - active]) / 119
(goal - sep_milk_ev_silos with [status - active]) / 117
(activity - sep_milk_ev_silos with [current_goals -
                                    [E1ST,E1PR,E1SS,E1SI,E1LN,E1SL]]) / 118
is_on(E1ST,[E1ST,E1PR,E1SS,E1SI,E1LN,E1SL]) / 0
(process_route - E1ST with [current_input - [MSL1,MSL3]]) / 106
call(filter_choices([MSL1,MSL3],[product - milk],[MSL1,MSL3])) / 0
([MSL1,MSL3] \= []) / 0
call(filter_choices([MSL1,MSL3],[current_product_grade - 2],
                                                    [MSL1])) / 0
([MSL1] \= []) / 0
/* the rule which has fired is looking for a milk silo with Grade 2 milk and calls
select_element on Milk Silo 1 */
Selecting Element: MSL1
/* this instigates binary constraint check */
Intermediate_choices: /* choices remaining */
[[EVS1,EVS2],
[ELN1],
[SKLN],
[SIP1,SIP2],
[SPS1,SPS2,SPS3],
[SEP1,SEP2,SEP3],
[RLN1,RLN2],
[MSL1]]              /* the selected element */
Remaining_choices: /* results of the binary constraint check */
[[EVS1,EVS2],
[ELN1],
[SKLN],
[SIP1,SIP2],         /* No additional constraints introduced
[SPS1,SPS2,SPS3],    through the selection of MSL1 */
[SEP1,SEP2,SEP3],
[RLN1,RLN2],
[MSL1]]
Changes: _0005CAF4
/* second rule action introduces new context
to choose a separator */
adding - (goal - choose_separator with [status - active])
rule fired: evs3a
```

```
conflict set: evs2 evs2a evs1
rule selected: evs2
/* rule conditions matched */
(goal - choose_separator with [status - active]) / 134
(goal - sep_milk_ev_silos with [status - active]) / 117
(activity - sep_milk_ev_silos with [current_goals -
                                [E1ST,E1PR,E1SS,E1SI,E1LN,E1SL]]) / 118
is_on(E1PR,[E1ST,E1PR,E1SS,E1SI,E1LN,E1SL]) / 0
(process_route - E1PR with [current_processor -
                                              [SEP1,SEP2,SEP3]]) / 125
call(filter_choices_NOT([SEP1,SEP2,SEP3],[descriptor - SEP3],
                                              [SEP1,SEP2,SEP3])) / 0
([SEP1,SEP2] \= []) / 0
/* call to select_element Separator 1 */
Selecting Element: SEP1
Intermediate_choices: /* choices remaining */
[[EVS1,EVS2],
[ELN1],
[SKLN],
[SIP1,SIP2],
[SPS1,SPS2,SPS3],
[SEP1],                              /* SEP1 is selected element */
[RLN1,RLN2],
[MSL1]]
Remaining_choices: /* results of binary check */
[[EVS1,EVS2],
[ELN1],
[SKLN],
[SIP1,SIP2],
[SPS1],                    /* SPS2, SPS3 lost as choices */
[SEP1],
[RLN1],                    /* RLN2 lost as choice */
[MSL1]]
Changes: true              /* a Change in the route is posted */
Remaining_choices: /* constraint propagation through
[[EVS1,EVS2],                second binary check */
[ELN1],
[SKLN],
[SIP1,SIP2],
[SPS1],
[SEP1],
[RLN1],
[MSL1]]
Changes: _0003D69A                /* No Change is posted
                                     constraint checking exits */
/* New context to choose a skim line input */
adding - (goal - skim_line_input with [status - active])
rule fired: evs2

conflict set: evs5a evs1
rule selected: evs5a
/* rule conditions matched */
(goal - skim_line_input with [status - active]) / 149
(goal - sep_milk_ev_silos with [status - active]) / 117
(activity - sep_milk_ev_silos with [current_goals -
                                [E1ST,E1PR,E1SS,E1SI,E1LN,E1SL]]) / 118
is_on(E1SI,[E1ST,E1PR,E1SS,E1SI,E1LN,E1SL]) / 0
(process_route - E1SI with [current_processor - [SIP1,SIP2]]) / 144
```

```
/* element selected on default criterion of "first in list" */
Selecting Element: SIP1
Intermediate_choices:
[[EVS1,EVS2],
[ELN1],
[SKLN],
[SIP1],                                  /* selected element */
[SPS1],
[SEP1],
[RLN1],
[MSL1]]
Remaining_choices:
[[EVS1,EVS2],
[ELN1],
[SKLN],
[SIP1],
[SPS1],
[SEP1],
[RLN1],
[MSL1]]
Changes: _000ADA4E /* No change is posted */
/* New context choose an evaporator silo */
adding - (goal - evap_silo with [status - active])
rule fired: evs5a

conflict set: evs5 evs1
rule selected: evs5
/* rule conditions matched */
(goal - evap_silo with [status - active]) / 164
(goal - sep_milk_ev_silos with [status - active]) / 117
(activity - sep_milk_ev_silos with [current_goals -
                                    [E1ST,E1PR,E1SS,E1SI,E1LN,E1SL]]) / 118
is_on(E1SL,[E1ST,E1PR,E1SS,E1SI,E1LN,E1SL]) / 0
(process_route - E1SL with [current_output - [EVS1,EVS2]]) / 163
/* Evaporator silo 2 chosen on basis of current contents */
Selecting Element: EVS2
Intermediate_choices:
[[EVS2],                                 /* selected element */
[ELN1],
[SKLN],
[SIP1],
[SPS1],
[SEP1],
[RLN1],
[MSL1]]
Remaining_choices:
[[EVS2],
[ELN1],
[SKLN],
[SIP1],
[SPS1],
[SEP1],
[RLN1],
[MSL1]]
Changes: _00095C64                       /* No Changes posted. N.B. any element
                                            lost at this point it would indicate
                                            a problem with system data because the
                                            configuration is now complete with a
                                            single plant item in each position */
rule fired: evs5
/* the rule has removed all contexts associated with this route configuration so the
conflict set is now empty and the inference cycle finishes */
conflict set:
inference cycle done
```

```
/* (Point 10 ) */
daemon as_update_pr on process route: E1ST
daemon sequence_update
updating the sequence with: E1ST
daemon as_update_pr on process route: E1PR
daemon sequence_update
updating the sequence with: E1PR
daemon as_update_pr on process route: E1SS
daemon sequence_update
updating the sequence with: E1SS
daemon as_update_pr on process route: E1SI
daemon sequence_update
updating the sequence with: E1SI
daemon as_update_pr on process route: E1LN
daemon sequence_update
updating the sequence with: E1LN
daemon as_update_pr on process route: E1SL
daemon sequence_update
updating the sequence with: E1SL
/* (Point 11) */

/* second call to configure activity for skimmed milk supply to the evaporator
department */
Configuration of activity: sep_milk_ev_silos
Goallist:
[E2SL,E2SS,E2PR,E2ST]
Ordered_goals:
[E2ST,E2PR,E2SS,E2SL]
Initial_choices: /* choices defined in route data */
[[EVS1,EVS2],
[ELN2],
[SPS1,SPS2,SPS3],
[SEP1,SEP2,SEP3],
[RLN1,RLN2],
[MSL1,MSL2,MSL3,MSL4]]
Intermediate_choices: /* results of unary constraint check */
[[EVS1,EVS2],
[ELN2],
[SPS1,SPS2,SPS3],
[SEP1,SEP2,SEP3],
[RLN1,RLN2],
[MSL1,MSL3]]            /* MSL2 and MSL4 lost on the basis of being
                           empty */
Remaining_choices: /* results of binary constraint check */
[[EVS1,EVS2],
[ELN2],
[SPS2,SPS3],            /* SPS1 is lost as a choice because it
[SEP1,SEP2,SEP3],          is already connected to ELN1 and
[RLN1,RLN2],               cannot make another connection to
[MSL1,MSL3]]               ELN2 */
Changes: true           /* A Change is posted so constraint propagation
                           must occur */
/* Although SPS1 was lost on this pass it's immediate input SEP1 was not even
though it is already in use in another route and can only make a connection for
skim routing to SPS1. This is because the direction of the binary constraint check
passes up the route from MSL1 to EVS2. Therefore when SEP1 was being checked
against potential output connections SPS1 still existed as a choice. This shows the
problem with only making a single pass through the network as described by
Mackworth that changes later in the network can make the parts checked earlier
inconsistent again. Therefore constraint propagation is required. */
```

```
Remaining_choices: /* results of second binary check */
[[EVS1,EVS2],
[ELN2],
[SPS2,SPS3],
[SEP2,SEP3],                    /* SEP1 is lost on this pass as it now no
[RLN1,RLN2],                      longer has a potential output choice */
[MSL1,MSL3]]
Changes: true                   /* A Change is posted so constraint propagation
                                   must occur again */
Remaining_choices: /* Results of third binary check */
[[EVS1,EVS2],
[ELN2],
[SPS2,SPS3],
[SEP2,SEP3],
[RLN1,RLN2],
[MSL1,MSL3]]
Changes: _0007C872              /* No more Changes */
Final_choices: /* Currently available choices for this route */
[[EVS1,EVS2],
[ELN2],
[SPS2,SPS3],
[SEP2,SEP3],
[RLN1,RLN2],
[MSL1,MSL3]]
/* configure_activity instigates preferential element selection */
conflict set: evs7
rule selected: evs7
/* rule conditions matched */
(goal - sep_milk_ev_silos with [status - active]) / 258
(activity - sep_milk_ev_silos with [current_goals -
                                    [E2ST,E2PR,E2SS,E2SL]]) / 259
is_on(E2PR,[E2ST,E2PR,E2SS,E2SL]) / 0

adding - (goal - choose_milk_silo with [status - active])
rule fired: evs7

conflict set: evs9b evs9a evs7
rule selected: evs9a
/* rule conditions matched */
(goal - choose_milk_silo with [status - active]) / 260
(goal - sep_milk_ev_silos with [status - active]) / 258
(activity - sep_milk_ev_silos with [current_goals -
                                    [E2ST,E2PR,E2SS,E2SL]]) / 259
is_on(E2ST,[E2ST,E2PR,E2SS,E2SL]) / 0
(process_route - E2ST with [current_input - [MSL1,MSL3]]) / 251
call(filter_choices([MSL1,MSL3],[current_product_grade - 2],
                                                          [MSL1])) / 0
([MSL1] \= []) / 0
/* Milk Silo 1 is again chosen as the source element for this
route */
Selecting Element: MSL1
Intermediate_choices: /* initial route choices remaining */
[[EVS1,EVS2],
[ELN2],
[SPS2,SPS3],
[SEP2,SEP3],
[RLN1,RLN2],
[MSL1]]                         /* element selected */
```

```
Remaining_choices: /* results of first binary constraint check */
[[EVS1,EVS2],
[ELN2],
[SPS2],                    /* SPS3 lost */
[SEP2],                    /* SEP3 lost */
[RLN1],                    /* RLN2 lost */
[MSL1]]
Changes: true              /* A change is posted */
/* On this first binary constraint check three items were lost because they all relied
on each other for a connection, but were lost from the lower positions of the route
upwards unlike the situation before with SPS1 and SEP1. In this case RLN2 was
lost because it could not take an input from MSL1, and not other silo choices now
exist. SEP3 was then lost because it can only take an input connection from RLN2,
and SPS3 was lost because it can only take an input from SEP3. */
Remaining_choices: /* result of second binary constraint check */
[[EVS1,EVS2],
[ELN2],
[SPS2],
[SEP2],
[RLN1],
[MSL1]]
Changes: _0003EB8A      /* No Change posted this time */
/* New context is to choose a separator */
adding - (goal - choose_separator with [status - active])
rule fired: evs9a

conflict set: evs8 evs8a evs7
rule selected: evs8
/* rule conditions matched */
(goal - choose_separator with [status - active]) / 271
(goal - sep_milk_ev_silos with [status - active]) / 258
(activity - sep_milk_ev_silos with [current_goals -
                                    [E2ST,E2PR,E2SS,E2SL]]) / 259
is_on(E2PR,[E2ST,E2PR,E2SS,E2SL]) / 0
(process_route - E2PR with [current_processor -
                                    [SEP2,SEP3]]) / 125
call(filter_choices_NOT([SEP2,SEP3],[descriptor - SEP3],
                                    [SEP2])) / 0
([SEP2] \= []) / 0

(process_route - E2PR with [current_processor - [SEP2]]) / 266
/* Separator 2 is chosen as the only available item */
Selecting Element: SEP2
Intermediate_choices:
[[EVS1,EVS2],
[ELN2],
[SPS2],
[SEP2],
[RLN1],
[MSL1]]
Remaining_choices:
[[EVS1,EVS2],
[ELN2],
[SPS2],
[SEP2],
[RLN1],
[MSL1]]
Changes: _0005AD30 /* No Change posted */
/* The selection of SEP2 did not introduce any new constraints because it was a
single plant item choice anyway */
/* Final context added is to choose an evaporator silo */
adding - (goal - evap_silo with [status - active])
rule fired: evs8
```

conflict set: evs11 evs7
rule selected: evs11
/* rule conditions matched */
(goal - evap_silo with [status - active]) / 282
(goal - sep_milk_ev_silos with [status - active]) / 258
(activity - sep_milk_ev_silos with [current_goals -
[E2ST,E2PR,E2SS,E2SL]]) / 259
is_on(E2SL,[E2ST,E2PR,E2SS,E2SL]) / 0
(process_route - E2SL with [current_output - [EVS1,EVS2]]) / 281
/* Evaporator silo 2 chosen on the same basis as route 1 */
Selecting Element: EVS2
Intermediate_choices:
[[EVS2],
[ELN2],
[SPS2],
[SEP2],
[RLN1],
[MSL1]]
Remaining_choices:
[[EVS2],
[ELN2],
[SPS2],
[SEP2],
[RLN1],
[MSL1]]
Changes: _00077818 /* No Change posted */
rule fired: evs11

conflict set:
inference cycle done
/* This route configuration is complete and the route is added to the process sequence and the plant connections are updated */
daemon as_update_pr on process route: E2ST
daemon sequence_update
updating the sequence with: E2ST
daemon as_update_pr on process route: E2PR
daemon sequence_update
updating the sequence with: E2PR
daemon as_update_pr on process route: E2SS
daemon sequence_update
updating the sequence with: E2SS
daemon as_update_pr on process route: E2SL
daemon sequence_update
updating the sequence with: E2SL

/* the rule fact1 makes it's second call to configure activity for the supply of skimmed milk to the cottage cheese department */
Configuration of activity: sep_milk_cc_silos
Goallist:
[CSLS,CSSS,CSP1,CCST]
Ordered_goals:
[CCST,CSP1,CSSS,CSLS]
/* The initial choices for this route are very similar to those of the evaporator supply routes */
Initial_choices:
[[CCS1,CCS2],
[CRLN],
[SPS1,SPS2,SPS3],
[SEP1,SEP2,SEP3],
[RLN1,RLN2],
[MSL1,MSL2,MSL3,MSL4]]

