The Knowledge Engineering Review, Vol. 17:3, 261–294. © 2002, Cambridge University Press
DOI: 10.1017/S0269888902000498 Printed in the United Kingdom

Operating procedure synthesis: science or art?

RAFAEL BATRES,[1] JAMES SOUTTER,[2] STEVEN P. ASPREY[3] and PAUL CHUNG[4]

[1] *Chemical Resources Laboratory, Tokyo Institute of Technology, Japan*
[2] *Division of Informatics, University of Edinburgh, UK*
[3] *Centre for Process Systems Engineering, Imperial College, London, UK*
[4] *Computer Science Department, Loughborough University, UK*

Abstract

In this paper, we provide a review of concepts and developments in operating procedure synthesis (OPS), starting from its early development through to its current state. Operating procedure synthesis is a problem in which a set of equipment manipulations and their orderings must be generated to take the process from an initial state to a goal state. While there has been ongoing research for about 30 years in this area, only a few systems have been reported to be industrially deployed. The approach taken in this paper is, first, to describe the problem in general terms; second, to discuss previous work in this area; and, finally, to present ideas and directions for future work.

1 Introduction

According to the High Pressure Gas Safety Institute of Japan, statistics show that 46% of the accidents that occurred in petrochemical plants between 1952 and 1990 took place during transient operations such as startup and shutdown (KHK, 1991). Too frequently, design engineers do not include storage vessels, purge lines or vessels or lines for storing intermediate products. If, during the early stages of the process design, transient operations such as startup and shutdown are not contemplated, the plant may not be suitably designed for the unsteady state conditions that arise soon after the commissioning of the plant. In Japan, under the High-Pressure Gas Law, refineries need to shut down operations every two years to conduct inspections and carry out maintenance, a situation that makes proper operating procedures even more critical.

In the United States, technically, accurate operating procedures are mandatory documents that must be developed to comply with the OSHA standard for Process Safety Management of Highly Hazardous Chemicals (OSHA, 1992) operating procedures are frequently referred to as Standard Operating Procedures (SOPs) that contain the tasks to be performed for startup and shutdown, data to be recorded, operating conditions to be maintained, samples to be collected, what to do when an upset condition occurs, which alarms and instruments are pertinent if an upset condition occurs and so on. Furthermore, it has been recognised that improving the control of startups, shutdowns and grade changes presents a potential of world benefits of about 100 million US$ per year (Benson & Perkins, 1997).

Operating procedure synthesis can be described as a planning problem where the sequences of actions need to be found to satisfy one or more goals. While a number of approaches in the area of automatic planning for scheduling and production have been used in industry, the generation of operating procedures is still left to expert engineers and plant personnel. In scheduling, the set of actions to be performed is known in advance; on the other hand, operating procedure synthesis is a problem in which the set of equipment manipulations and their orderings must be generated to take the process from an initial state to a goal state. The orderings of actions must satisfy a number of

constraints such as the chemistry of the process, product quality requirements, process constraints (e.g. those related to catalyst decomposition), safety constraints (flammability, explosiveness and toxicity), and mechanical constraints (design temperature and pressure limits). The synthesis of operating procedures is a creative and synthetic process and for this reason Japanese researchers know it as *sousa sekkei* (operations design).

This paper is intended to cover several aspects of the historical development of operating procedure synthesis approaches. The earliest work on operating procedure synthesis (OPS) was published by Rivas *et al*. (1974). Their work is based on early artificial intelligence research, particularly the development of the GPS (General Problem-Solver) system described by Ernst and Newell (1969). However, it is worth mentioning that computer-based analysis, rather than synthesis, of operating procedures was considered six years earlier by Peck (1968). After the publication of the seminal paper by Rivas *et al*. (1974), very little work was done on OPS for approximately twelve years; only five papers were published during this period all of which were short conference papers that outlined ideas for future work.

There was renewed interest in OPS around 1987; at the time, computers were thought to be cheap and powerful enough to make OPS practical. In 1988 a special issue of *Computers in Chemical Engineering* was published with OPS as a main theme. This burst of interest died down around 1991, when it was no longer clear how the then current OPS paradigms could be advanced. In the following eight years, OPS work had moved away from the procedure synthesis towards procedure analysis and procedure optimisation. However, very recently new approaches have been proposed that tackle OPS problems in a more realistic way. In particular, recent OPS systems address problems constrained by quantitative constraints and dynamic behaviour.

2 Operating procedure synthesis: problem statement

OPS can be viewed as searching for a set of sequenced primitive operations that transform a plant system from an initial state to a pre-specified goal state through a series of intermediate states. These primitive operations must be carried out in such a way that no violations are made of any relevant process or of mechanical, safety and environmental constraints. A plant system is a set of equipment items among which there is flow of material, energy or information. In general, planning problems are defined using initial and goal states. In its simplest form the problem lies in determining the order and kinds of action needed to move between the initial and goal states.

The general form of the operations-planning problem can be written as:

$$\min_{\delta_i} \| \mathbf{x}_0 - \mathbf{x}_{goal} \| \tag{1}$$

subject to

$$\mathbf{g}(\mathbf{x}) \leq 0$$

where $\mathbf{x}_0$ and $\mathbf{x}_{goal}$ represent the initial and goal states respectively, δ_i is an integer-valued variable representing an operation over an equipment item i and $\mathbf{g}(\mathbf{x})$ is a vector of inequalities representing process, safety and other constraints.

Solution of OPS problems as stated in equation (1) involves the following set of models:

1. *Plant model* – a model of the plant structure in terms of the topology of the interconnected equipment. This model constrains significantly the synthesis of operating procedures.
2. *Process models* – these are models that mimic the physicochemical behaviour of the process. Contrary to planning problems in other domains, in OPS the "state of the world" that is of most interest to engineers is not the state of the *hardware* structure but the state of the process described by the properties of the chemicals that are processed within the plant.
3. *Action templates* – these models describe when and how the effects of the actions will change the state of any given kind of equipment item.

4. *Execution model* – for an operating procedure to be useful, a representation format is needed such that procedures can be understood by humans and control systems to be executed in a reproducible fashion. The simplest model is that in which a series of sentences such as "open valve V-1" are generated. An example of a more flexible model is that in which plan steps are represented as conditions with extra information such as the purpose of the action. An execution model can be seen hierarchically as a procedure composed of plan steps. A plan step applies to a particular piece of equipment (e.g. open valve V-101) at a particular time in a particular operating procedure. On the other hand, an action template will define the skeleton of the plan step (e.g. opening a control valve is possible if the valve-state is closed and the valve-state becomes open).

The approaches for the solution to OPS problems can be organised into five categories: state graph; approaches based on deterministic effects; action synergy approaches; action ordering systems; and simulation-based planning approaches. The following sections provide details of approaches in each category.

3 State graph paradigm

The state graph approach to procedure synthesis was introduced independently by Ivanov *et al*. (1980a, 1980b) and by Kinoshita *et al*. (1982). State graph planning is the simplest form of procedure synthesis. In this approach, a plant is modelled by a graph describing all of its possible states. Nodes in the graph represent plant states and directed arcs in the graph represent the use of an action to move from one state to another. An example of a state graph is shown in Figure 1.

If a plant is represented by a state graph then an operating procedure for the plant can be represented by a path through the graph. The path starts at the node that represents the initial state of the procedure. The path traverses a sequence of arcs corresponding to the sequence of steps in the procedure. The path finishes at a node where the objectives of the procedure are satisfied. In this way, operating procedure synthesis is the task of finding a path that satisfies a given set of requirements. One acceptable path-searching algorithm is shown in Figure 2. This algorithm assumes that all modelling problems and safety issues are resolved when a state graph is created, e.g. every path through a graph represents a valid operating procedure for some objective.

Optimal operating procedures can be created using a similar algorithm if the arcs in the state graph are weighted with an associated cost.

The limitation of the state graph approach is the difficulty involved in creating and maintaining the state graph model of a plant. The size of a state graph is related exponentially to the number of variables in the planning domain. For instance, for a small plant that contains 17 valves, even if only the valve states of open and closed are considered, there will be 2^{17} states. A practical model of the plant will contain at least 100,000 states. It is not realistic to expect a team of people to create this huge graph by hand or to expect the resulting graph to be free from errors.

4 OPS based on deterministic effects

Most approaches to the generation of operating procedures find their roots in artificial intelligence planning techniques. The AI community has been developing a variety of approaches for the past 30

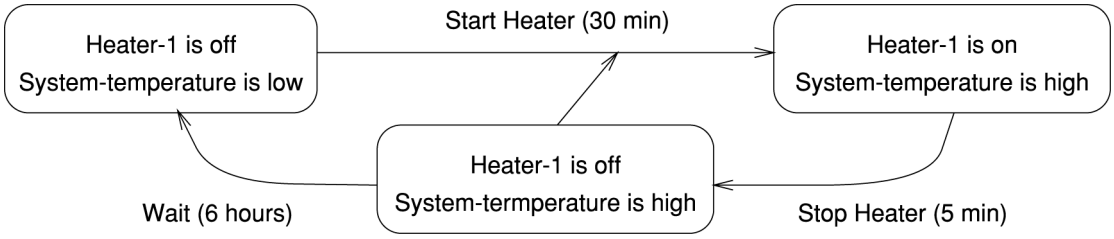

Figure 1 The state graph for a single heater

years; most of which fall into the category of the so-called "state-based planning" approaches or, more properly, "planning approaches" based on deterministic effects. In these types of planner, the current state of the plant and process and the action performed solely determine the next state of the plant. In other words, the OPS system does not have to check to make sure the action worked.

Deterministic effects planning approaches have origins in the STRIPS (Stanford Research Institute Problem Solver) planning system (Fikes & Nilsson, 1971). In STRIPS-like planners, action templates (referred to as *operators*), consist of two parts: preconditions and effects. The preconditions specify the state of the world (or the plant) under which the action can be executed; effects represent the state of the world (e.g. the state of valves and process variables).

4.1 Forward-chaining paradigm

The forward-chaining paradigm is in some ways very similar to the state graph paradigm. The two paradigms consider the same search space, the space represented by a state graph. They also use the same basic search algorithm involving choosing an action, finding the effects of that action and then planning on from the resulting state.

The difference in the forward-chaining approach is that the state graph is not represented explicitly during planning. Instead, an OPS system is given enough knowledge to implicitly generate and explore the state graph during synthesis.

Action templates in the forward-chaining paradigm are similar to the operators of STRIPS-like planners. The action template model used in forward chaining consists of three components: a model defining when each action template is applicable (preconditions), a model of the effects of each action template and a model to decide when the state of the plant is safe. The basic forward-chaining algorithm is shown in Figure 3. In this algorithm, the three models are used in steps 2, 5 and 7 respectively.

The forward-chaining algorithm of Figure 3 has three important properties:

1. New actions are always added to the end of a procedure. In other words, the n^{th} action added to the procedure will also be the nth action in the finished procedure. To simulate the effect of any operating procedure, one can simply simulate the effect of the first action in the initial state and then the second action in the resulting state and so on. If new actions are always added to the end of a procedure then the procedure can be simulated incrementally by simulating each new action. This incremental simulation is used in the forward-chaining algorithm to evaluate the safety of each action and to check that the preconditions of each action are satisfied.

StateGraphPlan(*start*, *goal*, *nodes*, *arcs*, *seen*, *procedure*)

1. if *goal_start* $\subset$ then succeed.
2. *options* = The set of all arcs that connect from *start* to some other node.
3. if options = $\varnothing$ then backtrack.
4. edge = choose an arc from *options*.
5. new = select the node from *nodes* such that edge = $\langle$ *start*, *new* $\rangle$.
6. if new $\in$ *seen* then backtrack.
7. call StateGraphPlan(*new*, *goal*, *nodes*, *arcs*, *seen* $\cup$ *new*, *procedure* + *edge*)

Figure 2 A state graph planning algorithm

ForwardChainingPlan(*start*, *goal*, *action-templates*, *seen*, *procedure*)

1. if *goal* $\subset$ *start* return success.
2. options = set of all *op* such that *op* is an instantiation of some action-template in action-templates and all the preconditions of *op* are true in start.
3. if *options* = $\varnothing$ then backtrack.
4. *action* = choose an action from *options*.
5. *new* = simulate the effects of action in the state start.
6. if *new* $\in$ *seen* then backtrack.
7. if $\neg$ safe(*new*) then backtrack.
8. call ForwardChainingPlan(*new*, *goal*, *actio-templates*, *seen*) $\bigcup$ {*new*} *procedure* + *action*)

Figure 3 The basic forward chaining planning algorithm

2. When attempting to solve an OPS problem, the algorithm effectively considers the space of all valid procedures that have the required initial state. From this space, the algorithm chooses a procedure that has the desired effects. The forward-chaining algorithm is algorithmically correct and complete because all possible procedures are considered. Furthermore, difficulties that occur in more complex OPS systems are avoided, e.g. the need to resolve conflict between objectives and the need to resolve safety problems.
3. Step 6 in the algorithm is necessary to prevent looping. If it were not for this step then plans of the form "open valve-a, close valve-a, open valve-a etc." would be explored.

All current forward-chaining systems use action-template selection heuristics to improve on the performance of the basic forward-chaining algorithm. These heuristics attempt to avoid considering the actions that will not help achieve the given objectives. When *intelligent* action-template selection is used in a system, the system loses the second property of the basic forward-chaining algorithm. The system does not consider all possible procedures and so is not guaranteed to be algorithmically complete.

All current systems are incomplete for two reasons: (i) they fail to select the actions needed to solve some problems where the solutions to two goals conflict in a particular way;[1] (ii) the actions needed to resolve some safety problems are not selected or are selected too late. The repair of an unsafe procedure may require adding new actions into the body of the existing procedure. The evaluation of the safety in forward-chaining OPS is dependent on the assumption that actions will only be added to the end of the procedure. As such, the forward-chaining algorithm would have to be changed significantly to allow new actions to be added to the middle of a procedure. No current forward-chaining OPS systems attempt this.

In conclusion, the ideal forward-chaining algorithm is correct and complete, but is too slow to be practical. Current forward-chaining OPS systems reduce the search space by using intelligent action-template selection algorithms; however, the action-template selection algorithms used by current systems are incomplete.

[1] As a point of interest, McDermott(1996) describes a forward-chaining algorithm that uses intelligent action selection and is complete. The algorithm was developed to solve planning problems and has not been tried in OPS domains.

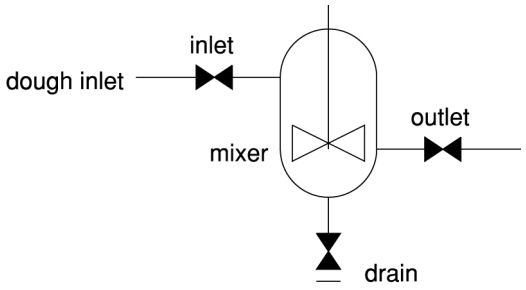

Figure 4 A plant to demonstrate Rivas *et al.*'s system

4.1.1 Connectors and valves network approach

The work by Rivas *et al.* (1974)[2] is the earliest forward-chaining OPS system. The system is designed specifically for valve sequencing tasks. The procedures produced by the system only involve opening and closing valves. The goals given to the system describe the need for a flow or a purge or the need to isolate a unit or to confine a given chemical species. As an example of the problem-solving ability of the system, consider the plant in Figure 4 and the operation of moving dough into the mixer. In the initial state the mixer is full of cleaning water; the water should not come into contact with the dough.

For the plant considered here, Rivas *et al.*'s system can produce the three-step procedure to open the drain valve in order to remove the water, close the drain valve to avoid losing the dough and then open the inlet valve to add the dough. This procedure is created in response to the single goal "trap the dough in the mixer".

The high-level synthesis algorithm used in the system is given in Figure 5. The implementation of this algorithm is essentially composed of three steps: select action-template, determine action effects and verify safety.

In the first step, the algorithm selects action templates that are possible, safe and contribute to the goal. Verification of safety is considered during action-template selection, rather than later in the cycle, because of a need to constrain the set of possible actions. For implementation reasons, the planner will

RivasRuddPlanner(*start*, *goal*, *action-templates*, *procedure*)

1. *differences* = goals – start
2. if *differences* = ∅ then *success*.
3. *options* = set of all *op* such that *op* is an instantiation of some action-template in action-templates and all the preconditions of *op* are true in start and (∃ *g* ∈ differences).(related(*op*, *g*) and safe(determine_action_effects(*op*, *current*))).
4. if *options* = ∅ then repair plan safety.
5. *action* = select any action from *options*. This choice is not remade through backtracking.
6. *new* = simulate(*action*, *start*).
7. call RivasRuddPlan(*new*, *goal*, *action-templates*, *procedure* + *action*)

Figure 5 Rivas *et al.*'s OPS algorithm

[2] This work is also reported in Rivas and Rudd (1975).

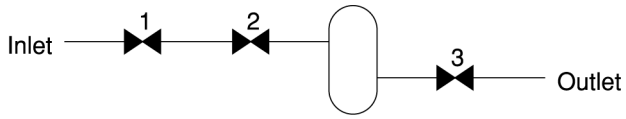

Figure 6 The problem of creating a flow into a tank

not backtrack and thus the set of possible actions must be as small as possible. An instance of an action template *contributes* to a goal if it satisfies a rule specific to the type of goal. For example, an action-template contributes to a goal that creates flow of a chemical through a specific point if the effect of the action-template creates a flow into that point or if the action-template can create a flow out of that point. Consider the goal to create a flow through the vessel in Figure 6. Opening valve 3 contributes to the goal since this action creates a flow out of the vessel; however, opening valve 1 or valve 2 does not contribute to the goal because opening either of these two valves on their own will not create a flow. In general, the planner cannot solve flow goals that require two or more valves to be operated. In other words, the OPS system is incomplete.