```
Intermediate_choices: /* Results of unary constraint check */
[[CCS1,CCS2],
[CRLN],
[SPS1,SPS2,SPS3],
[SEP1,SEP2,SEP3],
[RLN1,RLN2],
[MSL1,MSL3]]              /* MSL2 and MSL4 lost */
Remaining_choices: /* Result of first binary check */
[[CCS1,CCS2],
[CRLN],
[SPS3],                   /* SPS1 and SPS2 are lost */
[SEP1,SEP2,SEP3],
[RLN1,RLN2],
[MSL1,MSL3]]
Changes: true             /* A change has been posted */
/* SPS1 and SPS2 were lost because they could not make a connection to CRLN on
the basis of their current outputs. However, SEP1 and SEP2 were not lost on this
pass because of the direction of the checking */
Remaining_choices: /* Result of second binary check */
[[CCS1,CCS2],
[CRLN],
[SPS3],
[SEP3],                   /* SEP1 and SEP2 are now lost */
[RLN1,RLN2],
[MSL1,MSL3]]
Changes: true             /* A change has been posted */
Remaining_choices: /* result of third binary check *?
[[CCS1,CCS2],
[CRLN],
[SPS3],
[SEP3],
[RLN2],                   /* RLN1 is now lost because of the previous
[MSL1,MSL3]]                loss of SEP1 and SEP2 */
Changes: true             /* A change has been posted */
Remaining_choices: /* Result of fourth binary check */
[[CCS1,CCS2],
[CRLN],
[SPS3],
[SEP3],
[RLN2],
[MSL3]]                   /* MSL1 is now lost because RLN1 no longer exists */
Changes: true             /* A change has been posted */
Remaining_choices: /* Result of a fifth binary check */
[[CCS1,CCS2],
[CRLN],
[SPS3],
[SEP3],
[RLN2],
[MSL3]]
Changes: _00069C18        /* No change is posted */
Final_choices: /* Final choices remaining are
[[CCS1,CCS2],             considerably reduced */
[CRLN],
[SPS3],
[SEP3],
[RLN2],
[MSL3]]
/* The fact that five passes to propagate constraints had to be made indicates the
potential severity of connectivity constraints and also the potential risk of
unnacceptable performance time if the route structures were too big */
```

/* Preferential element selction is now instigated as before */
conflict set: ccs5 ccs1
rule selected: ccs1
/* rule conditions matched */
(goal - sep_milk_cc_silos with [status - active]) / 348
(activity - sep_milk_cc_silos with [current_goals -
[CCST,CSP1,CSSS,CSLS]]) / 349

adding - (goal - choose_milk_silo with [status - active])
rule fired: ccs1

conflict set: ccs5 ccs2a ccs1
rule selected: ccs2a
/* rule conditions matched */
(goal - choose_milk_silo with [status - active]) / 350
(goal - sep_milk_cc_silos with [status - active]) / 348
(process_route - CCST with [current_input - [MSL3]]) / 341
call(filter_choices([MSL3],[current_product_grade - 1],[MSL3])) / 0
([MSL3] \= []) / 0

Selecting Element: MSL3
Intermediate_choices:
[[CCS1,CCS2],
[CRLN],
[SPS3],
[SEP3],
[RLN2],
[MSL3]]
Remaining_choices:
[[CCS1,CCS2],
[CRLN],
[SPS3],
[SEP3],
[RLN2],
[MSL3]]
Changes:
_000ADE04
adding - (goal - choose_separator with [status - active])
rule fired: ccs2a

conflict set: ccs5 ccs3 ccs1
rule selected: ccs3
/* rule conditions matched */
(goal - choose_separator with [status - active]) / 361
(goal - sep_milk_cc_silos with [status - active]) / 348
(process_route - CSP1 with [current_processor - [SEP3]]) / 356

Selecting Element: SEP3
Intermediate_choices:
[[CCS1,CCS2],
[CRLN],
[SPS3],
[SEP3],
[RLN2],
[MSL3]]
Remaining_choices:
[[CCS1,CCS2],
[CRLN],
[SPS3],
[SEP3],
[RLN2],
[MSL3]]
Changes:
_0004F23C

adding - (goal - choose_cc_silo with [status - active])
rule fired: ccs3

conflict set: ccs5 ccs4 ccs1
rule selected: ccs4
/* rule conditions matched */
(goal - choose_cc_silo with [status - active]) / 372
(goal - sep_milk_cc_silos with [status - active]) / 348
(process_route - CSLS with [current_output - [CCS1,CCS2]]) / 371

Selecting Element: CCS2
Intermediate_choices:
[[CCS2],
[CRLN],
[SPS3],
[SEP3],
[RLN2],
[MSL3]]
Remaining_choices:
[[CCS2],
[CRLN],
[SPS3],
[SEP3],
[RLN2],
[MSL3]]
Changes:
_00076CC4
rule fired: ccs4

conflict set: ccs5 ccs1
rule selected: ccs5
/* rule conditions matched */
(goal - sep_milk_cc_silos with [status - active]) / 348

rule fired: ccs5

conflict set:
inference cycle done

daemon as_update_pr on process route: CCST
daemon sequence_update
updating the sequence with: CCST
daemon as_update_pr on process route: CSP1
daemon sequence_update
updating the sequence with: CSP1
daemon as_update_pr on process route: CSSS
daemon sequence_update
updating the sequence with: CSSS
daemon as_update_pr on process route: CSLS
daemon sequence_update
updating the sequence with: CSLS
/* (Point 12) */
rule fired: fact1

conflict set:

/* (Point 13) */
inference cycle done
send_process_sequence
/* (point 14) */

```
current_sequence:
[[CSLS,126,LK07,   ,CRLN,CCS2,   ],
[CSSS,126,LK35,   ,SPS3,LK07,   ],
[CSP1,126,LK08,   ,SEP3,LK35,   ],
[CCST,126,MSL3,   ,RLN2,LK08,   ],
[E2SL,91,LK11,   ,ELN2,EVS2,   ],
[E2SS,91,LK34,   ,SPS2,LK11,   ],
[E2PR,91,LK12,   ,SEP2,LK34,   ],
[E2ST,91,MSL1,   ,RLN1,LK12,   ],
[E1SL,91,LK09,   ,ELN1,EVS2,   ],
[E1LN,91,LK31,   ,SKLN,LK09,   ],
[E1SI,91,LK32,   ,SIP1,LK31,   ],
[E1SS,91,LK33,   ,SPS1,LK32,   ],
[E1PR,91,LK10,   ,SEP1,LK33,   ],
[E1ST,91,MSL1,   ,RLN1,LK10,   ]]
/*(point 15) */
protocol 4
line_no: 1
send_process_sequence
.
.
.
protocol 4
line_no: 14
send_process_sequence
current_sequence:
[]

/* (point 16) */
send_cip_sequence
current_sequence:
[[CSKA,15,SK1A,   ,PNIU,PNIU,   ]]
protocol 12
line_no: 1
send_cip_sequence
current_sequence:
[]
 /*(point 17) */
send_link ends
/*(point 18)*/
protocol: 5
name : CCS2
code : V06
current_rate: 0.301
activity state : 7
protocol: 9
name : CCS2
code : V06
contents : 0.000
protocol: 6
name : CRLN
code : P29
activity state : 5
daemon as_update_pr on process route: CSLS
protocol: 6
name : SPS3
code : P36
activity state : 5
daemon as_update_pr on process route: CSSS
protocol: 6
name : SEP3
code : P09
activity state : 5
daemon as_update_pr on process route: CSP1
```

/*(point 19) */
protocol: 5
name : MSL3
code : V03
current_rate: -0.334
activity state : 8
protocol: 9
name : MSL3
code : V03
contents : 210.000
protocol: 6
name : RLN2
code : P19
activity state : 5
daemon as_update_pr on process route: CCST
protocol: 5
name : EVS2
code : V08
current_rate: 0.301
activity state : 7
protocol: 9
name : EVS2
code : V08
contents : 20.000
protocol: 6
name : ELN2
code : P22
activity state : 5
daemon as_update_pr on process route: E2SL
protocol: 6
name : SPS2
code : P35
activity state : 5
daemon as_update_pr on process route: E2SS
protocol: 6
name : SEP2
code : P08
activity state : 5
daemon as_update_pr on process route: E2PR
protocol: 5
name : MSL1
code : V01
/*(point 20) */
current_rate: -0.334
activity state : 8
protocol: 9
name : MSL1
code : V01
contents : 227.300
protocol: 6
name : RLN1
code : P18
activity state : 5
daemon as_update_pr on process route: E2ST
daemon as_update_pr on process route: E1ST
protocol: 5
name : EVS2
code : V08
current_rate: 0.601
activity state : 7
protocol: 9
name : EVS2
code : V08
contents : 20.000

protocol: 6
name : ELN1
code : P21
activity state : 5
daemon as_update_pr on process route: E1SL
protocol: 6
name : SKLN
code : P20
activity state : 5
daemon as_update_pr on process route: E1LN
protocol: 6
name : SIP1
code : P30
activity state : 5
daemon as_update_pr on process route: E1SI
protocol: 6
name : SPS1
code : P34
activity state : 5
daemon as_update_pr on process route: E1SS
protocol: 6
name : SEP1
code : P07
activity state : 5
daemon as_update_pr on process route: E1PR
/*(point 21)*/
protocol: 5
name : MSL1
code : V01
current_rate: -0.668
activity state : 8
protocol: 9
name : MSL1
code : V01
contents : 227.300
protocol: 6
name : RLN1
code : P18
/*(point 22) */
activity state : 5

APPENDIX C (9.3). RESULTS OF CONFIGURATION CYCLE

/* active batches with updated activity statuse */

```
Frame: batch Instance: ev01
  Slot: temporal_activity_status -
                    [val [sep_milk_ev_silos in_progress ]
                    [evap_skim cannot_start ]]
  Slot: urgency - [val critical]
  Slot: total_float - [val -712.950]
  Slot: activity_status -
                    [val [sep_milk_ev_silos 0 [0 -440.495 ]]
                    [evap_skim 0 [272.455 10 ]]]
  Slot: ako - [val batch]
  Slot: batch_code - [val ev01]
  Slot: product - [val skim_concentrate]
Time stamp: 501

Frame: batch Instance: ev02
  Slot: temporal_activity_status -
                    [val [sep_milk_ev_silos in_progress ]
                    [evap_skim cannot_start ]]
  Slot: urgency - [val critical]
  Slot: total_float - [val -712.950]
  Slot: activity_status -
                    [val [sep_milk_ev_silos 0 [0 -440.495 ]]
                    [evap_skim 0 [272.455 10 ]]]
  Slot: ako - [val batch]
  Slot: batch_code - [val ev02]
  Slot: product - [val skim_concentrate]
Time stamp: 527

Frame: batch Instance: cc01
  Slot: temporal_activity_status -
                    [val [sep_milk_cc_silos in_progress ]
                    [past_skim_cc_vats cannot_start ]]
  Slot: urgency - [val critical]
  Slot: total_float - [val -839.156]
  Slot: activity_status -
                    [val [sep_milk_cc_silos 0 [0 -461.910 ]]
                    [past_skim_cc_vats 0 [377.246 10 ]]]
  Slot: ako - [val batch]
  Slot: batch_code - [val cc01]
  Slot: product - [val cottage_cheese]
Time stamp: 479
```

/* storage vessels with updated connections and activity statuses */

Frame: storage Instance: MSL1
Slot: time_contents_last_updated - [val 0]
Slot: contents_at_last_update - [val 227.300]
Slot: code - [val V01 ]
Slot: activity_state_attribute4 - [val 8]
Slot: current_rate_in - [val -0.668]
Slot: current_outputs - [val RLN1 RLN1 ]
Slot: ako - [val storage]
Slot: description - [val milk silo 1]
Slot: capacity - [val 227.300]
Slot: current_contents - []
Slot: current_rate_out - [val 0]
Slot: time_product_out - [val 0]
Slot: time_product_in - [val -350]
Slot: time_last_stop_fill - [val -200]
Slot: time_last_stop_empty - [val 0]
Slot: contents_status - [val 1]
Slot: system_event - [val maxcon]
Slot: configuration - [val MSL1 ---> and : [i,o]
i ---> and : [OLN1,OLN2]
o ---> or : [RLN1,RLN2] ]
Slot: work_area - [val milk_reception]
Slot: current_inputs - [val ]
Slot: current_product_grade - [val 2]
Slot: product - [val milk]
Time stamp: 532

Frame: storage Instance: MSL2
Slot: ako - [val storage]
Slot: description - [val milk silo 2]
Slot: code - [val V02]
Slot: capacity - [val 227.300]
Slot: contents_at_last_update - [val 0]
Slot: current_contents - []
Slot: time_contents_last_updated - [val 0]
Slot: current_rate_in - [val 0]
Slot: current_rate_out - [val 0]
Slot: time_product_out - [val 0]
Slot: time_product_in - [val 0]
Slot: time_last_stop_fill - [val 0]
Slot: time_last_stop_empty - [val 0]
Slot: contents_status - [val 2]
Slot: system_event - [val mincon]
Slot: configuration - [val MSL2 ---> and : [i,o]
i ---> and : [OLN1,OLN2]
o ---> or : [RLN1,RLN2] ]
Slot: work_area - [val milk_reception]
Slot: current_inputs - [val ]
Slot: current_outputs - [val ]
Slot: activity_state_attribute4 - [val 1]
Slot: current_product_grade - [val 0]
Slot: product - [val milk]
Time stamp: 1

Frame: storage Instance: MSL3
Slot: time_contents_last_updated - [val 0]
Slot: contents_at_last_update - [val 210.000]
Slot: code - [val V03 ]
Slot: activity_state_attribute4 - [val 8]
Slot: current_rate_in - [val -0.334]
Slot: current_outputs - [val RLN2 ]
Slot: ako - [val storage]
Slot: description - [val milk silo 3]
Slot: capacity - [val 227.300]
Slot: current_contents - []
Slot: current_rate_out - [val 0]
Slot: time_product_out - [val 0]
Slot: time_product_in - [val -200]
Slot: time_last_stop_fill - [val 0]
Slot: time_last_stop_empty - [val 0]
Slot: contents_status - [val 0]
Slot: system_event - [val maxcon]
Slot: configuration - [val MSL3 ---> and : [i,o]
i ---> and : [OLN1,OLN2]
o ---> or : [RLN1,RLN2] ]
Slot: work_area - [val milk_reception]
Slot: current_inputs - [val ]
Slot: current_product_grade - [val 1]
Slot: product - [val milk]
Time stamp: 477

Frame: storage Instance: CCS2
Slot: time_contents_last_updated - [val 0]
Slot: contents_at_last_update - [val 0.000]
Slot: code - [val V06 ]
Slot: time_product_in - [val 0]
Slot: activity_state_attribute4 - [val 7]
Slot: current_rate_in - [val 0.301]
Slot: current_inputs - [val CRLN ]
Slot: ako - [val storage]
Slot: description - [val cc skim silo 2]
Slot: capacity - [val 90.920]
Slot: current_contents - []
Slot: current_rate_out - [val 0]
Slot: time_product_out - [val 0]
Slot: time_last_stop_fill - [val 0]
Slot: time_last_stop_empty - [val 0]
Slot: contents_status - [val 0]
Slot: system_event - [val mincon]
Slot: configuration - [val CCS2 ---> and : [i,o]
i ---> and : [CRLN]
o ---> and : [CPST] ]
Slot: work_area - [val cottage_cheese]
Slot: current_outputs - [val ]
Slot: current_product_grade - [val 1]
Slot: product - [val skim]
Time stamp: 462

```
Frame: storage Instance: EVS2
  Slot: time_contents_last_updated - [val 0]
  Slot: contents_at_last_update - [val 20.000]
  Slot: code - [val V08 ]
  Slot: activity_state_attribute4 - [val 7]
  Slot: current_rate_in - [val 0.601]
  Slot: current_inputs - [val ELN2 ELN1 ]
  Slot: ako - [val storage]
  Slot: description - [val evaporator skim silo 2]
  Slot: capacity - [val 90.920]
  Slot: current_contents - []
  Slot: current_rate_out - [val 0]
  Slot: time_product_out - [val 0]
  Slot: time_product_in - [val -50]
  Slot: time_last_stop_fill - [val 0]
  Slot: time_last_stop_empty - [val 0]
  Slot: contents_status - [val 0]
  Slot: system_event - [val mincon]
  Slot: configuration - [val EVS2 ---> and : [i,o]
                                       i ---> and : [ELN1,ELN2]
                                       o ---> or : [EVR1,EVR2] ]
  Slot: work_area - [val evaporators]
  Slot: current_outputs - [val ]
  Slot: current_product_grade - [val 2]
  Slot: product - [val skim]
Time stamp: 509