Action-template selection requires an agenda of the goals that are left to be solved. In Rivas *et al*. (1974), the agenda is defined as the set of goals not satisfied in the current state. In Figure 5, this agenda is called *differences*. Synthesis stops when the agenda is empty and so all goals have been solved. As a result, the planner only generates procedures that satisfy their goals and so the system is algorithmically correct. A goal may appear on the agenda after it has already been solved once. This will occur if some action negates the original solution to a goal. In some cases, procedures will gain redundant actions as a result.

In rare cases, the OPS system will fail to terminate. As a worst-case example, consider the plant in Figure 7 and the objectives: (1) isolate the section of pipe between valves 1 and 2, (2) create a flow from the inlet and (3) create a flow to the outlet. The correct solution to this problem is to open valves 3 and 4. The system may start by opening valves 1 and 2 to create the flow. The system will then close valves 1 and 2 to isolate the pipe section. The actions to open valves 1 and 2 are now redundant, as the valves have been closed again. The system does not perform loop-checking and so there is no reason why the cycle of opening and closing valves 1 and 2 should not continue indefinitely.

The determination of action effects takes the form of a hard-coded function. The function essentially analyses a plant state to find where each chemical species is flowing and where it is confined. It is significant that this action-effect model can deduce that closing a valve will block an existing flow. Later OPS systems have difficulty protecting a flow because they cannot easily reason about when a flow becomes blocked.

As in all forward-chaining systems, a procedure is considered safe unless some unsafe situation is caused by the procedure. The safety of a procedure is evaluated though the simulation of the effects of each action. In the Rivas *et al*. planner, unsafe situations are represented by logical statements. There are five different types of situation that can be defined as unsafe: (1) the mixing of two or more chemicals, (2) a flow between two units, (3) the blocking of a particular flow, (4) a particular chemical

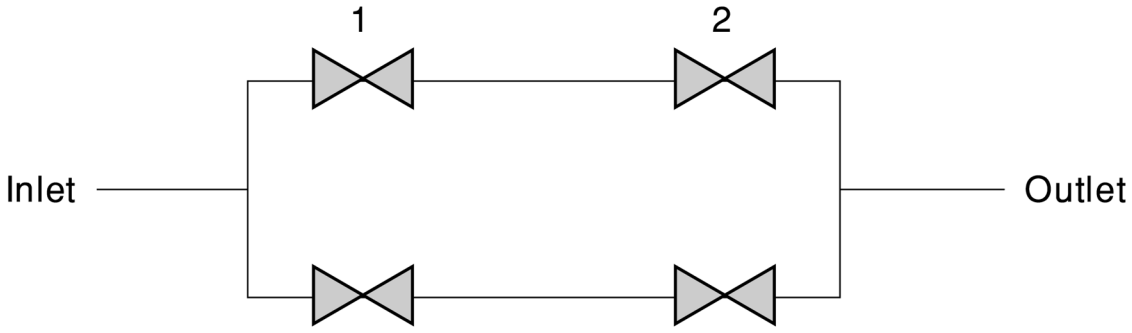

Figure 7 A case where looping may occur

reaching a particular outlet, and (5) a chemical entering a particular unit. A majority of later OPS systems cannot represent all five types of safety constraint; the exception is Strimaitis (1987). When needed, the planner will add a step to drain a unit in order to prevent the creation of an unsafe mixture of chemicals. Draining is modelled as completely removing a chemical from a unit. A draining step is added if all actions that are positively related to a goal are unsafe because they cause an unsafe mixture to form. The planner will not act to avoid unsafe situations that do not involve mixing. For example, the planner will not reroute a protected flow of a chemical in order to prevent the flow from becoming blocked.

4.1.2 Stationary-state approach

Fusillo and Powers (1987, 1988a, 1988b) describe an attempt to solve general OPS problems using an adaptation of the procedure synthesis algorithm given in Rivas *et al*. (1974). This approach introduced the concept of stationary-state, which is an intermediate state that is changing slowly with respect to time and is located between the initial and final states with the ability to wait until the next operation is taken.

The presence of simultaneous inverse operations, such as evaporation and condensation, indicates the possibility of stationary states. This original concept breaks down the whole planning problem into more simplified subproblems. Additionally, a stationary state can function as a recovery state in case of emergency, including the possibility of partially shutting down some sections of the plant while keeping the rest in operation.

An example from Fusillo and Powers (1987) demonstrates the kind of problems their planner can solve. The example is based around the plant in Figure 8. The four goals are to start the heater, compressor, methane inlet and chlorine inlet. There are three significant safety constraints:

1. The heater may not be started while the compressor is off.
2. Chlorine and methane should not mix while the system is cold.
3. Methane should not be added to chlorine.

When run on this problem, the planner proposes to start the compressor, then the heater, then the methane feed, and finally the chlorine feed. The algorithm used in Fusillo and Powers (1987) is shown in Figure 9. The most notable difference between this algorithm and the work in Rivas *et al*. (1974) is that action selection and action ordering are separated. This separation limits the system's ability to select the actions that will make up a procedure.

The action-template selection method used in Fusillo and Powers (1987) selects exactly one action for each goal that is not true in the start state. This prevents looping because there is a limit on the

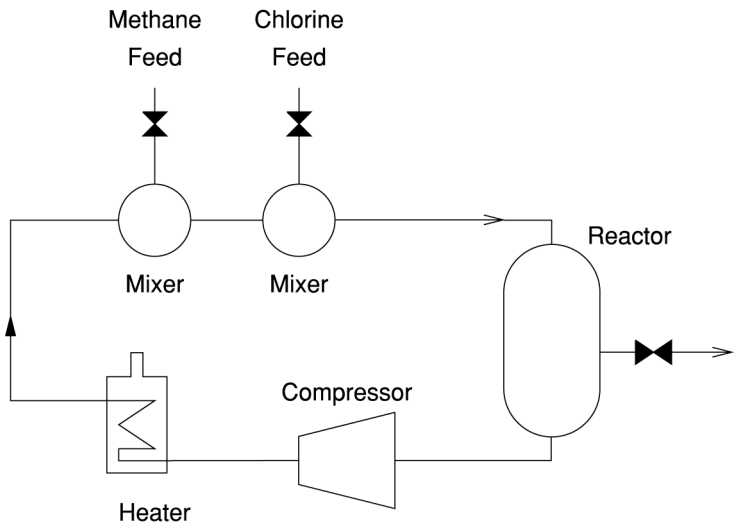

Figure 8 A plant to demonstrate Fusillo and Powers system

FP_Plan(*start*, *goals*, *action-templates*, *procedure*)

1. *actions* = FP_SelectActions(*goals*, *action-templates*)
2. return FP_OrderActions(*start*, *actions*, *procedure*)

FP_SelectActions(*goals*, *action-templates*)

1. if *goals* = ∅ return ∅
2. *g* = select any *g* from *goals*.
3. if *g* is true in start goto step 7
4. options = set of all op such that op is an instantiation of some action-template in action-templates and op achieves g.
5. if options = ∅ then fail.
6. action = choose an action from options.
7. return {*action*} ∪ FP_SelectActions(*goals* - *g*, *action-templates*).

FP_OrderActions(start, actions, procedure)

1. if actions = ∅ then succeed.
2. *act* = choose an action from actions such that all the preconditions of *act* are true in start.
3. *new* = determine the effects of *act* in the state *start*.
4. if ¬ safe(*new*) then backtrack.
5. call FP_OrderActions(*new*, *actions* - *act*, *procedure* + *act*)

Figure 9 Fusillo and Powers's OPS algorithm

number of steps in a procedure. Means–ends analysis is used to form the goals that are not yet satisfied. When a difference between the current state and goal state was found, a subgoal was produced to eliminate the difference. In the operation selection, the planner locates equipment that could be manipulated to satisfy each goal. For example, if one of the goals is to increase the temperature of a certain part of a plant, then a source of thermal energy is sought.

By performing action-template selection before synthesis, the system is less able to resolve conflict between actions. Consider the operating procedure shown in Figure 10. The procedure involves compressing a chemical but finishing with the chemical's temperature unchanged. The compressor heats up any chemical it operates on and so some form of cooling will be required during the

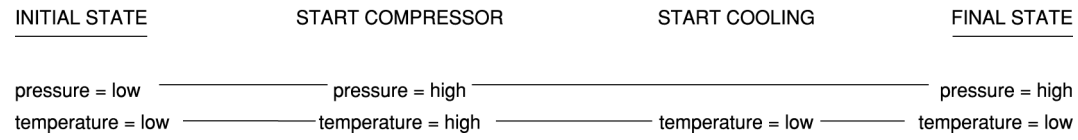

Figure 10 Actions may conflict

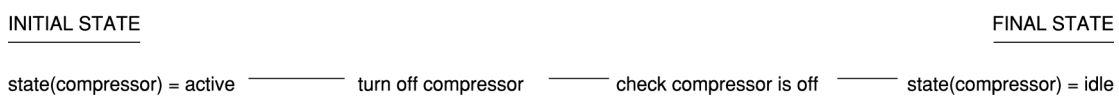

Figure 11 Some goals cannot be solved by a single action

procedure. However, since the chemical is cool at the start of the procedure, the action-template selection algorithm used by Fusillo and Powers's system will not propose a needed cooling step. Instead, the system will generate an incorrect procedure containing the single action "start compressor".