/* semi-continuous process plant items with updated activity
   status and connections */
/* separators and skimmed milk outputs */

Frame: continuous Instance: SEP1
  Slot: activity_state_attribute4 - [val 5]
  Slot: code - [val P07 ]
  Slot: current_process_route - [val E1PR ]
  Slot: current_outputs - [val SPS1 ]
  Slot: current_inputs - [val RLN1 ]
  Slot: ako - [val continuous]
  Slot: descriptor - [val SEP1]
  Slot: configuration - [val SEP1 ---> and : [i,o]
                                       i ---> and : [RLN1]
                                       o ---> and : [SPS1,SPC1] ]
  Slot: work_area - [val dairy]
  Slot: current_rate - [val 0.334 / klitres_min]
Time stamp: 529

Frame: process_line Instance: SPS1
  Slot: activity_state_attribute4 - [val 5]
  Slot: code - [val P34 ]
  Slot: current_process_route - [val E1SS ]
  Slot: current_outputs - [val SIP1 ]
  Slot: current_inputs - [val SEP1 ]
  Slot: ako - [val process_line]
  Slot: descriptor - [val SPS1]
  Slot: configuration - [val SPS1 ---> and : [i,o]
                                   i ---> and : [SEP1]
                                   o ---> or : [SIP1,SIP2,CRLN,ELN2] ]
  Slot: work_area - [val dairy]
  Slot: current_rate - [val 0]
Time stamp: 525
```

Frame: continuous Instance: SEP2
Slot: activity_state_attribute4 - [val 5]
Slot: code - [val P08 ]
Slot: current_process_route - [val E2PR ]
Slot: current_outputs - [val SPS2 ]
Slot: current_inputs - [val RLN1 ]
Slot: ako - [val continuous]
Slot: descriptor - [val SEP2]
Slot: configuration - [val SEP2 ---> and : [i,o]
i ---> and : [RLN1]
o ---> and : [SPS2,SPC2] ]
Slot: work_area - [val dairy]
Slot: current_rate - [val 0.334 / klitre_min]
Time stamp: 496

Frame: process_line Instance: SPS2
Slot: activity_state_attribute4 - [val 5]
Slot: code - [val P35 ]
Slot: current_process_route - [val E2SS ]
Slot: current_outputs - [val ELN2 ]
Slot: current_inputs - [val SEP2 ]
Slot: ako - [val process_line]
Slot: descriptor - [val SPS2]
Slot: configuration - [val SPS2 ---> and : [i,o]
i ---> and : [SEP2]
o ---> or : [SIP1,SIP2,CRLN,ELN2] ]
Slot: work_area - [val dairy]
Slot: current_rate - [val 0]
Time stamp: 492

Frame: continuous Instance: SEP3
Slot: activity_state_attribute4 - [val 5]
Slot: code - [val P09 ]
Slot: current_process_route - [val CSP1 ]
Slot: current_outputs - [val SPS3 ]
Slot: current_inputs - [val RLN2 ]
Slot: ako - [val continuous]
Slot: descriptor - [val SEP3]
Slot: configuration - [val SEP3 ---> and : [i,o]
i ---> and : [RLN2]
o ---> and : [SPS3,SPC3] ]
Slot: work_area - [val dairy]
Slot: current_rate - [val 0.334 / klitres_min]
Time stamp: 474

Frame: process_line Instance: SPS3
Slot: activity_state_attribute4 - [val 5]
Slot: code - [val P36 ]
Slot: current_process_route - [val CSSS ]
Slot: current_outputs - [val CRLN ]
Slot: current_inputs - [val SEP3 ]
Slot: ako - [val process_line]
Slot: descriptor - [val SPS3]
Slot: configuration - [val SPS3 ---> and : [i,o]
i ---> and : [SEP3]
o ---> or : [SIP1,SIP2,CRLN,ELN2] ]
Slot: work_area - [val dairy]
Slot: current_rate - [val 0]
Time stamp: 470

/* process lines with updated activity status and connections */

Frame: process_line Instance: ELN1
Slot: activity_state_attribute4 - [val 5]
Slot: code - [val P21 ]
Slot: current_process_route - [val E1SL ]
Slot: current_outputs - [val EVS2 ]
Slot: current_inputs - [val SKLN ]
Slot: ako - [val process_line]
Slot: descriptor - [val ELN1]
Slot: configuration - [val ELN1 ---> and : [i,o]
i ---> or : [SKM1,SKM2,SKM3,SKLN]
o ---> or : [EVS1,EVS2] ]
Slot: work_area - [val evaporators]
Slot: current_rate - [val 0]
Time stamp: 513

Frame: process_line Instance: ELN2
Slot: activity_state_attribute4 - [val 5]
Slot: code - [val P22 ]
Slot: current_process_route - [val E2SL ]
Slot: current_outputs - [val EVS2 ]
Slot: current_inputs - [val SPS2 ]
Slot: ako - [val process_line]
Slot: descriptor - [val ELN2]
Slot: configuration - [val ELN2 ---> and : [i,o]
i ---> or : [SPS1,SPS2,SPS3]
o ---> or : [EVS1,EVS2] ]
Slot: work_area - [val evaporators]
Slot: current_rate - [val 0]
Time stamp: 488

Frame: process_line Instance: RLN1
Slot: activity_state_attribute4 - [val 5]
Slot: code - [val P18 ]
Slot: current_process_route - [val E2ST E1ST ]
Slot: current_outputs - [val SEP2 SEP1 ]
Slot: current_inputs - [val MSL1 MSL1 ]
Slot: ako - [val process_line]
Slot: descriptor - [val RLN1]
Slot: configuration - [val RLN1 ---> and : [i,o]
i ---> or : [MSL1,MSL2,MSL3,MSL4]
o ---> and : [SEP1,SEP2] ]
Slot: work_area - [val dairy]
Slot: current_rate - [val 0]
Time stamp: 533

Frame: process_line Instance: RLN2
Slot: activity_state_attribute4 - [val 5]
Slot: code - [val P19 ]
Slot: current_process_route - [val CCST ]
Slot: current_outputs - [val SEP3 ]
Slot: current_inputs - [val MSL3 ]
Slot: ako - [val process_line]
Slot: descriptor - [val RLN2]
Slot: configuration - [val RLN1 ---> and : [i,o]
i ---> or : [MSL1,MSL2,MSL3,MSL4]
o ---> and : [SEP3] ]
Slot: work_area - [val dairy]
Slot: current_rate - [val 0]
Time stamp: 481

```
Frame: process_line Instance: CRLN
  Slot: activity_state_attribute4 - [val 5]
  Slot: code - [val P29 ]
  Slot: current_process_route - [val CSLS ]
  Slot: current_outputs - [val CCS2 ]
  Slot: current_inputs - [val SPS3 ]
  Slot: ako - [val process_line]
  Slot: descriptor - [val CRLN]
  Slot: configuration - [val CRLN ---> and : [i,o]
                                       i ---> or : [SPS1,SPS2,SPS3]
                                       o ---> or : [CCS1,CCS2] ]
  Slot: work_area - [val cottage_cheese]
  Slot: current_rate - [val 0]
Time stamp: 466

Frame: process_line Instance: SKLN
  Slot: activity_state_attribute4 - [val 5]
  Slot: code - [val P20 ]
  Slot: current_process_route - [val E1LN ]
  Slot: current_outputs - [val ELN1 ]
  Slot: current_inputs - [val SIP1 ]
  Slot: ako - [val process_line]
  Slot: descriptor - [val SKLN]
  Slot: configuration - [val SKLN ---> and : [i,o]
                                  i ---> and : [SIP1,SIP2]
                                  o ---> or : [SKM1,SKM2,SKM3,ELN1] ]
  Slot: work_area - [val dairy]
  Slot: current_rate - [val 0]
Time stamp: 517

Frame: process_line Instance: SIP1
  Slot: activity_state_attribute4 - [val 5]
  Slot: code - [val P30 ]
  Slot: current_process_route - [val E1SI ]
  Slot: current_outputs - [val SKLN ]
  Slot: current_inputs - [val SPS1 ]
  Slot: ako - [val process_line]
  Slot: descriptor - [val SIP1]
  Slot: configuration - [val SIP1 ---> and : [i,o]
                                       i ---> or : [SPS1,SPS2,SPS3]
                                       o ---> and : [SKLN] ]
  Slot: work_area - [val dairy]
  Slot: current_rate - [val 0]
Time stamp: 521