The action-template selection algorithm is also incomplete. If an OPS problem has exactly one goal then the problem cannot be solved if the goal is two steps from the start state. For example, the procedure in Figure 11 cannot be generated. This is similar to the limitation that prevents Rivas *et al.*'s planner from opening two or more valves to create a flow.

The method for verifying safety is similar to the method of safety used by Rivas *et al.* (1974). Unsafe situations are represented by LISP equations. These equations are called global constraints; a state is unsafe if any global constraint evaluates to true in any state produced by the procedure. If adding a new action to a procedure causes an unsafe state to be formed, the system simply backtracks. This approach is incomplete. For example, consider the plant system in Figure 12 and the goal of starting the heater. The heater cannot be started if the compressor is off but the system will not automatically propose the action to start the compressor because starting the compressor does not achieve a goal directly.

Fusillo and Powers (1988a) describe an extension to the action selection strategy in the earlier OPS system. The extended system can predict some of the actions that will be needed to resolve safety problems during synthesis. The strategy is based around creating a hypothetical state for an OPS problem. In this hypothetical state, the goals of the problem are true and any literal in the initial state which is not contradicted by the goal state is also true. Note that adding new goals to a problem changes the hypothetical state for that problem. For example, in Figure 12, if a goal were added to require the compressor to be on in the goal state then the compressor would be on in the hypothetical state.

The Fusillo and Powers planner will select an action for every difference between the hypothetical state and the initial state (every goal). Hence the hypothetical state can be viewed as the state that the system intends to achieve. Synthesis will usually fail if the hypothetical state is unsafe because the system is prevented from achieving an unsafe state.[3] The strategy used in Fusillo and Powers (1988a) is to examine the hypothetical state and add new goals to make the state safe. For example, if a heater cannot be turned on without a compressor being on and if the hypothetical state violates this safety

EQUIPMENT TO BE OPERATED	STATE		
	INITIAL	GOAL	HYPOTHETICAL
Heater	off	on	on
Compressor	off	-	off

Figure 12 The relation between the initial, goal and hypothetical states

[3] If no action in a procedure has a side effect then carrying out the procedure will result in the hypothetical state. Hence if the hypothetical state is unsafe then synthesis will fail given that no actions have side effects. However, if the actions in a procedure do have side effects then carrying out the procedure will produce some alternative state, rather than the hypothetical state. In some rare cases, this alternative state is safe, and so achievable, even though the hypothetical state is unsafe, and so unachievable.

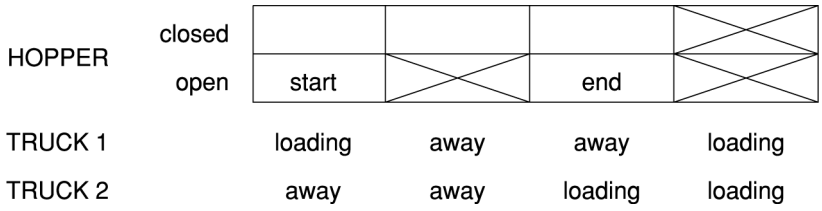

Figure 13 The Karnaugh map (state space) for the truck-and-hopper problem

constraint then the system will add a new goal to turn the compressor on given that the compressor is off in the initial state[4]. The strategy represents a significant attempt to allow a forward-chaining OPS system to work with safety constraints. The strategy has two limitations:

1. The system assumes that each action will have only one effect. The safety problems caused by the side effects of actions are not considered. For example, consider the situation in which a step is added to a procedure to purge a particular vessel with nitrogen; if nitrogen is not in the vessel at the start of the procedure then the system will not evaluate whether it is safe for nitrogen to be in the vessel at the end of the procedure.
2. In some situations the system will fail to propose all the actions needed in the creation of a safe procedure, even though the hypothetical state is safe and each action has only one effect and a safe procedure exists. Consider a simple domain involving a hopper full of gravel and two trucks. In the initial state, the first truck is being filled with gravel and the second truck is away. The goal is to have the second truck being filled with gravel and the first truck away. The safety constraints prevent both trucks being in the loading bay at the same time and also prevents the gravel hopper from being open while no truck is in the loading bay. The state space for this problem is depicted by the Karnaugh map (Karnaugh, 1953) in Figure 13. Adjacent cells in this map represent states joined by a single action. Crossed cells represent unsafe states.

The only safe solution to the truck-and-hopper problem is to close the gravel hopper, drive the first truck away, park the second truck and then open the hopper again. The difficulty for the system considered here is the need to operate the gravel hopper; it is not apparent from the start and goal states of the procedure that the hopper will need to be operated.

4.1.3 Subgoal-state

The work described in Tomita *et al*. (1989) and Hwang *et al*. (1991) takes a novel approach to plant modelling in order to create a more powerful OPS system. On the surface, this system appears to be very different to the work discussed so far. In practice the methodology is similar to Fusillo and Powers's approach in that it involves an action-selection phase and an action ordering phase. However, this work seems to have been developed independently of Fusillo and Powers's approach. As an example of the problem-solving ability of the system, consider the startup of the column in Figure 14. Safety constraints dictate that the heater cannot be started when the pump is off. Physical constraints prevent the column from being started before the heater is started. The resulting procedure is shown in Figure 15.

The action-template selection algorithm is based on a backward-chaining search. The basic algorithm involves selecting an action to achieve each goal and also each precondition of each action already selected. The algorithm used in the paper is more sophisticated than this description, and the reader is referred to the original document for further details. The basic algorithm may solve a goal in a way that prevents the solution to some other goal. The system described in the paper prevents conflict

[4] The work in Fusillo and Power (1988a) considers specifically the problem of adding purge operations to prevent dangerous mixtures of chemicals forming. The heater and compressor problem cannot be solved using the system described in the paper, but only due to the implementation of the methodology used in the system rather than the methodology itself.

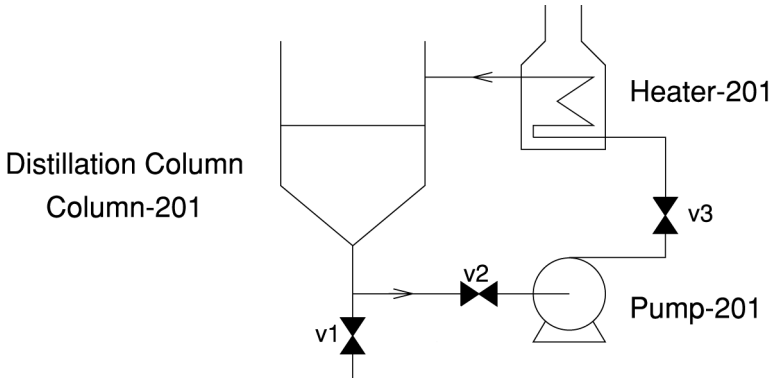

Figure 14 A forced reboiler shown with valves

1. Start the flow of oil to pump-201.
2. Start the flow of cooling water to pump-201.
3. Open v2.
4. Open v3.
5. Start pump-201.
6. Start the flow of fuel to heater-201.
7. Start heater-201.
8. Start column-201.

Figure 15 The procedure to start the reboiler

between actions by developing the idea of a subgoal state. The subgoal state is a consistent state in which the preconditions and effects of all currently selected actions are true. Actions are only selected if they are consistent with the subgoal state.

A simplified version of the action-template selection algorithm is shown in Figure 16. The full version of this algorithm is given in a preprint of Tomita *et al.* (1989).

The advantage of this action-template selection strategy is an ability to solve problems where a goal is more than one step away from the start state. Figure 11 demonstrates a problem that can be solved using this approach, but cannot be solved using earlier work.

The limitation of this action-template selection algorithm is that each unit can change state only once during a procedure. For example, the system cannot model a procedure where a vessel is unclean in the initial state and then cleaned during the procedure and finally used as the mixing vessel for a batch. The limitation is inherent in the way the subgoal state is used to develop a consistent strategy for operating the plant.

In this methodology, the plant is described as a directed graph in which the nodes represent most of the equipment in the plant while arcs represent pipelines and valves. Examples of nodes are reactors, columns and pumps. Valves and pipes are handled separately and are not counted as nodes. State information is associated to each arc to indicate qualitative values of the behaviour in terms of temperature, pressure, phase and components in the corresponding stream. Each node has information about the state of the equipment item, input and output arcs, conditions to operate the equipment, and a functional model of the equipment in the form of if–then rules. A PROLOG-like inference engine is used to perform backward and forward chaining. The state of the equipment item takes the values on

Backward_Planner(*goals*, *subgoal_state*, *functional_rules*)

1. if *goals* - *subgoal_state* = ∅ then succeed returning *subgoal state*.
2. *g* = select any *g* from (*goal* - *subgoal state*).
3. *rule* = choose a rule from *functional rules* such that the $g \in$ effects(*rule*). The effects of a rule are described by the 'then' part of the rule.
4. *new* = effects(*rule*)N preconditions(*rule*). The preconditions of a rule are described by the 'if' part of that rule.
5. if *new* is inconstant with subgoal_state (e.g. if $x \in$ *new* and $\neg x \in$ *subgoal state*) then backtrack.
6. if *new* is inconstant with goals then backtrack.
7. return Backward_Planner(*goals* ⋃ preconditions(*rule*), *subgoal state* ⋃ *new*, *functional rules*)

Figure 16 Action-template Selection used in work by Tomita and others

or off. The functional model of the equipment is used to determine the effect of a new state that results of the execution of an action.

The operation of a single node is represented by a sequence of plan steps in an operating procedure. The sequence involves performing preparatory work, such as obtaining permission to work, then opening the necessary valves and finally operating the node itself. The sequence is dependent on the desired effect of the unit and is generated by simply formatting stored knowledge about that unit. The algorithm for translating the operation of a node to a sequence of steps is possibly too simplistic, as it assumes that the route for the flow of chemical into a unit (node) will be fixed depending on the role of that unit. It is also assumed that the flow route will be clear of possible contaminants. The large amount of valve sequencing work in the OPS literature suggests that finding a safe route for a flow is the hard part of some practical problems. It is also not clear how this model of a unit can be used to represent batch processes involving a number of steps in one vessel. For example, consider modelling a process in a single vessel to make strawberry yoghurt from yoghurt culture by first making plain yoghurt from the culture. In this process, the vessel has one of at least three states depending on its contents.

4.1.4 Process-retrofitting approach

Aelion and Powers (1991) continued on the work started by Fusillo and Powers. The synthesis algorithm used in their work is shown in Figure 17, and is based on the STRIPS planning algorithm (Fikes & Nilsson, 1971).

The action-template selection strategy used in this system can be viewed as an elaboration of the action-template selection strategy used in Tomita *et al.* (1989). As in the earlier work, the system can solve problems where a goal needs to be solved by more than one action. However, the system can also solve problems where a unit must change state more than once during a procedure.

Line (c) of Figure 18 shows a procedure that can be generated by Aelion and Powers's system but not by any earlier forward-chaining OPS system. The procedure involves depressurising a unit. The procedure is required to achieve two conditions: that the pressure of the unit is atmospheric and that the bleed valve is closed. The procedure is difficult to generate for two reasons: (1) the solution to the two goals together requires more steps than the combined solution of the two goals individually and (2) the bleed valve changes state twice during the procedure. The features that make this problem difficult appear frequently in OPS domains.

AelionPowersPlan(*start*, *goals*, *action-templates*, *procedure*)

1. if *goals* ⊂ *start* then succeed.
2. *g* = choose a *g* from *goals* that is not in *start*.
3. *options* = set of all *op* such that *op* is an instantiation of some action-template in *action-templates* and *op* achieves *g*.
4. *planstep* = choose a plan step from *options*.
5. *added* = set of all preconditions from action which are not true in start.
6. *subplan* = AelionPowersPlan(*start*, *added*, *action-templates*, *procedure*)
7. *mid* = simulate the effects of *subplan* in the state start.
8. *new* = simulate the effects of *planstep* in the state *mid*.
9. if ¬ safe(*new*) then apply rules to repair plan safety.
10. call AelionPowersPlan(*new*, *goal*, *action-templates*, *procedure* + *subplan* + *planstep*)

Figure 17 Aelion and Powers OPS algorithm

The system will solve the problem if the goal to depressurise is chosen before the goal to close the bleed valve. The system tries all possible orderings of the goals of a problem, and thus the correct solution to this problem will be found; however, the system will also generate an incorrect solution. This OPS system, as described, does not check that its goals are met by all the plans it suggests. The incorrect solution will occur when the goal to close the bleed valve is considered first and will be similar to the plan steps in Figure 18a.

While the approach considered here is an improvement on earlier work, it is not complete. There is a well-known problem with the STRIPS planning algorithm that Aelion and Powers's system will fail to solve. The problem is known in the AI planning literature as the Sussman anomaly (Waldinger, 1977). A chemical engineering version of the Sussman anomaly is exhibited by the plant in Figure 19.

A difficult problem for this plant involves achieving a situation where the store tank is clean and a reaction is taking place in the bio-reactor. In the initial state for this problem, the store tank contains bacteria culture and the reaction vessel is cleaned. The reaction is started by placing the culture in the reactor and then continuously adding bio-food to the reactor. It is intended that the flow of bio-food should continue past the end of the procedure. The correct operation of the plant in Figure 19 involves moving the culture to the reactor, cleaning the store with cleaning water and then starting the flow of bio-food. Any other order of instructions would either destroy the culture when the store was cleaned or would interrupt the reaction in order to clean the store. This procedure is difficult to generate due

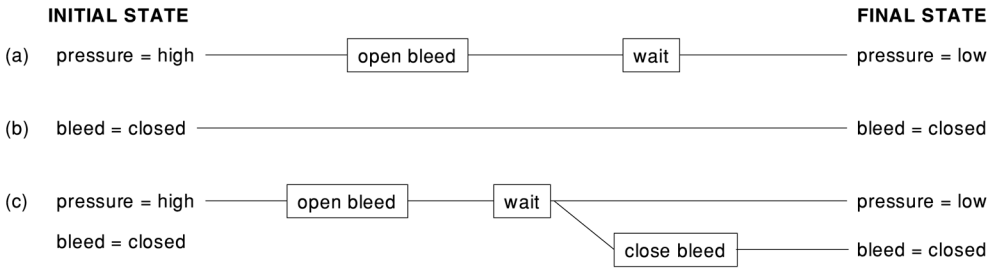

Figure 18 The procedure for bleeding a unit

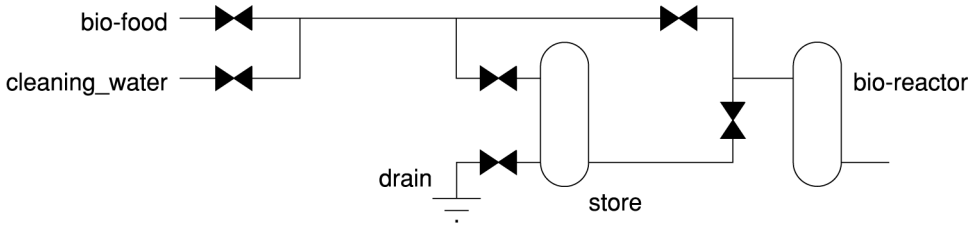

Figure 19 A plant on which to demonstrate the Sussman anomaly

to the fact that the two goals are interleaved. The system gets part way to starting the reaction, then cleans the store, then finishes starting the reaction. The system considered here assumes that goals will not have to be interleaved in this way and so cannot solve this problem.[5]

The model of safety used in Aelion and Powers (1991) is similar to the model of safety in Fusillo and Powers (1987). However, the system will add new preconditions rather than backtracking if an unsafe state is generated during synthesis. This handling of safety is similar to the handling of mixing constraint violations in Rivas *et al.* (1974). It is not complete because the actions to resolve a safety problem cannot be inserted into the portion of the operating procedure that has already been generated. Later work by Aelion and Powers (1992, 1993) moves away from the procedure synthesis problem to look at methods for evaluating the safety of operating procedures.

4.2 Action synergy approaches

The action synergy paradigm is concerned specifically with valve sequencing problems. The approach is not suitable for more general OPS problems. Forward-chaining OPS systems expect an action to have a fixed set of effects or, in the case of Tomita *et al.* (1989), a small number of different effects based on the state of the plant. Valve operations do not fit into this way of thinking; opening a valve will often cause one of a large number of different effects to occur, depending on the state of the plant. As a result, current forward-chaining systems have a limited ability to reason about valve operations.

Rather than reasoning about a valve as having a particular effect in a particular state, it is more constructive to reason about the effect of a set of valve operations. As an example, consider the pipe arrangement in Figure 20 and note that all valves are initially closed. Opening any one valve will have very little effect; opening the sequence of valves {a, b, c} will create a flow of oxygen. This flow will also have the side effect of contaminating some of the pipes in the plant with oxygen. All of these effects can be seen to result from synergy (interaction) between the three valve operations chosen.