/* configured process routes */
/* sub-routes making up evaporator skimmed milk supply route 1 */
```

Frame: process_route Instance: E1SL
Slot: activity_state_attribute4 - [val 5]
Slot: current_input - [val LK09 ]
Slot: remaining_batchsize - [val 91]
Slot: current_batch - [val ev02]
Slot: current_output - [val EVS2 ]
Slot: current_processor - [val ELN1 ]
Slot: ako - [val process_route]
Slot: description - [val skim to evap silos via evap line 1]
Slot: links_up - [val ]
Slot: links_down - [val E1LN ]
Slot: output_choices - [val EVS1 EVS2 ]
Slot: processor_choices - [val ELN1 ]
Slot: input_choices - [val links_down LK09 ]
Slot: current_yield - [val 0.900]
Slot: input_factor - [val 1]
Slot: current_activity - [val sep_milk_ev_silos]
Time stamp: 512

Frame: process_route Instance: E1LN
Slot: activity_state_attribute4 - [val 5]
Slot: current_input - [val LK31 ]
Slot: current_output - [val LK09 ]
Slot: remaining_batchsize - [val 91]
Slot: current_batch - [val ev02]
Slot: current_processor - [val SKLN ]
Slot: ako - [val process_route]
Slot: description - [val skim line routing]
Slot: links_up - [val E1SL ]
Slot: links_down - [val E1SI ]
Slot: output_choices - [val links_up LK09 ]
Slot: processor_choices - [val SKLN ]
Slot: input_choices - [val links_down LK31 ]
Slot: current_yield - [val 1]
Slot: input_factor - [val 1]
Slot: current_activity - [val sep_milk_ev_silos]
Time stamp: 516

Frame: process_route Instance: E1SI
Slot: activity_state_attribute4 - [val 5]
Slot: current_input - [val LK32 ]
Slot: current_output - [val LK31 ]
Slot: remaining_batchsize - [val 91]
Slot: current_batch - [val ev02]
Slot: current_processor - [val SIP1 ]
Slot: ako - [val process_route]
Slot: description - [val skim line inputs]
Slot: links_up - [val E1LN ]
Slot: links_down - [val E1SS ]
Slot: output_choices - [val links_up LK31 ]
Slot: processor_choices - [val SIP1 SIP2 ]
Slot: input_choices - [val links_down LK32 ]
Slot: current_yield - [val 1]
Slot: input_factor - [val 1]
Slot: current_activity - [val sep_milk_ev_silos]
Time stamp: 520

Frame: process_route Instance: E1SS
Slot: activity_state_attribute4 - [val 5]
Slot: current_input - [val LK33 ]
Slot: current_output - [val LK32 ]
Slot: remaining_batchsize - [val 91]
Slot: current_batch - [val ev02]
Slot: current_processor - [val SPS1 ]
Slot: ako - [val process_route]
Slot: description - [val skim line routing]
Slot: links_up - [val E1SI ]
Slot: links_down - [val E1PR ]
Slot: output_choices - [val links_up LK32 ]
Slot: processor_choices - [val SPS1 SPS2 SPS3 ]
Slot: input_choices - [val links_down LK33 ]
Slot: current_yield - [val 1]
Slot: input_factor - [val 1]
Slot: current_activity - [val sep_milk_ev_silos]
Time stamp: 524

Frame: process_route Instance: E1PR
Slot: activity_state_attribute4 - [val 5]
Slot: current_input - [val LK10 ]
Slot: current_output - [val LK33 ]
Slot: remaining_batchsize - [val 91]
Slot: current_batch - [val ev02]
Slot: current_processor - [val SEP1 ]
Slot: ako - [val process_route]
Slot: description - [val separators to skim outputs]
Slot: links_up - [val E1SS ]
Slot: links_down - [val E1ST ]
Slot: output_choices - [val links_up LK33 ]
Slot: processor_choices - [val SEP1 SEP2 SEP3 ]
Slot: input_choices - [val links_down LK10 ]
Slot: current_yield - [val 0.900]
Slot: input_factor - [val 1]
Slot: current_activity - [val sep_milk_ev_silos]
Time stamp: 528

Frame: process_route Instance: E1ST
Slot: activity_state_attribute4 - [val 5]
Slot: current_output - [val LK10 ]
Slot: remaining_batchsize - [val 91]
Slot: current_batch - [val ev02]
Slot: current_processor - [val RLN1 ]
Slot: current_input - [val MSL1 ]
Slot: ako - [val process_route]
Slot: description - [val milk storage to seps]
Slot: links_up - [val E1PR ]
Slot: links_down - [val ]
Slot: output_choices - [val links_up LK10 ]
Slot: processor_choices - [val RLN1 RLN2 ]
Slot: input_choices - [val MSL1 MSL2 MSL3 MLS4 ]
Slot: current_yield - [val 1]
Slot: input_factor - [val 1]
Slot: current_activity - [val sep_milk_ev_silos]
Time stamp: 505

/* sub-routes for evaporator supply route 2 */

Frame: process_route Instance: E2SL
  Slot: activity_state_attribute4 - [val 5]
  Slot: current_input - [val LK11 ]
  Slot: remaining_batchsize - [val 91]
  Slot: current_batch - [val ev01]
  Slot: current_output - [val EVS2 ]
  Slot: current_processor - [val ELN2 ]
  Slot: ako - [val process_route]
  Slot: description - [val skim to evap silos via evap line 2]
  Slot: links_up - [val ]
  Slot: links_down - [val E2SS ]
  Slot: output_choices - [val EVS1 EVS2 ]
  Slot: processor_choices - [val ELN2 ]
  Slot: input_choices - [val links_down LK11 ]
  Slot: current_yield - [val 0.900]
  Slot: input_factor - [val 1]
  Slot: current_activity - [val sep_milk_ev_silos]
Time stamp: 487

Frame: process_route Instance: E2SS
  Slot: activity_state_attribute4 - [val 5]
  Slot: current_input - [val LK34 ]
  Slot: current_output - [val LK11 ]
  Slot: remaining_batchsize - [val 91]
  Slot: current_batch - [val ev01]
  Slot: current_processor - [val SPS2 ]
  Slot: ako - [val process_route]
  Slot: description - [val sep skim outputs]
  Slot: links_up - [val E2SL ]
  Slot: links_down - [val E2PR ]
  Slot: output_choices - [val links_up LK11 ]
  Slot: processor_choices - [val SPS1 SPS2 SPS3 ]
  Slot: input_choices - [val links_down LK34 ]
  Slot: current_yield - [val 1]
  Slot: input_factor - [val 1]
  Slot: current_activity - [val sep_milk_ev_silos]
Time stamp: 491

Frame: process_route Instance: E2PR
  Slot: activity_state_attribute4 - [val 5]
  Slot: current_input - [val LK12 ]
  Slot: current_output - [val LK34 ]
  Slot: remaining_batchsize - [val 91]
  Slot: current_batch - [val ev01]
  Slot: current_processor - [val SEP2 ]
  Slot: ako - [val process_route]
  Slot: description - [val separators to evap line 2]
  Slot: links_up - [val E2SS ]
  Slot: links_down - [val E2ST ]
  Slot: output_choices - [val links_up LK34 ]
  Slot: processor_choices - [val SEP1 SEP2 SEP3 ]
  Slot: input_choices - [val links_down LK12 ]
  Slot: current_yield - [val 0.900]
  Slot: input_factor - [val 1]
  Slot: current_activity - [val sep_milk_ev_silos]
Time stamp: 495

Frame: process_route Instance: E2ST
Slot: activity_state_attribute4 - [val 5]
Slot: current_output - [val LK12 ]
Slot: remaining_batchsize - [val 91]
Slot: current_batch - [val ev01]
Slot: current_processor - [val RLN1 ]
Slot: current_input - [val MSL1 ]
Slot: ako - [val process_route]
Slot: description - [val milk storage to seps]
Slot: links_up - [val E2PR ]
Slot: links_down - [val ]
Slot: output_choices - [val links_up LK12 ]
Slot: processor_choices - [val RLN1 RLN2 ]
Slot: input_choices - [val MSL1 MSL2 MSL3 MLS4 ]
Slot: current_yield - [val 1]
Slot: input_factor - [val 1]
Slot: current_activity - [val sep_milk_ev_silos]
Time stamp: 502

/* cottage cheese skimmed milk supply route */

Frame: process_route Instance: CSLS
Slot: activity_state_attribute4 - [val 5]
Slot: current_input - [val LK07 ]
Slot: remaining_batchsize - [val 126]
Slot: current_batch - [val cc01]
Slot: current_output - [val CCS2 ]
Slot: current_processor - [val CRLN ]
Slot: ako - [val process_route]
Slot: description - [val cc line to silos]
Slot: links_up - [val ]
Slot: links_down - [val CSSS ]
Slot: output_choices - [val CCS1 CCS2 ]
Slot: processor_choices - [val CRLN ]
Slot: input_choices - [val links_down LK07 ]
Slot: current_yield - [val 0.900]
Slot: input_factor - [val 1]
Slot: current_activity - [val sep_milk_cc_silos]
Time stamp: 465

Frame: process_route Instance: CSSS
Slot: activity_state_attribute4 - [val 5]
Slot: current_input - [val LK35 ]
Slot: current_output - [val LK07 ]
Slot: remaining_batchsize - [val 126]
Slot: current_batch - [val cc01]
Slot: current_processor - [val SPS3 ]
Slot: ako - [val process_route]
Slot: description - [val sep skim outputs]
Slot: links_up - [val CSLS ]
Slot: links_down - [val CSP1 ]
Slot: output_choices - [val links_up LK07 ]
Slot: processor_choices - [val SPS1 SPS2 SPS3 ]
Slot: input_choices - [val links_down LK35 ]
Slot: current_yield - [val 1]
Slot: input_factor - [val 1]
Slot: current_activity - [val sep_milk_cc_silos]
Time stamp: 469

Frame: process_route Instance: CSP1
Slot: activity_state_attribute4 - [val 5]
Slot: current_input - [val LK08 ]
Slot: current_output - [val LK35 ]
Slot: remaining_batchsize - [val 126]
Slot: current_batch - [val cc01]
Slot: current_processor - [val SEP3 ]
Slot: ako - [val process_route]
Slot: description - [val separators to sep line]
Slot: links_up - [val CSSS ]
Slot: links_down - [val CCST ]
Slot: output_choices - [val links_up LK35 ]
Slot: processor_choices - [val SEP1 SEP2 SEP3 ]
Slot: input_choices - [val links_down LK08 ]
Slot: current_yield - [val 0.900]
Slot: input_factor - [val 1]
Slot: current_activity - [val sep_milk_cc_silos]
Time stamp: 473

Frame: process_route Instance: CCST
Slot: activity_state_attribute4 - [val 5]
Slot: current_output - [val LK08 ]
Slot: remaining_batchsize - [val 126]
Slot: current_batch - [val cc01]
Slot: current_processor - [val RLN2 ]
Slot: current_input - [val MSL3 ]
Slot: ako - [val process_route]
Slot: description - [val milk silos to separators for cc skim]
Slot: links_up - [val CSP1 ]
Slot: links_down - [val ]
Slot: output_choices - [val links_up LK08 ]
Slot: processor_choices - [val RLN1 RLN2 ]
Slot: input_choices - [val MSL1 MSL2 MSL3 MSL4 ]
Slot: current_yield - [val 1]
Slot: input_factor - [val 1]
Slot: current_activity - [val sep_milk_cc_silos]
Time stamp: 480

## BATCH SEQUENCE

| PROCESS STATUS | ROUTES ROUTE NAME | INITIAL BATCH | REM/NG BATCH | START TIMED | LINKED TIME | PRIORITY | INDEX 1 | INDEX 2 |
|---|---|---|---|---|---|---|---|---|