Action synergy systems reason about the set of valve operations necessary to achieve some desired effect in a safe way. Action synergy systems usually have the following four components:

1. An algorithm to search for the sequence of valve operations needed to achieve some desired effect. For example, to create a flow of oxygen in Figure 20, a system must select and open either the valves {a, b, c} or the valves {a, d, g, e, c}. It may be necessary to backtrack this step. For example,

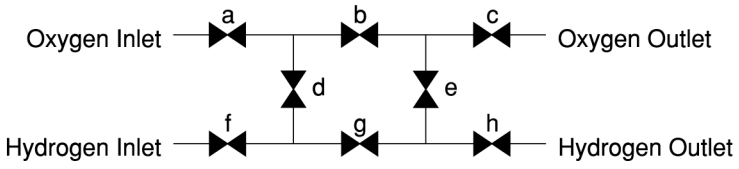

Figure 20 A pipe arrangement where valve sequencing is needed

[5] As a point of interest, the system described in Tomita *et al.* (1989) allows goals to be interleaved. Hence this earlier system is able to solve the version of the Sussman anomaly presented here.

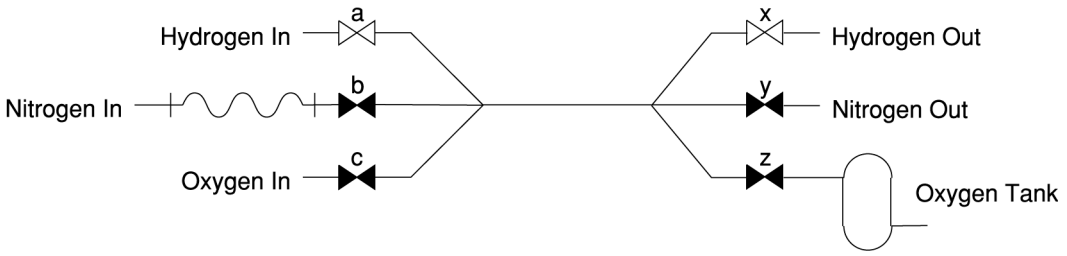

Figure 21 The bottleneck problem

if the system happens to choose the flow route {a, d, g, e, c} and if a flow of hydrogen is also required, then at some time the system will have to backtrack to choose the route {a, b, c}.

2. An algorithm to decide how to limit the side effect of a sequence of operations. For example, if a flow of oxygen is created by opening the valves {a, b, c}, then the valves {d, e} should be closed so as to limit the flow of oxygen.
3. An algorithm to order the selected operations. This algorithm can be based on simple rules. For example, in creating a single flow, all valve-closing operations should be performed before any valves are opened.
4. An algorithm to simulate the effects of a sequence of operations and to make sure that the sequence is safe.

Action synergy systems are limited in their ability to plan. These systems can be compared to a forward-chaining system by considering the sequence of steps needed to create each flow or purge as a single forward-chaining "action". Under this mapping, most systems operate like the forward-chaining system described in Fusillo and Powers (1987). For example, action synergy systems have difficulty when a goal is more than one "action" away from the start state. Also most, but not all, action synergy systems simply backtrack if a procedure is unsafe.

Figure 21 describes the start state for a procedure that no action synergy system can create. The goal is to achieve a state in which the tank is full of oxygen and hydrogen is flowing. To solve this problem, a system must address the goal of filling the tank before the goal of restoring the flow of hydrogen. Action synergy systems assume that all goals can be solved in parallel or that a goal ordering will be provided by the user.

4.2.1 Topology-based approach

O'Shima (1978) describes the first action synergy system – the only goal considered by the system is the task of creating a flow of chemical. A flow goal is solved by searching for a path from the given source to the given sink possibly via some specified intermediate components, perhaps a reactor. The search algorithm is similar to an algorithm for searching for a path through a maze. Actions are proposed to open the valves along this path and to close the valves around the path, so that the species does not stray. When a flow is created, chemicals are moved into new locations in a plant. The system is able to predict the affected plant units without resorting to the complex simulation used in Rivas *et al*. (1974). Essentially, the only units affected by a flow are the units along the flow path.

4.2.2 Default state of valves

Strimaitis (1987) and later Roach *et al*. (1990) use action synergy techniques to solve the case study from Rivas *et al*. (1974). An interesting idea from this work is that valves should have a default state; most valves in the plant default to closed. Procedure synthesis involves creating a flow, then closing all valves in the plant, then creating a second flow, and then removing the redundant valve-closing operations. Using this approach, the system minimises propagation of chemicals around the plant. When the system must maintain a flow of a chemical, the valves along the initial route are kept open. They can only be closed if some alternative flow route is found. Using this mechanism, the system is able to maintain a flow route during a procedure.

4.2.3 Interactive creation of operating procedures

Foulkes *et al*. (1988) provide a tool that is designed primarily to aid the plant action-template to create a correct operating procedure, rather than creating such a procedure for the operator. At each stage in a two-stage cycle, the user asks the computer to move a chemical from one part of a process plant to another. The computer then generates for the user all safe routes along which this chemical can flow. When a route has been chosen by the user, the computer updates its model of the plant and a new cycle starts.

As with O'Shima's (1978) system, this approach creates a flow of a chemical by finding a path through the plant from the given source point to the given sink. In contrast to earlier work, the system requires that a flow route contain at least one pump; the system will turn on this pump as part of creating the flow. The system will also order the actions involved in a flow of chemical so that, for example, all the valves in a flow are opened before any pump along the flow route is started.

Similar to Rivas *et al*. (1974), the system allows safety considerations to be specified. Safety is defined in terms of unsafe situations. To monitor the safety of a procedure, the system maintains the information about the current state of the plant so that safety constraints can be evaluated. No attempt is made to resolve a situation where a goal cannot be achieved because of the safety constraints on the system. Instead, the user is told why a particular goal cannot be achieved. In effect, it is the responsibility of the user to find some modification that will later allow that goal to be solved. Safety is evaluated in terms of units and their contents, but not in terms of flow. It is not clear that the system can be asked to maintain a flow of chemical; this may cause problems with the use of pumping operations. Although the system will turn on an appropriate pump in order to create a flow of chemical, it is not clear that the appropriate pumps will be turned off when a flow path becomes blocked.

The final point of interest is the idea of a normal path. There are often many flow routes that are possible in theory but undesirable in practice, e.g. a flow route that uses a reactor as if it were a simple pipe. Also, plants are often designed with the intention that the flow of a certain chemical should travel along a certain route and the plant operator is usually aware of this intention. The idea of a normal flow captures knowledge about the intended use of the plant and search is only used in situations where the plant cannot be operated in its intended way.

4.2.4 Qualitative mixing constraints

Lakshmanan and Stephanopoulos (1990) provide a method for adding purge operations to prevent unsafe mixtures of chemicals from forming. The allowable mixing constraints are described as being *binary* and *qualitative*. The term *binary* is used to mean that if three chemicals cannot mix then two of these three chemicals also cannot mix. The term *qualitative* is used to mean that two chemicals cannot mix in any proportions. To put this into context, most valve sequencing systems consider arbitrary qualitative mixing constraints but do not consider purge. The need for purge operations is found by examining a predicted worst-case state for a procedure. The worst-case state is defined as the state in which all new flows have been created and no existing flows have been blocked. Purge operations are added to clean the pipes that contain unsafe mixtures of chemical in the worst-case state. If a purge operation is found to be required, the system uses action synergy techniques to route the flow of purge through the area to be cleaned; after this purge has taken place the system will go on to try to achieve the objectives of the final procedure.

Many algorithms used in the system are shown to have polynomial time complexity. However, the time complexity of the system as a whole is left undecided. The time complexity of the algorithm is interesting because the approach claims to work in polynomial time; however, it is not yet clear that an OPS system can be designed to solve practical problems in polynomial time. A simple example based on the plant in Figure 22 demonstrates that the system will work forever when attempting to solve some problems. In effect, the time complexity of the system is infinite.

Consider the operation of the plant system in Figure 22. The plant may contain four chemicals: Chem-a, Chem-b, Chem-x and Chem-y. The pairs of chemicals which cannot be mixed are ⟨Chem-a,Chem-x⟩, ⟨Chem-a,Chem-y⟩ and ⟨Chem-b,Chem-y⟩. At the start of a particular OPS problem, Chem-a is flowing to the "Waste Stream Out" as shown in the diagram. The objective is to create a flow

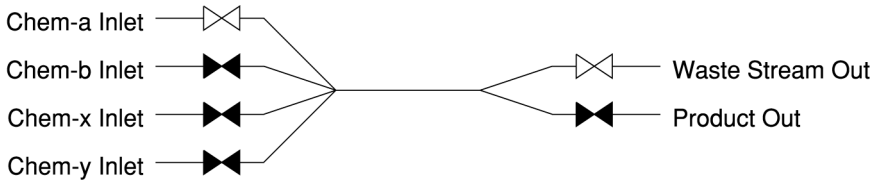

Figure 22 A difficult purge problem

of Chem-y to the "Product Out". One solution is to purge the common pipe first with Chem-b, and then with Chem-x, and finally to flow Chem-y. The difficulty with this problem involves choosing a suitable purge chemical. The need to purge with Chem-b in the first operation is apparent, but the system can then purge with Chem-x or Chem-a. The system always chooses the first available purge and so can be made to choose Chem-a rather than Chem-x. The cycle of purging with Chem-b then Chem-a will be repeated indefinitely.