| I | strt | 0.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | CSLS | 126.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | CSSS | 126.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | CSP1 | 126.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | CCST | 126.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2SL | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2SS | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2PR | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2ST | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1SL | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1LN | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1SI | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1SS | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1PR | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1ST | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B1SL | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B1ML | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B1VH | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B2SL | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B2ML | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B2VH | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B3SL | 481.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B3ML | 481.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B3VH | 481.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B1SL | 481.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B1ML | 481.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B1VH | 481.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B2SL | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B2ML | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B2VH | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2SL | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2SS | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2PR | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2ST | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1SL | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1LN | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1SI | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1SS | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1PR | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1ST | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| R | EPR2 | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| R | EPR1 | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| R | CSLS | 25.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| R | CSSS | 25.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| R | CSP1 | 25.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| R | CCST | 25.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| CIP ROUTES | | | | | | | | |
| I | stcp | 0.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | CSKA | 0.0 | 15.0 | 0.0 | 0.0 | 1 | 0 | 0 |

## PROCESS ROUTE SCHEDULE

| | ROUTE /CODE | START TIME | STOP TIME | REM/NG BATCH | ELEMENT 1 | CONTENTS 1 | ELEMENT 2 | ELEMENT 3 | CONTENTS 2 | SAVE /REST |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CSLS | 0.0 | 126.0 | LK07 | 0.0 | CRLN | CCS2 | | 0.0 | 0 |
| | R37 | | 302.5 | 25.0 | | 0.0 | P29 | V06 | 90.9 | 0 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | CSSS | 0.0 | | 126.0 | LK35 | 0.0 | SPS3 | LK07 | 0.0 |
| | R49 | | 302.5 | 25.0 | | 0.0 | P36 | | 0.0 |
| 3 | CSP1 | 0.0 | | 126.0 | LK08 | 0.0 | SEP3 | LK35 | 0.0 |
| | R38 | | 302.5 | 25.0 | | 0.0 | P09 | | 0.0 |
| 4 | CCST | 0.0 | | 126.0 | MSL3 | 210.0 | RLN2 | LK08 | 0.0 |
| | R39 | | 302.5 | 25.0 | V03 | 109.0 | P19 | | 0.0 |
| 5 | E2SL | 0.0 | | 91.0 | LK11 | 0.0 | ELN2 | EVS2 | 20.0 |
| | R22 | | 118.0 | 51.6 | | 0.0 | P22 | V08 | 90.9 |
| 6 | E2SS | 0.0 | | 91.0 | LK34 | 0.0 | SPS2 | LK11 | 0.0 |
| | R48 | | 118.0 | 51.6 | | 0.0 | P35 | | 0.0 |
| 7 | E2PR | 0.0 | | 91.0 | LK12 | 0.0 | SEP2 | LK34 | 0.0 |
| | R23 | | 118.0 | 51.6 | | 0.0 | P08 | | 0.0 |
| 8 | E2ST | 0.0 | | 91.0 | MSL1 | 227.3 | RLN1 | LK12 | 0.0 |
| | R24 | | 118.0 | 51.6 | V01 | 148.5 | P18 | | 0.0 |
| 9 | E1SL | 0.0 | | 91.0 | LK09 | 0.0 | ELN1 | EVS2 | 20.0 |
| | R41 | | 118.0 | 51.6 | | 0.0 | P21 | V08 | 90.9 |
| 10 | E1LN | 0.0 | | 91.0 | LK31 | 0.0 | SKLN | LK09 | 0.0 |
| | R47 | | 118.0 | 51.6 | | 0.0 | P20 | | 0.0 |
| 11 | E1SI | 0.0 | | 91.0 | LK32 | 0.0 | SIP1 | LK31 | 0.0 |
| | R46 | | 118.0 | 51.6 | | 0.0 | P30 | | 0.0 |
| 12 | E1SS | 0.0 | | 91.0 | LK33 | 0.0 | SPS1 | LK32 | 0.0 |
| | R45 | | 118.0 | 51.6 | | 0.0 | P34 | | 0.0 |
| 13 | E1PR | 0.0 | | 91.0 | LK10 | 0.0 | SEP1 | LK33 | 0.0 |
| | R42 | | 118.0 | 51.6 | | 0.0 | P07 | | 0.0 |
| 14 | E1ST | 0.0 | | 91.0 | MSL1 | 227.3 | RLN1 | LK10 | 0.0 |
| | R43 | | 118.0 | 51.6 | V01 | 148.5 | P18 | | 0.0 |
| 15 | B1SL | 40.0 | | 500.0 | LK02 | 0.0 | OLN1 | MSL4 | 0.0 |
| | R01 | | 81.8 | 481.0 | | 0.0 | P27 | V04 | 28.9 |
| 16 | B1ML | 40.0 | | 500.0 | LK01 | 0.0 | MLN1 | LK02 | 0.0 |
| | R02 | | 81.8 | 481.0 | | 0.0 | P24 | | 0.0 |
| 17 | B1VH | 40.0 | | 500.0 | ACCM | 19.0 | BY1A | LK01 | 0.0 |
| | R03 | | 81.8 | 481.0 | TR09 | 0.0 | P01 | | 0.0 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 18 | B2SL | 60.0 | | 500.0 | LK04 | 0.0 | OLN1 | MSL4 | 9.1 |
| | R04 | | 112.7 | 476.0 | | 0.0 | P27 | V04 | 63.7 |
| 19 | B2ML | 60.0 | | 500.0 | LK03 | 0.0 | MLN2 | LK04 | 0.0 |
| | R05 | | 112.7 | 476.0 | | 0.0 | P25 | | 0.0 |
| 20 | B2VH | 60.0 | | 500.0 | ACCM | 24.0 | BY2A | LK03 | 0.0 |
| | R06 | | 112.7 | 476.0 | TR10 | 0.0 | P02 | | 0.0 |
| 21 | B3SL | 90.0 | | 481.0 | LK06 | 0.0 | OLN1 | MSL4 | 32.7 |
| | R07 | | 117.9 | 468.3 | | 0.0 | P27 | V04 | 70.8 |
| 22 | B3ML | 90.0 | | 481.0 | LK05 | 0.0 | MLN3 | LK06 | 0.0 |
| | R08 | | 117.9 | 468.3 | | 0.0 | P26 | | 0.0 |
| 23 | B3VH | 90.0 | | 481.0 | BLKR | 12.7 | BY3A | LK05 | 0.0 |
| | R09 | | 117.9 | 468.3 | TR01 | 0.0 | P03 | | 0.0 |
| 24 | B1SL | 90.0 | | 481.0 | LK02 | 0.0 | OLN1 | MSL4 | 32.7 |
| | R01 | | 118.8 | 467.9 | | 0.0 | P27 | V04 | 71.5 |
| 25 | B1ML | 90.0 | | 481.0 | LK01 | 0.0 | MLN1 | LK02 | 0.0 |
| | R02 | | 118.8 | 467.9 | | 0.0 | P24 | | 0.0 |
| 26 | B1VH | 90.0 | | 481.0 | BLKR | 13.1 | BY1B | LK01 | 0.0 |
| | R03 | | 118.8 | 467.9 | TR02 | 0.0 | P04 | | 0.0 |
| 27 | B2SL | 112.7 | | 500.0 | LK04 | 0.0 | OLN1 | MSL4 | 63.7 |
| | R04 | | 142.4 | 486.5 | | 0.0 | P27 | V04 | 82.3 |
| 28 | B2ML | 112.7 | | 500.0 | LK03 | 0.0 | MLN2 | LK04 | 0.0 |
| | R05 | | 142.4 | 486.5 | | 0.0 | P25 | | 0.0 |
| 29 | B2VH | 112.7 | | 500.0 | BLKR | 13.5 | BY2B | LK03 | 0.0 |
| | R06 | | 142.4 | 486.5 | TR03 | 0.0 | P05 | | 0.0 |
| 30 | E2SL | 118.0 | | 51.6 | LK11 | 0.0 | ELN2 | EVS1 | 0.0 |
| | R22 | | 272.5 | 0.0 | | 0.0 | P22 | V07 | 92.9 |
| 31 | E2SS | 118.0 | | 51.6 | LK34 | 0.0 | SPS2 | LK11 | 0.0 |
| | R48 | | 272.5 | 0.0 | | 0.0 | P35 | | 0.0 |
| 32 | E2PR | 118.0 | | 51.6 | LK12 | 0.0 | SEP2 | LK34 | 0.0 |
| | R23 | | 272.5 | 0.0 | | 0.0 | P08 | | 0.0 |
| 33 | E2ST | 118.0 | | 51.6 | MSL1 | 148.5 | RLN1 | LK12 | 0.0 |
| | R24 | | 272.5 | 0.0 | V01 | 45.3 | P18 | | 0.0 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 34 | E1SL | 118.0 | | 51.6 | LK09 | 0.0 | ELN1 | EVS1 | 0.0 |
| 0 | | | | | | | | | |
| | R41 | | 272.5 | 0.0 | | 0.0 | P21 | V07 | 92.9 |
| 0 | | | | | | | | | |
| 35 | E1LN | 118.0 | | 51.6 | LK31 | 0.0 | SKLN | LK09 | 0.0 |
| 0 | | | | | | | | | |
| | R47 | | 272.5 | 0.0 | | 0.0 | P20 | | 0.0 |
| 0 | | | | | | | | | |
| 36 | E1SI | 118.0 | | 51.6 | LK32 | 0.0 | SIP1 | LK31 | 0.0 |
| 0 | | | | | | | | | |
| | R46 | | 272.5 | 0.0 | | 0.0 | P30 | | 0.0 |
| 0 | | | | | | | | | |
| 37 | E1SS | 118.0 | | 51.6 | LK33 | 0.0 | SPS1 | LK32 | 0.0 |
| 0 | | | | | | | | | |
| | R45 | | 272.5 | 0.0 | | 0.0 | P34 | | 0.0 |
| 0 | | | | | | | | | |
| 38 | E1PR | 118.0 | | 51.6 | LK10 | 0.0 | SEP1 | LK33 | 0.0 |
| 0 | | | | | | | | | |
| | R42 | | 272.5 | 0.0 | | 0.0 | P07 | | 0.0 |
| 0 | | | | | | | | | |
| 39 | E1ST | 118.0 | | 51.6 | MSL1 | 148.5 | RLN1 | LK10 | 0.0 |
| 0 | | | | | | | | | |
| | R43 | | 272.5 | 0.0 | V01 | 45.3 | P18 | | 0.0 |
| 0 | | | | | | | | | |
| 40 | EPR2 | 272.5 | | 91.0 | EVS1 | 92.9 | EVR2 | OUTP | 0.0 |
| 0 | | | | | | | | | |
| | R17 | | 0.0 | 0.0 | V07 | 0.0 | P11 | | 0.0 |
| 0 | | | | | | | | | |
| 41 | EPR1 | 272.5 | | 91.0 | EVS2 | 90.9 | EVR1 | OUTP | 0.0 |
| 0 | | | | | | | | | |
| | R16 | | 0.0 | 0.0 | V08 | 0.0 | P10 | | 0.0 |
| 0 | | | | | | | | | |
| 42 | CSLS | 302.5 | | 25.0 | LK07 | 0.0 | CRLN | CCS1 | 20.0 |
| 0 | | | | | | | | | |
| | R37 | | 0.0 | 0.0 | | 0.0 | P29 | V05 | 0.0 |
| 0 | | | | | | | | | |
| 43 | CSSS | 302.5 | | 25.0 | LK35 | 0.0 | SPS1 | LK07 | 0.0 |
| 0 | | | | | | | | | |
| | R49 | | 0.0 | 0.0 | | 0.0 | P34 | | 0.0 |
| 0 | | | | | | | | | |
| 44 | CSP1 | 302.5 | | 25.0 | LK08 | 0.0 | SEP1 | LK35 | 0.0 |
| 0 | | | | | | | | | |
| | R38 | | 0.0 | 0.0 | | 0.0 | P07 | | 0.0 |
| 0 | | | | | | | | | |
| 45 | CCST | 302.5 | | 25.0 | MSL3 | 109.0 | RLN1 | LK08 | 0.0 |
| 0 | | | | | | | | | |
| | R39 | | 0.0 | 0.0 | V03 | 0.0 | P18 | | 0.0 |
| 0 | | | | | | | | | |

CIP ROUTE SCHEDULE

| INDEX | CIRCUIT NAME | ROUTE NAME | START TIME | STOP TIME | ELEMENT 1 | CODE | ELEMENT 2 | CODE | TIME TO RUN |
|---|---|---|---|---|---|---|---|---|---|
| 1 | DARE | CSKA | 10.00 | 25.00 | STKR | TR19 | | | 0.00 |

APPENDIX C (9.4) SCHEDULE OUTPUT

BATCH SEQUENCE

| PROCESS STATUS | ROUTES ROUTE NAME | INITIAL BATCH | REM/NG BATCH | START TIMED | LINKED TIME | PRIORITY | INDEX 1 | INDEX 2 |
|---|---|---|---|---|---|---|---|---|
| I | strt | 0.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | CSLS | 126.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | CSSS | 126.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | CSP1 | 126.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | CCST | 126.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2SL | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2SS | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2PR | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2ST | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1SL | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1LN | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1SI | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1SS | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1PR | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1ST | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B1SL | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B1ML | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B1VH | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B2SL | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B2ML | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B2VH | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B3SL | 481.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B3ML | 481.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B3VH | 481.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B1SL | 481.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B1ML | 481.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B1VH | 481.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B2SL | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B2ML | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | B2VH | 500.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2SL | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2SS | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2PR | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E2ST | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1SL | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1LN | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1SI | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1SS | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1PR | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | E1ST | 51.6 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| R | EPR2 | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| R | EPR1 | 91.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| R | CSLS | 25.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| R | CSSS | 25.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| R | CSP1 | 25.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| R | CCST | 25.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| CIP ROUTES | | | | | | | | |
| I | stcp | 0.0 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| I | CSKA | 0.0 | 15.0 | 0.0 | 0.0 | 1 | 0 | 0 |

PROCESS ROUTE SCHEDULE

| ROUTE /CODE | START TIME | STOP TIME | REM/NG BATCH | ELEMENT CONTENTS 1 | | ELEMENT CONTENTS 2 | ELEMENT 3 | | SAVE /REST |
|---|---|---|---|---|---|---|---|---|---|
| 1 CSLS | 0.0 | 126.0 | LK07 | 0.0 | CRLN | CCS2 | | 0.0 | 0 |
| R37 | | 302.5 | 25.0 | 0.0 | P29 | V06 | | 90.9 | |

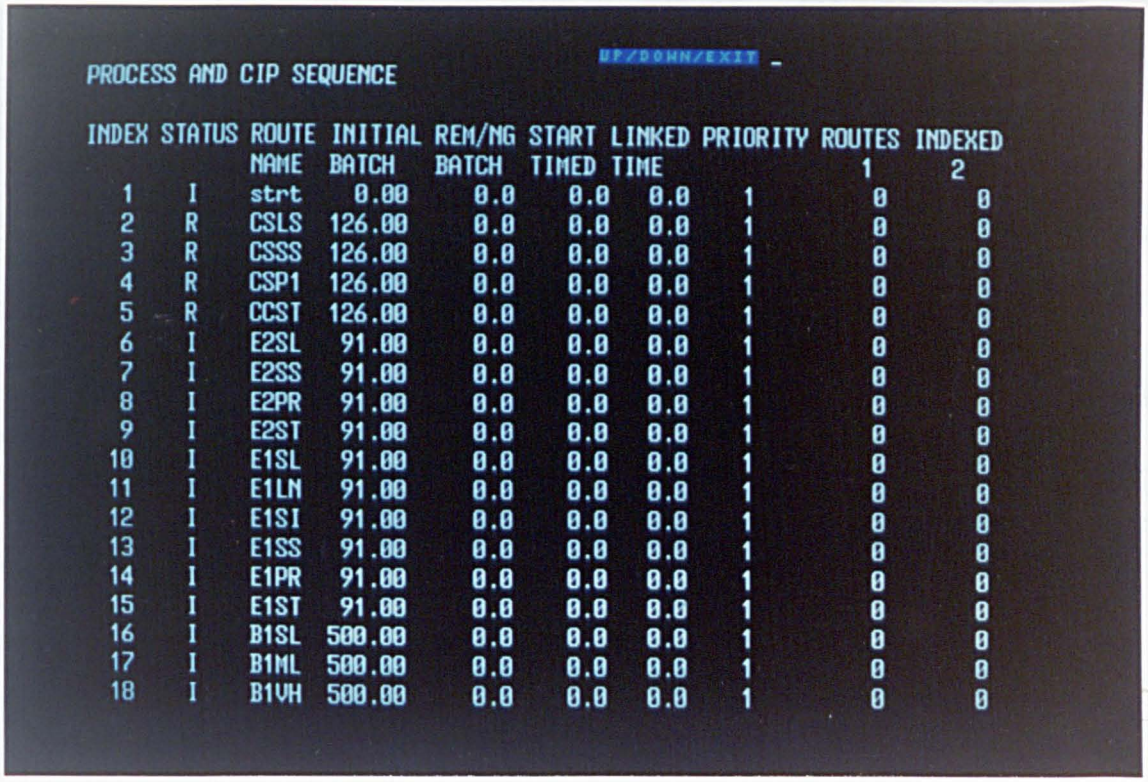
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | CSSS | 0.0 | | 126.0 | LK35 | 0.0 | SPS3 | LK07 | 0.0 |
| 0 | R49 | | 302.5 | 25.0 | | 0.0 | P36 | | 0.0 |
| 0 | | | | | | | | | |
| 3 | CSP1 | 0.0 | | 126.0 | LK08 | 0.0 | SEP3 | LK35 | 0.0 |
| 0 | R38 | | 302.5 | 25.0 | | 0.0 | P09 | | 0.0 |
| 0 | | | | | | | | | |
| 4 | CCST | 0.0 | | 126.0 | MSL3 | 210.0 | RLN2 | LK08 | 0.0 |
| 0 | R39 | | 302.5 | 25.0 | V03 | 109.0 | P19 | | 0.0 |
| 0 | | | | | | | | | |
| 5 | E2SL | 0.0 | | 91.0 | LK11 | 0.0 | ELN2 | EVS2 | 20.0 |
| 0 | R22 | | 118.0 | 51.6 | | 0.0 | P22 | V08 | 90.9 |
| 0 | | | | | | | | | |
| 6 | E2SS | 0.0 | | 91.0 | LK34 | 0.0 | SPS2 | LK11 | 0.0 |
| 0 | R48 | | 118.0 | 51.6 | | 0.0 | P35 | | 0.0 |
| 0 | | | | | | | | | |
| 7 | E2PR | 0.0 | | 91.0 | LK12 | 0.0 | SEP2 | LK34 | 0.0 |
| 0 | R23 | | 118.0 | 51.6 | | 0.0 | P08 | | 0.0 |
| 0 | | | | | | | | | |
| 8 | E2ST | 0.0 | | 91.0 | MSL1 | 227.3 | RLN1 | LK12 | 0.0 |
| 0 | R24 | | 118.0 | 51.6 | V01 | 148.5 | P18 | | 0.0 |
| 0 | | | | | | | | | |
| 9 | E1SL | 0.0 | | 91.0 | LK09 | 0.0 | ELN1 | EVS2 | 20.0 |
| 0 | R41 | | 118.0 | 51.6 | | 0.0 | P21 | V08 | 90.9 |
| 0 | | | | | | | | | |
| 10 | E1LN | 0.0 | | 91.0 | LK31 | 0.0 | SKLN | LK09 | 0.0 |
| 0 | R47 | | 118.0 | 51.6 | | 0.0 | P20 | | 0.0 |
| 0 | | | | | | | | | |
| 11 | E1SI | 0.0 | | 91.0 | LK32 | 0.0 | SIP1 | LK31 | 0.0 |
| 0 | R46 | | 118.0 | 51.6 | | 0.0 | P30 | | 0.0 |
| 0 | | | | | | | | | |
| 12 | E1SS | 0.0 | | 91.0 | LK33 | 0.0 | SPS1 | LK32 | 0.0 |
| 0 | R45 | | 118.0 | 51.6 | | 0.0 | P34 | | 0.0 |
| 0 | | | | | | | | | |
| 13 | E1PR | 0.0 | | 91.0 | LK10 | 0.0 | SEP1 | LK33 | 0.0 |
| 0 | R42 | | 118.0 | 51.6 | | 0.0 | P07 | | 0.0 |
| 0 | | | | | | | | | |
| 14 | E1ST | 0.0 | | 91.0 | MSL1 | 227.3 | RLN1 | LK10 | 0.0 |
| 0 | R43 | | 118.0 | 51.6 | V01 | 148.5 | P18 | | 0.0 |
| 0 | | | | | | | | | |
| 15 | B1SL | 40.0 | | 500.0 | LK02 | 0.0 | OLN1 | MSL4 | 0.0 |
| 0 | R01 | | 81.8 | 481.0 | | 0.0 | P27 | V04 | 28.9 |
| 0 | | | | | | | | | |
| 16 | B1ML | 40.0 | | 500.0 | LK01 | 0.0 | MLN1 | LK02 | 0.0 |
| 0 | R02 | | 81.8 | 481.0 | | 0.0 | P24 | | 0.0 |
| 0 | | | | | | | | | |
| 17 | B1VH | 40.0 | | 500.0 | ACCM | 19.0 | BY1A | LK01 | 0.0 |
| 0 | R03 | | 81.8 | 481.0 | TR09 | 0.0 | P01 | | 0.0 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 18 | B2SL | 60.0 | | 500.0 | LK04 | 0.0 | OLN1 | MSL4 | 9.1 |
| 0 | R04 | | 112.7 | 476.0 | | 0.0 | P27 | V04 | 63.7 |
| 0 | | | | | | | | | |
| 19 | B2ML | 60.0 | | 500.0 | LK03 | 0.0 | MLN2 | LK04 | 0.0 |
| 0 | R05 | | 112.7 | 476.0 | | 0.0 | P25 | | 0.0 |
| 0 | | | | | | | | | |
| 20 | B2VH | 60.0 | | 500.0 | ACCM | 24.0 | BY2A | LK03 | 0.0 |
| 0 | R06 | | 112.7 | 476.0 | TR10 | 0.0 | P02 | | 0.0 |
| 0 | | | | | | | | | |
| 21 | B3SL | 90.0 | | 481.0 | LK06 | 0.0 | OLN1 | MSL4 | 32.7 |
| 0 | R07 | | 117.9 | 468.3 | | 0.0 | P27 | V04 | 70.8 |
| 0 | | | | | | | | | |
| 22 | B3ML | 90.0 | | 481.0 | LK05 | 0.0 | MLN3 | LK06 | 0.0 |
| 0 | R08 | | 117.9 | 468.3 | | 0.0 | P26 | | 0.0 |
| 0 | | | | | | | | | |
| 23 | B3VH | 90.0 | | 481.0 | BLKR | 12.7 | BY3A | LK05 | 0.0 |
| 0 | R09 | | 117.9 | 468.3 | TR01 | 0.0 | P03 | | 0.0 |
| 0 | | | | | | | | | |
| 24 | B1SL | 90.0 | | 481.0 | LK02 | 0.0 | OLN1 | MSL4 | 32.7 |
| 0 | R01 | | 118.8 | 467.9 | | 0.0 | P27 | V04 | 71.5 |
| 0 | | | | | | | | | |
| 25 | B1ML | 90.0 | | 481.0 | LK01 | 0.0 | MLN1 | LK02 | 0.0 |
| 0 | R02 | | 118.8 | 467.9 | | 0.0 | P24 | | 0.0 |
| 0 | | | | | | | | | |
| 26 | B1VH | 90.0 | | 481.0 | BLKR | 13.1 | BY1B | LK01 | 0.0 |
| 0 | R03 | | 118.8 | 467.9 | TR02 | 0.0 | P04 | | 0.0 |
| 0 | | | | | | | | | |
| 27 | B2SL | 112.7 | | 500.0 | LK04 | 0.0 | OLN1 | MSL4 | 63.7 |
| 0 | R04 | | 142.4 | 486.5 | | 0.0 | P27 | V04 | 82.3 |
| 0 | | | | | | | | | |
| 28 | B2ML | 112.7 | | 500.0 | LK03 | 0.0 | MLN2 | LK04 | 0.0 |
| 0 | R05 | | 142.4 | 486.5 | | 0.0 | P25 | | 0.0 |
| 0 | | | | | | | | | |
| 29 | B2VH | 112.7 | | 500.0 | BLKR | 13.5 | BY2B | LK03 | 0.0 |
| 0 | R06 | | 142.4 | 486.5 | TR03 | 0.0 | P05 | | 0.0 |
| 0 | | | | | | | | | |
| 30 | E2SL | 118.0 | | 51.6 | LK11 | 0.0 | ELN2 | EVS1 | 0.0 |
| 0 | R22 | | 272.5 | 0.0 | | 0.0 | P22 | V07 | 92.9 |
| 0 | | | | | | | | | |
| 31 | E2SS | 118.0 | | 51.6 | LK34 | 0.0 | SPS2 | LK11 | 0.0 |
| 0 | R48 | | 272.5 | 0.0 | | 0.0 | P35 | | 0.0 |
| 0 | | | | | | | | | |
| 32 | E2PR | 118.0 | | 51.6 | LK12 | 0.0 | SEP2 | LK34 | 0.0 |
| 0 | R23 | | 272.5 | 0.0 | | 0.0 | P08 | | 0.0 |
| 0 | | | | | | | | | |
| 33 | E2ST | 118.0 | | 51.6 | MSL1 | 148.5 | RLN1 | LK12 | 0.0 |
| 0 | R24 | | 272.5 | 0.0 | V01 | 45.3 | P18 | | 0.0 |
| 0 | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 34 | E1SL | 118.0 | | 51.6 | LK09 | 0.0 | ELN1 | EVS1 | 0.0 | 0 |
| | R41 | | 272.5 | 0.0 | | 0.0 | P21 | V07 | 92.9 | 0 |
| 35 | E1LN | 118.0 | | 51.6 | LK31 | 0.0 | SKLN | LK09 | 0.0 | 0 |
| | R47 | | 272.5 | 0.0 | | 0.0 | P20 | | 0.0 | 0 |
| 36 | E1SI | 118.0 | | 51.6 | LK32 | 0.0 | SIP1 | LK31 | 0.0 | 0 |
| | R46 | | 272.5 | 0.0 | | 0.0 | P30 | | 0.0 | 0 |
| 37 | E1SS | 118.0 | | 51.6 | LK33 | 0.0 | SPS1 | LK32 | 0.0 | 0 |
| | R45 | | 272.5 | 0.0 | | 0.0 | P34 | | 0.0 | 0 |
| 38 | E1PR | 118.0 | | 51.6 | LK10 | 0.0 | SEP1 | LK33 | 0.0 | 0 |
| | R42 | | 272.5 | 0.0 | | 0.0 | P07 | | 0.0 | 0 |
| 39 | E1ST | 118.0 | | 51.6 | MSL1 | 148.5 | RLN1 | LK10 | 0.0 | 0 |
| | R43 | | 272.5 | 0.0 | V01 | 45.3 | P18 | | 0.0 | 0 |
| 40 | EPR2 | 272.5 | | 91.0 | EVS1 | 92.9 | EVR2 | OUTP | 0.0 | 0 |
| | R17 | | 0.0 | 0.0 | V07 | 0.0 | P11 | | 0.0 | 0 |
| 41 | EPR1 | 272.5 | | 91.0 | EVS2 | 90.9 | EVR1 | OUTP | 0.0 | 0 |
| | R16 | | 0.0 | 0.0 | V08 | 0.0 | P10 | | 0.0 | 0 |
| 42 | CSLS | 302.5 | | 25.0 | LK07 | 0.0 | CRLN | CCS1 | 20.0 | 0 |
| | R37 | | 0.0 | 0.0 | | 0.0 | P29 | V05 | 0.0 | 0 |
| 43 | CSSS | 302.5 | | 25.0 | LK35 | 0.0 | SPS1 | LK07 | 0.0 | 0 |
| | R49 | | 0.0 | 0.0 | | 0.0 | P34 | | 0.0 | 0 |
| 44 | CSP1 | 302.5 | | 25.0 | LK08 | 0.0 | SEP1 | LK35 | 0.0 | 0 |
| | R38 | | 0.0 | 0.0 | | 0.0 | P07 | | 0.0 | 0 |
| 45 | CCST | 302.5 | | 25.0 | MSL3 | 109.0 | RLN1 | LK08 | 0.0 | 0 |
| | R39 | | 0.0 | 0.0 | V03 | 0.0 | P18 | | 0.0 | 0 |

CIP ROUTE SCHEDULE

| INDEX | CIRCUIT NAME | ROUTE NAME | START TIME | STOP TIME | ELEMENT 1 | CODE | ELEMENT 2 | CODE | TIME TO RUN |
|---|---|---|---|---|---|---|---|---|---|
| 1 | DARE | CSKA | 10.00 | 25.00 | STKR | TR19 | | | 0.00 |



PROCESS AND CIP SEQUENCE

UP/DOWN/EXIT -

| INDEX | STATUS | ROUTE NAME | INITIAL BATCH | REM/NG BATCH | START TIMED | LINKED TIME | PRIORITY | ROUTES 1 | INDEXED 2 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | I | strt | 0.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 2 | R | CSLS | 126.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 3 | R | CSSS | 126.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 4 | R | CSP1 | 126.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 5 | R | CCST | 126.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 6 | I | E2SL | 91.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 7 | I | E2SS | 91.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 8 | I | E2PR | 91.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 9 | I | E2ST | 91.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 10 | I | E1SL | 91.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 11 | I | E1LN | 91.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 12 | I | E1SI | 91.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 13 | I | E1SS | 91.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 14 | I | E1PR | 91.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 15 | I | E1ST | 91.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 16 | I | B1SL | 500.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 17 | I | B1ML | 500.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |
| 18 | I | B1VH | 500.00 | 0.0 | 0.0 | 0.0 | 1 | 0 | 0 |

Photo 9.1

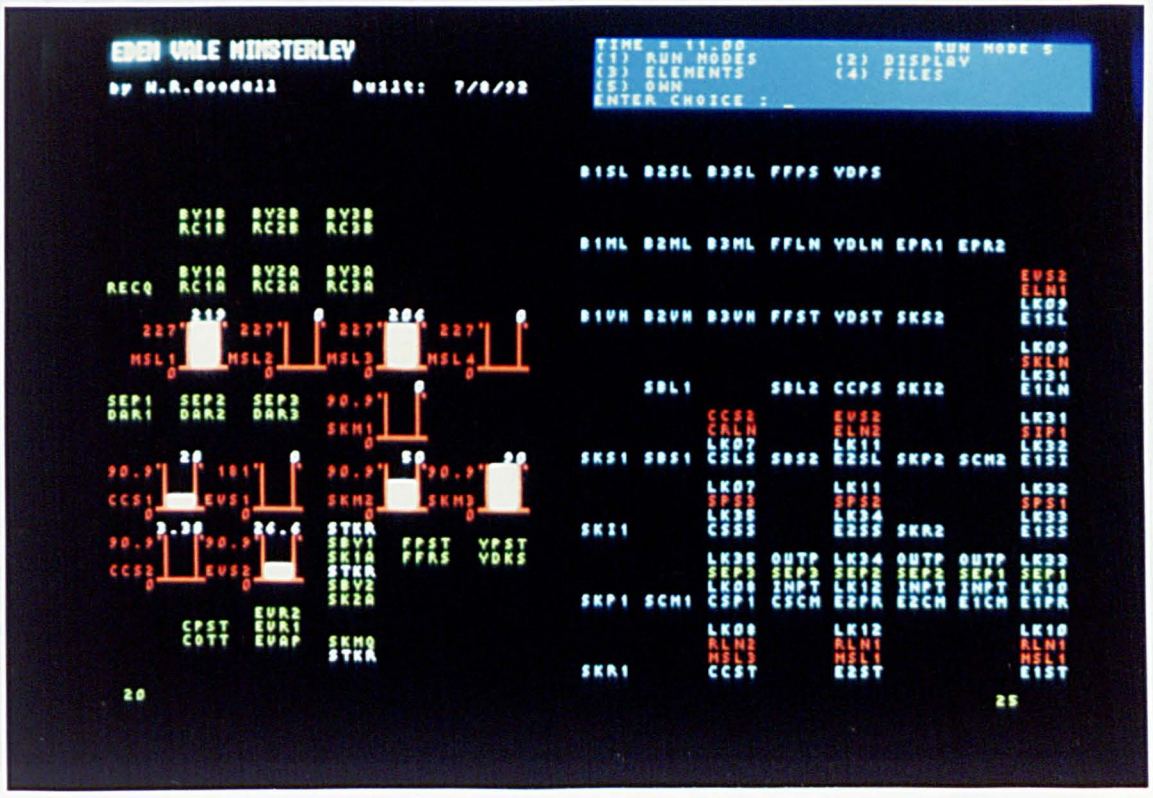



Photo 9.2.

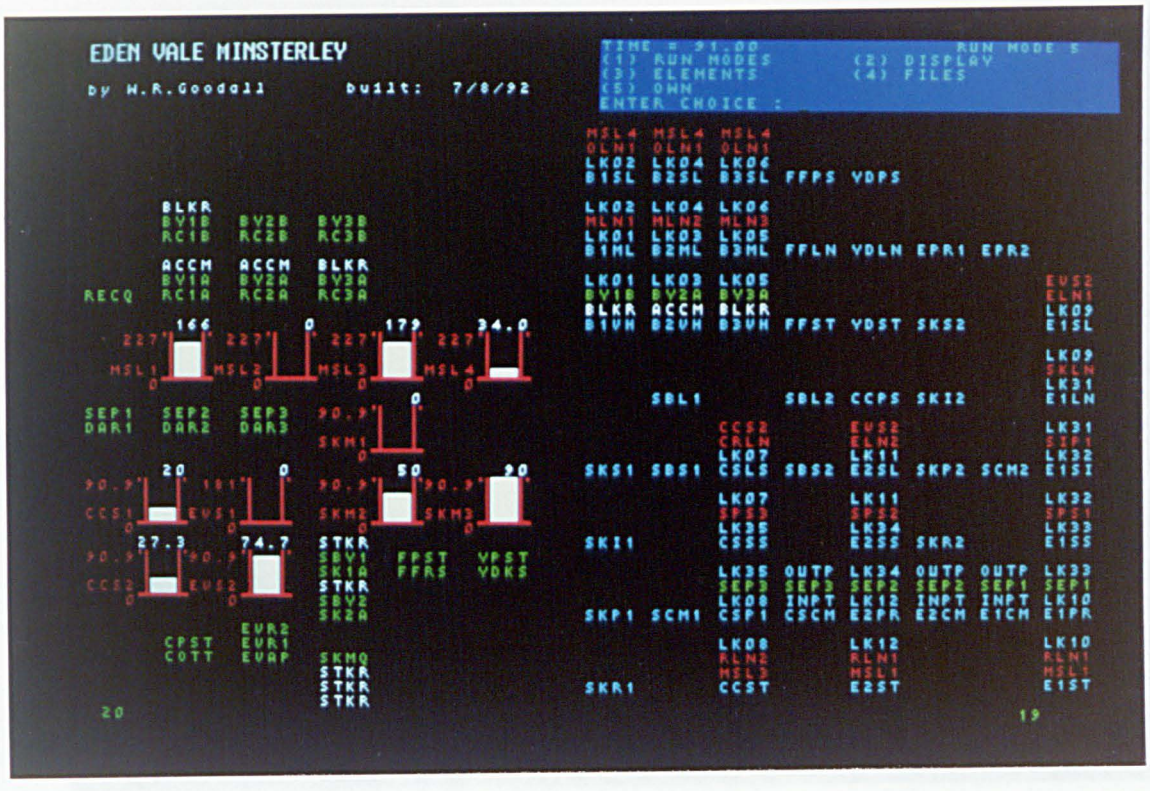



Photo 9.3

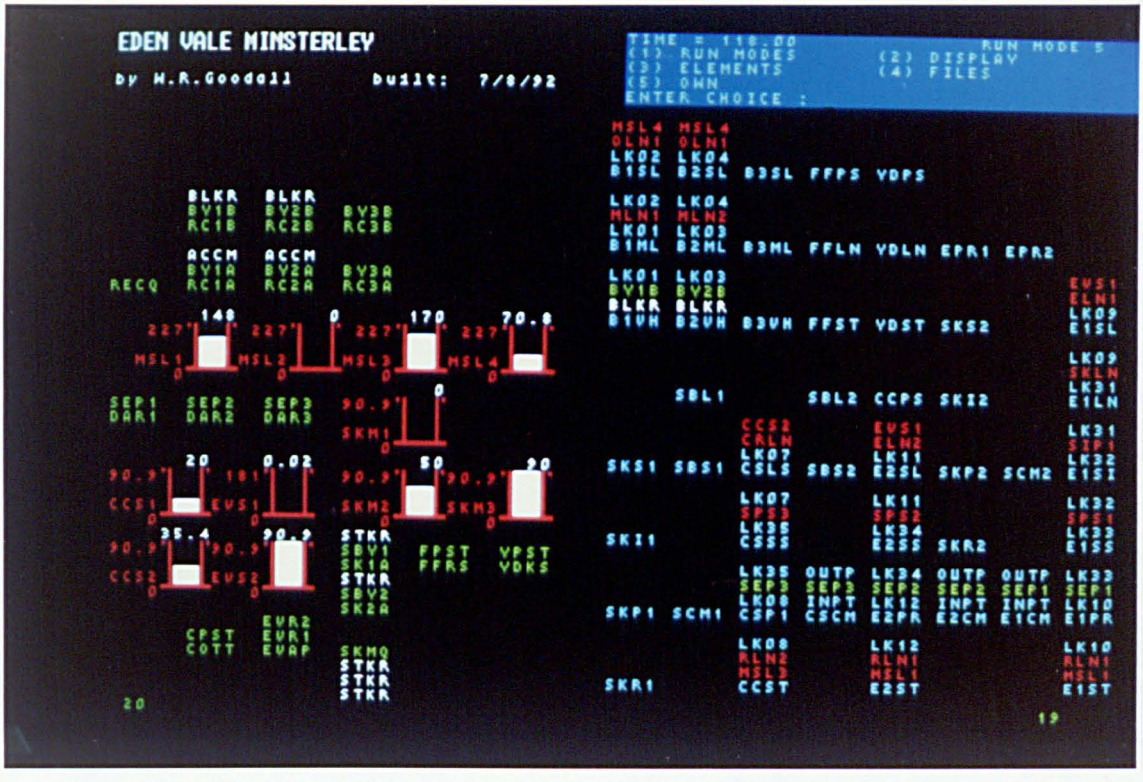



Photo 9.4.

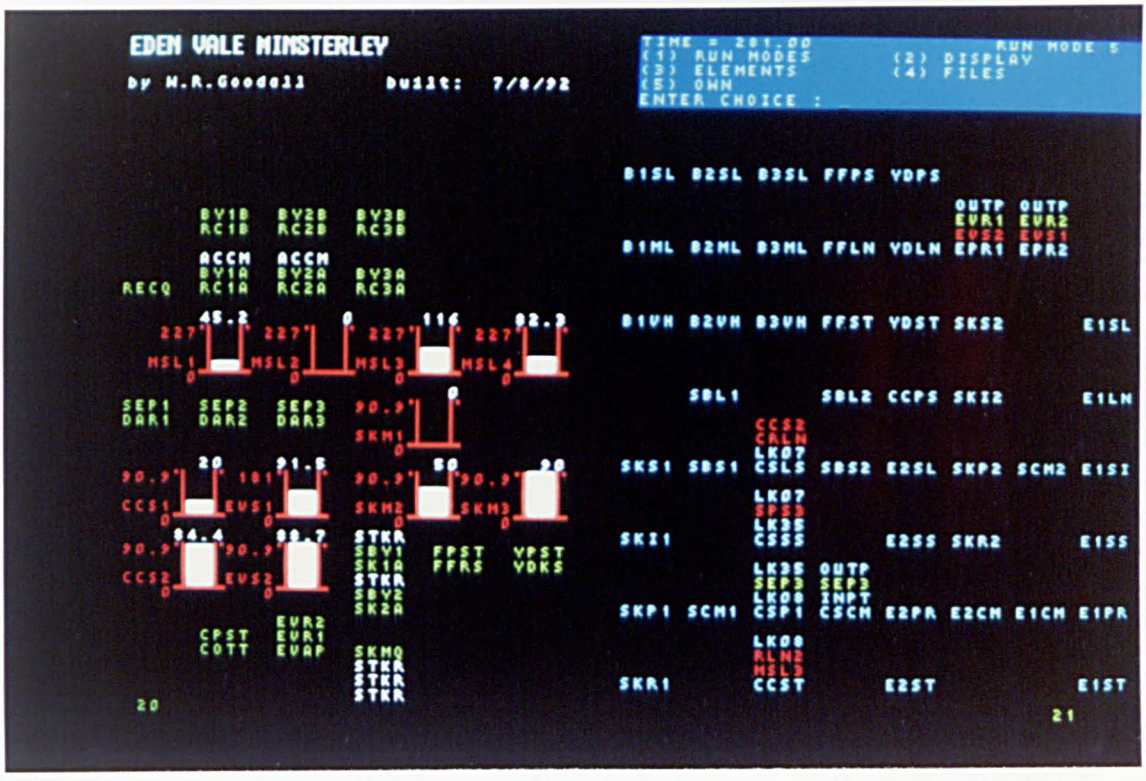
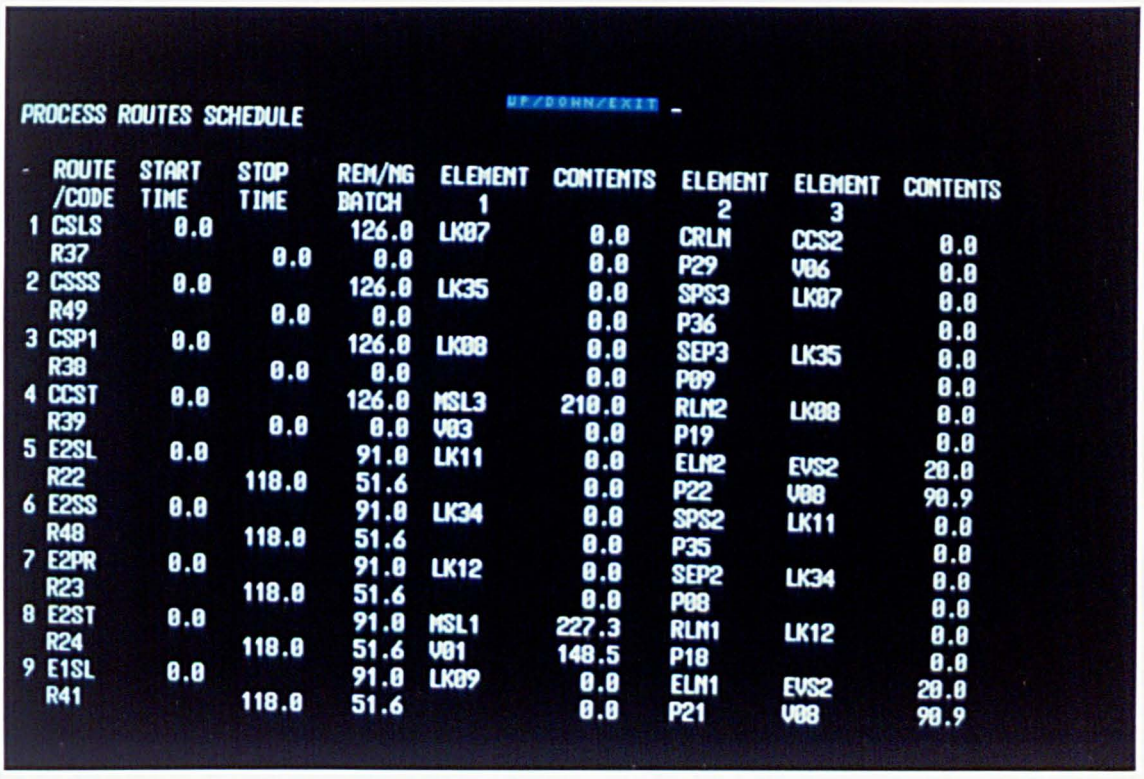



Photo 9.5




| - | ROUTE /CODE | START TIME | STOP TIME | REM/MG BATCH | ELEMENT 1 | CONTENTS | ELEMENT 2 | ELEMENT 3 | CONTENTS |
|---|---|---|---|---|---|---|---|---|---|
| 1 | CSLS | 0.0 | | 126.0 | LK07 | 0.0 | CRLN | CCS2 | 0.0 |
| | R37 | | 0.0 | 0.0 | | 0.0 | P29 | V06 | 0.0 |
| 2 | CSSS | 0.0 | | 126.0 | LK35 | 0.0 | SPS3 | LK07 | 0.0 |
| | R49 | | 0.0 | 0.0 | | 0.0 | P36 | | 0.0 |
| 3 | CSP1 | 0.0 | | 126.0 | LK08 | 0.0 | SEP3 | LK35 | 0.0 |
| | R38 | | 0.0 | 0.0 | | 0.0 | P09 | | 0.0 |
| 4 | CCST | 0.0 | | 126.0 | MSL3 | 210.0 | RLM2 | LK08 | 0.0 |
| | R39 | | 0.0 | 0.0 | V03 | 0.0 | P19 | | 0.0 |
| 5 | E2SL | 0.0 | | 91.0 | LK11 | 0.0 | ELM2 | EVS2 | 20.0 |
| | R22 | | 118.0 | 51.6 | | 0.0 | P22 | V08 | 90.9 |
| 6 | E2SS | 0.0 | | 91.0 | LK34 | 0.0 | SPS2 | LK11 | 0.0 |
| | R48 | | 118.0 | 51.6 | | 0.0 | P35 | | 0.0 |
| 7 | E2PR | 0.0 | | 91.0 | LK12 | 0.0 | SEP2 | LK34 | 0.0 |
| | R23 | | 118.0 | 51.6 | | 0.0 | P08 | | 0.0 |
| 8 | E2ST | 0.0 | | 91.0 | MSL1 | 227.3 | RLM1 | LK12 | 0.0 |
| | R24 | | 118.0 | 51.6 | V01 | 148.5 | P18 | | 0.0 |
| 9 | E1SL | 0.0 | | 91.0 | LK09 | 0.0 | ELM1 | EVS2 | 20.0 |
| | R41 | | 118.0 | 51.6 | | 0.0 | P21 | V08 | 90.9 |

Photo 9.6.

# The Development of a Hybrid Knowledge Based Simulation System for Scheduling and Control in the Batch Process Industry

R.Roy and W.R.Goodall

Warwick Manufacturing Group, Department of Engineering,
The University of Warwick, Coventry, CV4 7AL, United Kingdom.

**Abstract**
An approach to short term scheduling and control in the batch process industry using hybrid Discrete Event Simulation is presented. The system can be used to develop feasible short term schedules according to user defined scheduling rules. The benefits of the hybrid approach and the implementation of the system are discussed.