4.3 Action-ordering systems

The OPS systems considered so far view action selection as perhaps the most difficult part of synthesis. Action-ordering systems focus on problems where the steps in a procedure are determined by the design of the plant. The philosophy is that a plant is designed to be operated in a particular fashion, hence the design of a plant should determine the equipment used for any particular operation. These systems work by selecting an action to bridge each difference between the start state and the goal state. There is only one action for each kind of difference and so no backtracking is required. The focus of the work is to order the action so as to satisfy the preconditions of each operation.

The action-ordering paradigm is limited in that each unit can have only two basic states: an idle state and an active state. The objective of synthesis is either to start up the unit or to shut down the unit or to leave the unit as is. Some situations cannot be modelled using this approach; for example, consider a procedure in which a vessel starts off containing air and then has to be purged with nitrogen and finally filled with methane and chlorine. This procedure cannot be modelled as a single action-ordering task because the vessel must go through three states rather than two. The main influences on action-ordering work are the forward-chaining systems described in Fusillo and Powers (1987) and in Tomita *et al*. (1989).

4.3.1 The augmented-object approach

Lakshmanan and Stephanopoulos (1988a, 1988b) describe an action-ordering system – the role of the system is to create ordered sequences of valve operations. The system does not consider the mixing of chemicals, and so valve operations never conflict with one another. Hence, without any safety constraints, the system could choose any order for a set of valve operations. However, the system allows safety constraints in the form of temporal constraints. A temporal constraint has constraints of the form 'action x must come before action y'. The synthesis problem involves generating a sequence of operations that satisfies these constraints.

This approach is developed around a non-linear planner built upon a hierarchical modelling environment. The system takes advantage of the plant structure for constraint propagation and subgoaling through a hierarchical structure. A group of equipment (composed of valves, pumps and other equipment) represents an augmented object in which constraints and states can be assigned to its inputs and outputs.

An improvement on the STRIPS operator is proposed by the authors through the introduction of a functional action-template that is used to operate a piece of equipment using multiple preconditions and multiple effects. The effects of the action-template in the rest of the plant are obtained by means of simulation models and an object-oriented representation of the plant.

The planning methodology introduces many ideas from the partial-order planning approach proposed by Chapman (1987). In constructing a partial plan, the augmented-object approach used

temporal constraints and Binary Qualitative Mixing Constraints (BQMCs) such as "A and B should not come into contact with each other". Temporal constraints are of the type (T≺S) such that T should precede S. These kinds of constraint are propagated from a top-level equipment conglomerate called augmented unit. Mixing constraints of the type BQMC are transformed into temporal constraints through an algorithm based on influence graphs which become abstracted representations of the plant topology.

Finally, for the synthesis of the complete plans, the approach uses transition networks. A transition network is a directed graph consisting of nodes that represent states and edges that represent the action templates such as open(valve-1) that take the system from one state (node) to another (node). Once the transition network for a partial plan has been constructed, any path leading from the initial state node to the final state node is a candidate plan. Search algorithms (such as depth-first) are used to find a path that does not violate any constraint.

The system is based around the use of an intelligent user interface. The role of the user interface is to elicit the start state, goal state and temporal constraints involved in a problem. The user interface is intelligent in that information can be specified at any level of abstraction and is translated automatically into the level of valve operations. Actions are proposed for each difference between the start state and the goal state. For example, if valve-1 is open at the start and closed at the end then an action 'close valve-1' would be proposed. The set of actions is then ordered through constraint posting using a partial order plan representation; the system never backtracks and never needs to do so.

Difficulties arise because high-level ideas are translated down to low-level ideas before planning begins. Sometimes the translation process will be ambiguous; for example, there may be many different sets of valve operations that will achieve a given flow goal. It is the responsibility of the user to choose the correct interpretation for any ambiguous request, e.g. to decide the correct flow path; the decision is not backtracked by the system. The system is also limited because in each problem, each valve may change state only once. To work around this problem, the system encourages the user to represent complex problems by a sequence of simpler subproblems. For some processes such as catalyst regeneration operations, this planner (based on means-ends analysis) may not be able to solve the problem where the initial state and the goals are, in terms of the state representation, the same. Therefore manually dividing the goals into more specific subgoals becomes necessary.

4.3.2 Interleaved simulation and action-ordering

Alsop and Macchietto (1995) describe an extension of the approach used in Lakshmanan and Stephanopoulos (1988). The extension provides a new ordering algorithm that is able to deal with more general safety constraints. The extended work also looks at actions to operate machinery as well as to open and close valves. These actions are implicitly assumed not to have side effects that would conflict with the desired effect of earlier actions. However, the actions may have preconditions that must be satisfied before they can be operated.

Action-ordering, the focus of the paper, works on a two-stage cycle. All actions that have satisfied preconditions are selected in the first stage. These actions are then simulated. The system then starts the cycle again by identifying and selecting the actions that are applicable in the new state. This approach is similar to the one in Fusillo and Powers (1987) without backtracking. Backtracking is not required because of the simplicity of the actions used.

4.4 The CEP approach

In the Chemical Engineering Planner (CEP) (Soutter & Chung 1995; 1996), the operating procedure synthesis task is modelled as the sum of three functions: planning, valve sequencing and hazard avoidance.

The planning function delegates flow-related goals to the valve-sequencing function. The valve-sequencing module returns a set of lower-level goals involving the operation of valves and pumps. CEP implements a STRIPS-based, non-linear planner, based on a partial-order planning algorithm similar to the approach implemented in the UCPOP planning system (Weld, 1994; Penberthy & Weld, 1992).

```
Operator StartReboiler
{
     heater ?boiler1;
     pump ?pump1;
     column ?c;

     expand
          * reboiler-state of ?c is on;
     using
          heater of ?c is ?boiler;
          pump of c? is ?pump;
          state of ?boiler is on;
          state of ?pump is on;
     end
}
```

Figure 23 An expanded action template in the CEP language

Two kinds of action template are defined in CEP: the achieve-action template and the expanded-action template. The achieve-action template has preconditions and effects as in STRIPS operators. On the other hand, the expanded-action template specifies a global goal and a set of equivalent low-level goals (Figure 23).

Valve sequencing is the task of creating or blocking a flow of chemicals by using a sequence of pump and valve operations. The approach implemented in CEP is based on action synergy techniques similar to those described above (Section 4.2). The plant topology is represented using the QUEEN qualitative modelling language (Chung, 1993). The QUEEN model describes the connectivity between equipment items and the relations between the state of a valve and the flow across the valve.

Hazard avoidance is the task of preventing unsafe or otherwise undesirable situations from occurring during a plan. This is handled by a method based on restrictions that prevent actions being incorporated into the operating procedure to cause a dangerous situation.

Constraint violations are solved by adding new actions that negate the effects of one of the actions in the plan that violate the constraints. The new action can be obtained by adding new preconditions to an existing action. Alternatively CEP can resolve a violation by adding constraints to the variables in the action that is already in the plan.

Hazardous constraints are handled in CEP in terms of restrictions. Restrictions are global constraints that represent situations that should be avoided. A simplified restriction in the CEP language is shown in Figure 24. This restriction specifies that starting heat_exchanger1 before pump1 should be avoided.

The overall planning methodology is shown in Figure 25. CEP first selects a goal from a goal agenda. The goal is either solved by the planning function or is delegated to the valve-sequencing function. After a goal is solved, a conflict-resolution algorithm is used to ensure that one action is consistent with the effects of previously added actions. Subsequently, CEP starts the hazard avoidance function to resolve any violation to the safety constraints.