## 1. INTRODUCTION

The batch process industry includes food, brewing, fine chemicals, and pharmaceuticals. These areas are characterised by the production of a diverse range of final products starting from a few basic raw materials, which are processed in batches according to production recipes. A batch process plant typically consists of a network of process and storage vessels, linked by pipework through continuous process plant such as separators and heat exchangers. Output from the plant may be bulk liquid or powder products, or high volume discrete packaged consumer goods such as yoghurt. The recipes determine for the final product and any intermediate products, the processing tasks to be carried out, their order, and the plant requirements and processing conditions.

A plant may have a configuration to allow flexibility in batch routing, and the production of several products at the same time in parallel campaigns [1], however, its operation is restricted by constraints such as finite storage capacity, precedence between process stages, the time critical nature of some operations, and requirements for cleaning of the pipework, process plant and vessels. Although a plant may be optimally designed and scheduled to satisfy long/medium term forecasts of product demand, short term fluctuating demand patterns pose problems of day to day scheduling and control which still need to be addressed and there is a considerable amount of research interest in this area [2,3,4], which is the focus of this paper.

## 2. DISCRETE EVENT SIMULATION AND SHORT TERM SCHEDULING

Discrete Event Simulation is a long established Operational Research (OR) technique used for the analysis of the operating characteristics of manufacturing systems. More recently the use of simulation for short term production scheduling has been investigated [5,6]. A model is built which represents the dynamics and constraints of a manufacturing system, and which a scheduler can use experimentally to develop realistic, feasible short term production schedules.

Different approaches have been suggested for the experimental development of schedules and are discussed in [6]. The simulation is run using a set of rules which govern the sequencing of batches/jobs and choice of resources within the system. The event trace from the simulation, output in a suitable format, is a detailed schedule of the movement of batches/jobs and use of resources over time. The scheduler can examine the output and decide if it is acceptable. If unacceptable, alterations can be made to the parameters of the simulation to run it again, and through an iterative approach the scheduler can decide on the best schedule produced in relation to the objectives for the system. The chosen schedule can be released for use within the plant for production activity control and monitoring of plant performance.

## 3. KNOWLEDGE BASED SIMULATION

In recent years it has been recognised that simulation has a lot in common with the area of Artificial Intelligence (AI) [7] in that both approaches seek to represent explicit knowledge about the world through objects, their attributes and interrelationships. Researchers have concluded that simulation can benefit considerably from the use of AI techniques for example in aiding model developers through the use of an Intelligent Front End (IFE) to allow natural language input to a model generator [8] or the analysis and interpretation of simulation results [9]. The strong similarity between simulation models and the branch of AI known as Expert Systems has also been recognised [10], in that both are modular, both contain rules concerned with the behaviour of a system and both have an independent inference mechanism or executive that applies the rules and effects changes in the state of the system. Thus, the common purpose of both simulation and knowledge based or expert systems in providing a framework for representation of the behaviour of a manufacturing system is obvious, and the power of the latter, through the use of declarative languages such as PROLOG and Lisp, in representing the rules governing such behaviour is well established. However, unlike simulation, knowledge based systems do not provide a dynamic model dealing with the changes in the status of the manufacturing system over time. It has been concluded that the inclusion of a time advance mechanism within the knowledge based system framework will enable simulation models to be developed that can benefit from increased expressiveness in the rulebase, and flexible data structures. This approach has been used in the area of simulation for scheduling and control for example in [11], which uses PROLOG to model a Flexible Manufacturing System, and it's on-line control system.

## 4. HYBRID APPROACH TO MODEL DEVELOPMENT

Transferring all aspects of simulation modelling into the declarative format is not necessarily the definite direction to take [12], and there are positive features of the traditional procedural languages used for simulation which will be lost, such as the well developed event list management mechanisms, the efficient handling of large amounts of numeric computation which is a feature of most simulation models, and the already developed toolkits for graphical simulation such as SEE-WHY [13]. Therefore using a hybrid approach to exploit the strengths of both formats appears to be a promising way to develop a knowledge based simulation for a given application and this is the approach adopted for the system described in this paper.

## 5. SYSTEM ARCHITECTURE

The function of batch process scheduling is the assignment of equipment and process routes in a time coordinated manner for the processing of intermediate and final product batches through the plant to meet, if possible, the production goals for the plant. For a multiproduct plant this is a complex problem taking into account flexibility for batch routing, and constraints such as finite capacity and cleaning requirements. The scheduling function is generally carried out according to a set of heuristic rules. These rules may have been developed locally based on practical experience, or they may be based on priority dispatching rules, or other heuristics. The system developed is designed to allow the use of whatever rule based scheduling techniques are relevant to the particular plant being modelled, and to allow experimentation to develop a set of applicable scheduling rules, so that on input of a batch requirement list to the system the output will be a detailed feasible schedule of plant activity.

The system has a hybrid modular architecture, consisting of a database, control rulebase, and control inference mechanism implemented in PROLOG, with an interface to a dynamic simulation model of the domain implemented in FORTRAN using the Visual Interactive Simulation (VIS) toolkit SEE-WHY. It is a generic system, meant for user development by data input without the need for programming. A generic system is specific enough to cover the logic of the domain at which it is directed, but also flexible enough to allow many different models from that domain to be built using the same program so it does not become redundant when the real system is changed.

The database contains the domain entity specifications as frames. These are data structures [14], which represent the different classes of entity which make up the batch process domain, their hierarchical relationships, and their relevant characteristics, for example continuous process elements, storage vessels, incubation vessels, process lines, and transportation elements which are combined either into process sub-routes, for moving and processing batches through the plant, or cleaning routes as necessary. Also non-entity data such as the process recipes, and a 'planner' to handle the batch list and the current sequence of sub-routes, are represented by frames. To build up a model the system carries out a dialogue with the user to create specific instances of these frames, which are held in working memory and represent the system state in the simulation run. Once an initial model has been developed with frame instances, in subsequent sessions the initialisation dialogue with the user can be restricted to amendments, for example a daily update of the batch requirement list.

While this detailing of the generic system can be achieved using linked FORTRAN data arrays for each data type, (and was for the initial development of the system), PROLOG allows a more coherent and flexible data structure to be used. In the PROLOG environment the data about an entity is either stored directly in a frame instance, or can be inherited from a more general frame instance which details common characteristics of the entity class, or can be inferred from the application of rules. Because any particular data field or slot can contain a list of complex data items of any size, the ability to model the true complexity of the system is greatly enhanced. For example the use of an entity in a possible route is constrained by it's configuration with respect to other entities in the plant. This configuration can easily be modelled as an AND/OR structure in PROLOG, and held in a slot of the entity frame instance. Figure 1 represents part of a plant configuration in which three Separators can output to two out of three Storage Vessels. The configuration of Output Line One is shown with respect to the Separators and the Vessels, as an AND/OR graph in Figure 2.

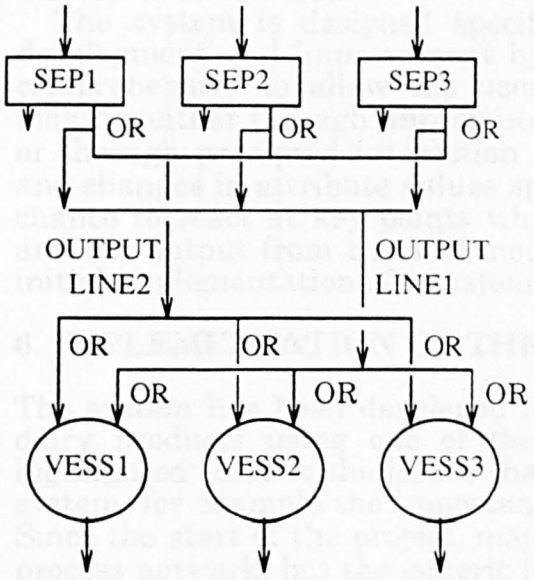



Figure 1. Part of a Plant Configuration.

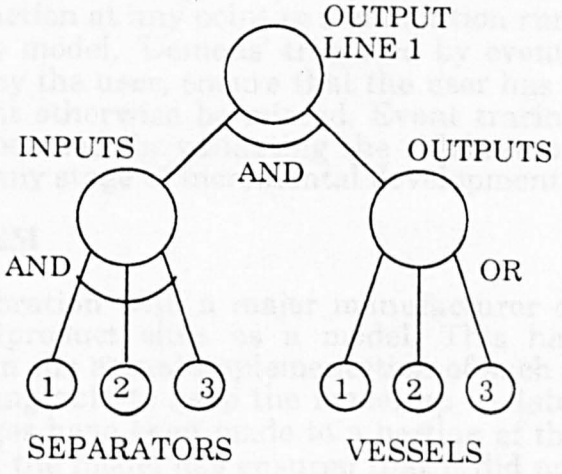



Figure 2. AND/OR Configuration Output Line 1.

The simulation module contains the logic relevant to a batch process environment to move the system forward dynamically in time, for example it contains the logic to schedule events for the end of a processing activity based on the elements which make up process-routes. Related sub-routes which comprise a process within the network are linked during the simulation to ensure coordination between the activities as necessary. The module also makes use of the SEE-WHY facilities for graphics so that the user can follow the progress of the simulation. The simulation model uses the same data used to build up the frame instances to ensure consistency within the system. As the simulation moves forward in time, when there is a change in state represented by an event it updates the slot values of the PROLOG frame instances with relevant dynamic data via the PROLOG/FORTRAN interface.

The rulebase contains the domain specific rules as developed by the user for scheduling the production of batches, and handling other domain specific situations, for example determining queue assignments for transportation elements in the system, taking into account the production recipes/procedures, system goals, and constraints.

The control module handles the application of the rules based on the system state to produce a sequence of process and cleaning routes, and the elements which will make them up, to be passed to the simulation. The simulation model is then activated, and the model moves forward in time to generate a new system state for the scheduling cycle to continue. At the end of the scheduling period, or at any point the user chooses, reports can be generated on batch status within the plant; and a predictive time phased schedule, as a complete 'Bill of Process' listing routes, elements, start and end times for activities, product types, batch

sizes etc.. This can be used to operate the plant, against which progress can be monitored for control.

The system is designed specifically to be interactive for incremental rule development and improvement by the user/system builder. These facilities are comprehensive to allow the user as much flexibility as possible in decision making, either through unprompted interaction at any point in a simulation run, or through prompted interaction from the model. 'Demons' triggered by events and changes in attribute values specified by the user, ensure that the user has a chance to react at key points which might otherwise be missed. Event tracing and the output from the VIS model can be used for validating the rulebase on initial implementation of a system and at any stage of incremental development.

## 6. IMPLEMENTATION OF THE SYSTEM

The system has been developed in collaboration with a major manufacturer of dairy products using one of their multiproduct sites as a model. This has highlighted some of the issues that arise in the actual implementation of such a system, for example the importance of being able to keep the model up to date. Since the start of the project, major changes have been made to a portion of the process network, but the generic format of the model has ensured that it did not become redundant as a result of this. The gathering of data to build the model is time consuming, but brings benefits through improved understanding of the plant, and the data is now recorded and available for any use that can be made of it. The development of a correct and 'good' rulebase is obviously a critical factor in the value of the system to the user, and a 'top down' approach was adopted for this project, using a series of interviews to get management objectives followed by interactive use of the model to develop the rulebase. It is important in this process that the system does not simply become a mimic of an already existing manual system, which may not be developing particularly good schedules anyway, so one of the objectives should be to assess current scheduling methods and incorporate appropriate scheduling practise from the literature.

Regular system demonstrations and communication between the system developer and the system users helps to build confidence and ensure that the system will do what is required of it. A gradual program of implementation should be adopted. In the first instance the system is used as a strategic tool to look at the general operation of the plant and identify critical areas. Confidence in the scheduling capabilities is increased by the reproduction of manually developed schedules. Then schedules developed by the system on a stand alone basis can be critically evaluated without being implemented, before the system is actually put 'on-line' to develop schedules for use on a day to day basis.

## 7. CONCLUSIONS

The aim of Discrete Event Simulation for scheduling is to provide the user with information on what is achievable, and what is not, in terms of the operation of a plant to meet a set of objectives under a particular configuration and operating rules/constraints. The system described in this paper will improve the flexibility available to represent the structure, constraints, and operating rules of a real system through the use of the declarative format of AI while retaining the useful features of procedurally based simulation systems.

Since the control module is completely separated from the simulation module, it should be possible to interface the former to the real system to act as a real-time controller; the control module would then act on dynamic system status data from the plant rather than the simulation model. However, this would need

further research on various issues. In the case of a deviation from the schedule should the objective be to direct the operations back towards the original schedule and what rules are required to achieve this? When should the simulation model be activated (e.g. in the case of a major plant breakdown)? These are some of the issues that need to be addressed to achieve the goal of developing a true on-line scheduler.

## REFERENCES


1. H.Ku, D.Rajagopalan, I.Karimi, Scheduling in Batch Processes, p35-45, Chemical Engineering Progress, v83, pt8, August 1987.
2. D.Daugherty, R.Felder, An Expert System for Scheduling Production in a Multipurpose Speciality Chemicals Plant, p44-49, Plant/Operations Progress, v9, n1, January 1990.
3. U.Egli, D.Rippin, Short Term Scheduling for Multiproduct Batch Chemical Plants, p303-325, Computers and Chemical Engineering, v10, n4, 1986.
4. F.Rodammer, K.White, A Recent Survey of Production Scheduling, p841-851, IEEE Transactions on Systems, Man, and Cybernetics, v18, n6, November/December 1988.
5. F.Grant, R.Lagoni, The Users Role in a Simulation Based Scheduling System, p936-941, Proceedings of the 1989 Winter Simulation Conference, Washington DC, December 1989, IEEE.
6. R.Roy, Scheduling and Capacity Management Using Simulation, to be published in Logistics Technology International 1992, Sterling Publications.
7. J.Vaucher, Views of Modelling: Comparing the Simulation and AI approaches, p3-7, Artificial Intelligence, Graphics and Simulation, Proceedings of the 1985 SCS Multiconference, San Diego CA, January 1985, SCS.
8. R.Paul and G.Doukidis, Further Developments in the Use of Artificial Intelligence Techniques which Formulate Simulation Problems, p787-810, Journal Operational Research Society, v37, n8, 1986.
9. S.Prakash, R.Shannon, Intelligent Back End of a Goal Directed Simulation Environment for Discrete Part Manufacturing, p883-891, Proceedings of the 1989 Winter Simulation Conference, Washington DC, December 1989, IEEE.
10. G.Doukidis, An Anthology on the Homology of Simulation with Artificial Intelligence, p701-712, Journal Operational Research Society, v38, n8, 1987.
11. P.Sackett and I.Fan, The Control of a Flexible Manufacturing System by Short Term Goal Identification, p123-143, International Journal of Advanced manufacturing Technology, v4, 1989.
12. W.Walker, K.Maughan, E.Fletcher, P.Smith, Knowledge Based Systems with Embedded Simulation Components, p318-323, Proceedings UKIT90 Conference, Southampton, March 1990, IEE.
13. SEE-WHY Users Manual, AT and T Istel Ltd.
14. E.Rich, K.Knight, Artificial Intelligence, McGraw-Hill, 1991.

FUTURE PROCESS

SILO LINE/ EVAP LINE 1 ø76
EVAP LINE 2 ø76
COTTAGE CHEESE ø76

FUTURE SEPARATOR

SEPARATOR 1 DRG.NO.DD900046
DI 214 (MILK AVAILABLE) DO 147 (ROUTE OK.)
7 M15 CIP >> 2 L15 SKIM ø63
T266 TT

SEPARATOR 2 DRG.NO.DD90046
DI 215 DO 146
8 L15 CIP >> 3 L15 SKIM ø63
T267 TT

SEPARATOR 3 DRG.NO.DD90047
DI 216 DO 145
9 G15 CIP >> 4 G15 SKIM ø76
T268 TT

NOTE:-
SEP.1&2 SEE DRG.NO.DD90046.
SEP.3 SEE DRG.NO.DD90047.

FB104 FB103 FIR FM232 FM233 FM234

FB75 FB76 FB77 FB90 FB91 FB92 FB93 FB94 FB95 ø76 W/M FM238 FB98 FB97 FB96 FB99 FB100 FB101

SEPARATORS MANIFOLD

WATER

FB102 ø76 ø51

EVAPORATOR (SEE DRG.NO.CE 90021)

EVAPORATOR LINE 2 — 13 AT T1
EVAPORATOR LINE 1 — 12 AT T2

V166 V165

P231(OF EVAPORATOR) 48000 L/HR. 18.5 kW V167

COTTAGE CHEESE (SEE DRG.NO.BC/SA/M90103)

6 AT M1
10 AT M1
AV30 AV32 AV79 AV68
S1 S2
CIP FEED COTTAGE CHEESE
CIP RETURN

EVAPORATOR SILO
VIA FLOWPLATE FP1
EVAPORATOR SILO

DO 155

DI 212 (MILK REQD.) DO 144 (MILK AVAILABLE) RESEARCH & DEV. 32 >> 31 CIP.RET.
DI 213 (MILK REQD.) DO 143 (MILK AVAILABLE) SKI COOL 34 J1 >> 33 J1 DRG.NO.DY 89064 CIP.RET.
FROMAGE FRAIS (HT1) 30 C1 >> 20 C1 DRG.NO.DF 90081 CIP.RET.

FB108 FB105 FB106 FB107
AIR FROM SOL.VALVE 106
92
AIR FROM SOL.VALVE 91(350) OF YOG.DRINKS
91/1 91/2 OF YOG.DRINKS.(N/O)
WATER (YOG.DRINKS)

WATER FB8 FB7 FB6 FB111 FB110 C256 FB5 FB4 DA46 DA47
WATER 2273 LITRES HL184 LL185 RECOV. RINSE 3410 LITRES HL188 ML189 LL187 HL190 DETERGENT 3410 LITRES
FB10 FB9 FB1 FB2 FB3 FB11
V136 STERILANT P112 .04 kW
T257 TIC C258 FS191 STEAM CV272 TIC T273
E/STOP ES 200 AIR PRESS AP201
P109 11 kW
CAUSTIC DETERGENT DETERGENT ACID DETERGENT ACID DETERGENT
P110 .18 kW
C.I.P. SET NO.3 FLOW RATE 34096 LITRES/HR.

SINGLE SHOT C.I.P. 40915 LITRES/HR.
WATER FB12 HL192 ML193 LL194 909 LITRES C.I.P. SET NO.1
STERILANT P116 PUMP .04 kW FB115 FB114 FB13 FB14 FB15 P113 5.5 kW
DA44 DA45 T259 C260 FS195 TIC C261
E/STOP ES 202 AIR PRESS AP203
CV274 STEAM TIC T275 FB130

SINGLE SHOT C.I.P. 40915 LITRES/HR.
WATER FB16 HL196 ML197 LL198 909 LITRES C.I.P. SET NO.2
STERILANT P120 PUMP .04 kW FB119 FB118 FB17 FB18 FB19 P117 5.5 kW
DA44 DA45 T261 C262 FS198 TIC C263
E/STOP ES 204 AIR PRESS AP 205
CV276 STEAM TIC T277 FB131

FB23 FB22 P121 7.5 kW FB20 FB21 P122 7.5 kW
ø51 HOSE ø51 HOSE
BAY 1 BAY 2 BAY 3

PD 210 F/M DISP. DO137 OFF CIP DI 206 DO138 DO139 OFF CIP BATCH DI 207 DI 208
DO 140 OFF CIP DI 209
PD 210 F/M DISP. DO141 DO 142 OFF CIP BATCH DI 210 DI 211

HL 217 LL 218 LT263 1
HL 219 LL 220 LT264 2
HL 221 LL 222 LT265 3
P/E ø100

FB33 FB32 FB31 FB44 FB43 FB42 FB55 FB54 FB53
FB74 FB35 FB46 ø76 ø25
FB123 35000 L/HR. 7.5 kW FB68
FB25 FB36 FB47
FB26 FB27 FB28 FB29 FB30 FB37 FB38 FB39 FB40 FB41 FB48 FB49 FB50 FB51 FB52
FB63 FB64 FB57 FB60 FB65 FB58 FB61 FB66 FB59 FB62 FB67 FB56
FM239 FIR 25000 L/HR. P124 3 kW FB69 FB70
FM235 FIR DA41 48000 L/HR. P125 7.5 kW FB71
FM236 FIR DA17 48000 L/HR. P126 7.5 kW FB72
FM237 FIR 48000 L/HR. P127 18.5 kW FB68A FB73
FB34 FB45 ø76

EV CALIBRATION REF NO.

| ISSUE | REV | DATE | SUBJECT OF CHANGE | INTS | FACTORY | ENG | TECH |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 29.1.92. | DRG.IMPORTED IN TO CAD SYSTEM | G.W. | | | |


Eden Vale
THE CREAMERY MINSTERLEY SHREWSBURY SHROPSHIRE SY5 OBN
TEL: 0743 791301 FAX:0743 790509

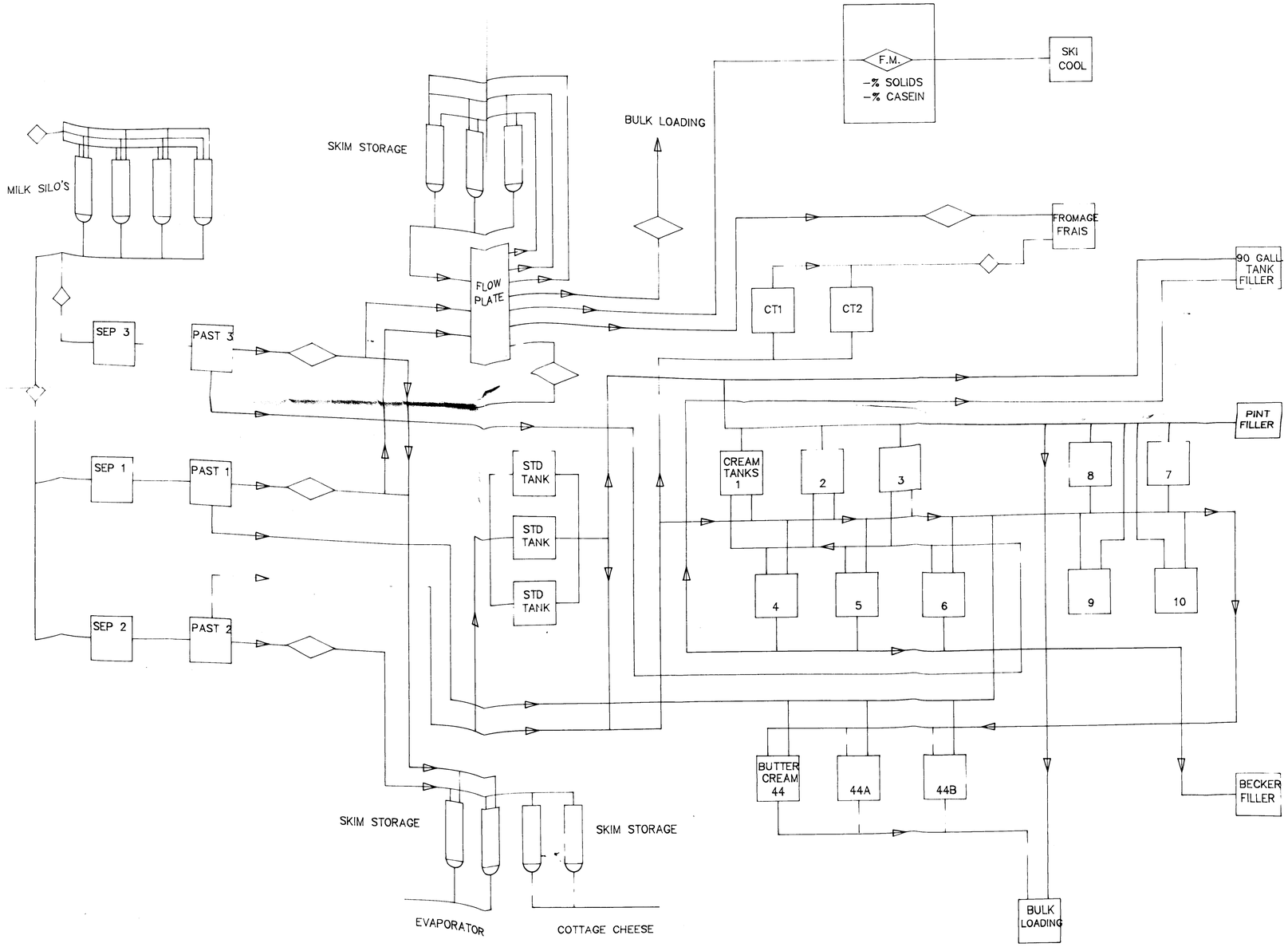

| DRAWN | DATE | CHECKED BY | DATE |
|---|---|---|---|
| GW | 5.3.91. | | |

MINSTERLEY QUALITY SYSTEM

| DEPARTMENTAL APPROVAL | | DRAWING No. |
|---|---|---|
| SIGN. | DATE | D/S/M/91030 |

TITLE: SKIM MILK STORAGE & DISPATCH PROCESS SCHEMATIC

C.A.D REF. DS91030H




| ISSUE | DATE | REVISIONS | INTS |
|---|---|---|---|


EDEN VALE

THE CREAMERY MINSTERLEY SHREWSBURY SHROPSHIRE SY5 0BN

TEL: 0743 791301 FAX: 0743 790509


PROJECT

C.A.D REF. GARY

TITLE

LAYOUT OF DAIRY
3 SILO SKIM DESPATCH

| DRAWN | CHECKED BY | APPROVED BY | SCALE | DRAWING No. |
|---|---|---|---|---|
| GW | | | N.T.S. | |
| DATE 3.9.91. | DATE | DATE | | |