In summary, CEP presents the following strengths:

```
restrictions
{
     prevent
     state of heat_exchanger1 is on;
     state of pump1 is off;
}
end
```

Figure 24 An example of restrictions in the CEP language

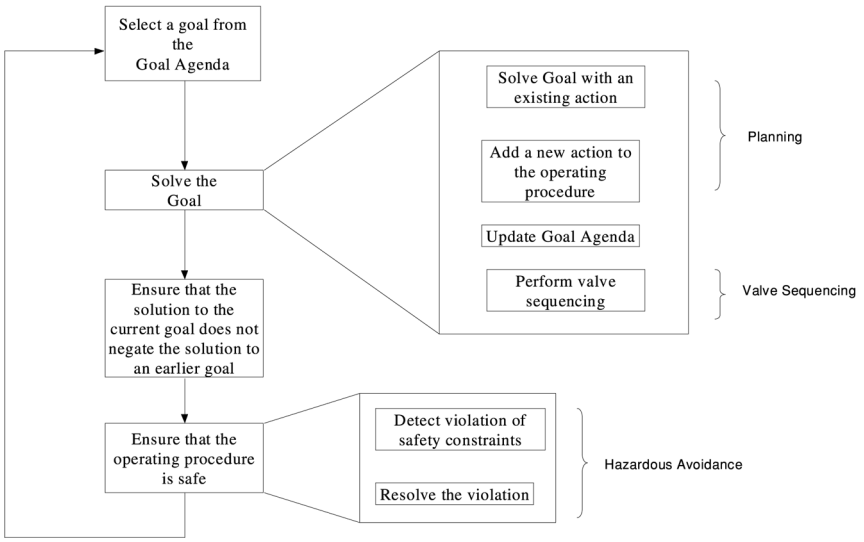

Figure 25 The CEP planning methodology

- CEP is the first OPS system to combine technically complete solutions to three OPS problems, namely planning, valve sequencing and hazard avoidance. CEP is actually one of the first systems to properly address any two of these three issues. As a result, CEP can solve complex, practical problems that were not addressable by earlier systems.
- CEP embodies an architecture for integrating planning, valve-sequencing and hazard avoidance components. The architecture is based around the non-linear, partial-order plan representation that is well studied by the AI planning literature. The architecture is flexible enough to allow sophisticated algorithms to be used for each of the three chosen OPS sub-tasks.
- CEP proposes a new solution to the hazard avoidance problem. The solution includes a plan-reasoning algorithm that is able to reason about the often vague partial-order plan representation. The solution also includes an algorithm to repair hazardous plans by adding new steps, such as purging a pipe, or by constraining the plan to avoid the hazard. Both algorithms are largely based on Chapman's modal truth criterion and are likely to be correct and complete although this has only been demonstrated empirically.

CEP has been applied in a number of case studies (Aylett *et al*., 1997; 2000). The limitation of the CEP system is the initial assumption that OPS is just planning, valve sequencing and hazard avoidance. Other tasks may also be relevant to OPS, particularly resource allocation (Crooks & Macchietto, 1992) and numerical plant-modelling (Fusillo & Powers, 1988b).

The CEP architecture supports the use of new techniques from the AI planning literature because the architecture uses a standard AI planning representation of a procedure. The planning literature includes work on resource allocation (Wilkins & Desimone, 1994) and on other techniques that may be relevant to OPS. As a result, the CEP architecture provides opportunities to address new problem areas.

4.5 Simulation-based planning

In simulation-based planning, process behaviour is described through simulation models that are represented separately from the action templates. Dynamic process behaviour models provide a planner with the ability to reason about time. Therefore it is possible to determine which actions can be carried out in parallel, as well as to provide information about the duration of the application of an action (the time needed to keep a valve in a certain position). Furthermore, optimum solutions can be obtained with the use of quantitative simulation models and quantitative constraints. Table 1 shows the differences between state-based and simulation-based planning approaches.

Table 1 The differences between state-based and simulation-based planning

	State-based planning	Simulation-based planning
process behaviour	behaviour is modelled as qualitative relations that are embedded in the effect of the action templates	process behaviour is described as simulation models
constraints	qualitative	qualitative, quantitative
parallel actions	plans are sequential (parallel operations are not possible)	parallel operations are possible
optimal solution	focus on feasibility	optimal action sequences are possible
time reasoning	time is not considered	reasoning about time is possible with dynamic models

4.5.1 Multi-agent based planning

This method is a simulation-based planning approach in which operations are planned and executed by a number of autonomous entities (agents) organised hierarchically. In order to improve the flexibility in the operations and to supply local actions with global knowledge, operations are organised in three hierarchical levels: plant, CGU and equipment levels (Figure 26). At the plant level, operations are carried out to control the interactions with contiguous plants or battery limits. Plant-level agents determine plant-wide goals that are communicated to agents in the lower hierarchies.

The plant is divided into smaller units called CGUs (Controlled Group Units) that are operated by agents who act cooperatively and report to the plant agent. A CGU can be identified from the flowsheet topology as an assembly of pieces of equipment surrounded by *Remotely Actionable Equipment* (RAE),[6] i.e. equipment that can be controlled from the control room such as control valves (Figure 27). An algorithm for the generation of the CGU topology is described in Figure 28. The overall plant is defined as a CGU that is made up of other CGUs.

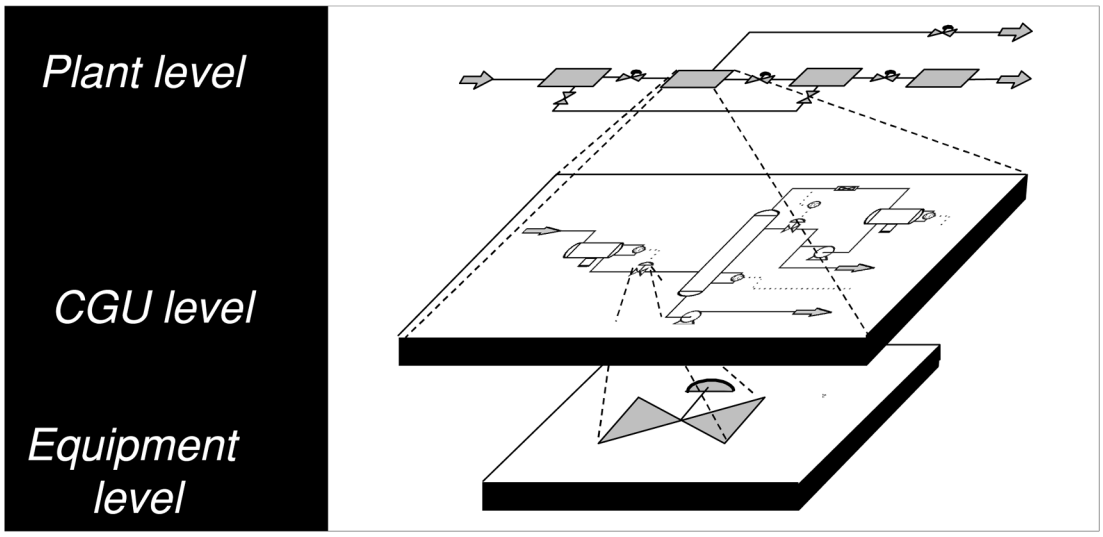

Figure 26 Plant hierarchical levels

[6] This definition extends the original CGU concept proposed by Naka (1989) and Naka *et al*. (1997) where the limits of a controlled group unit are control valves.

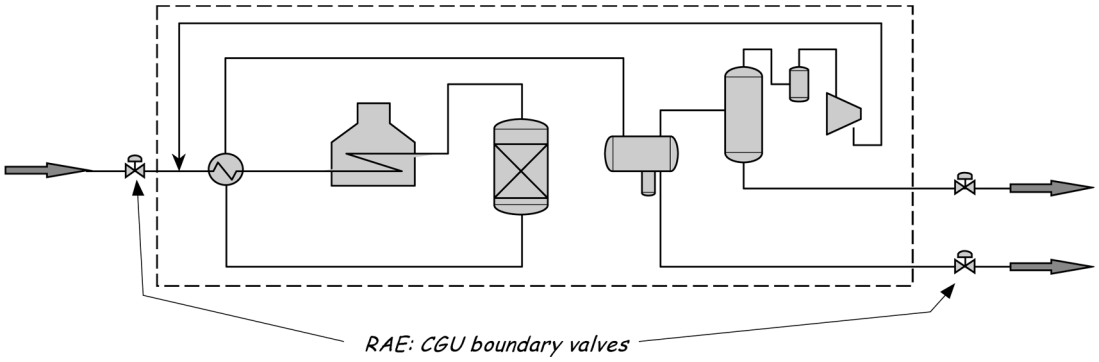

Figure 27 Controlled group unit

At the CGU level, a CGU is responsible for controlling its inventory of material by means of sending commands to agents at the equipment level that operate its input and output valves as well as internal equipment items. Bypass lines that are internal to a given CGU can be used to create stationary states (Fusillo & Powers, 1987), which facilitate bringing the plant to the final state. Operations at the equipment level are the manipulation of actionable equipment items such as valves and pumps. The operation of RAEs is constrained by conditional logic that specifies process constraints. These process constraints indicate to a planner whether the valve can be operated based on requirements of the process. For example, the condition "If PIC-06 > 75 Kgf/cm^{2}G and TY-28 > 175 °C then conclude that the condition-for-open of FCV-01 is true" indicates that the pressure as measured by PIC-06 should be more than 75 Kgf/cm^2 and that temperature at sensor TY-28 should be more than 175 °C to open valve FCV-01.

The synthesis of operations for startup and shutdown is carried out according to an inventory-control algorithm. Operations are performed by two operation components: the plant-inventory manager, and the CGU-inventory manager. The pseudo-code of the inventory-control algorithm is shown in Figures 29 and 30.

The plant-inventory manager operates control valves connected at the input or output ports of any plant Pl. In addition, the plant-inventory manager is responsible of the operation of bypasses and recycle lines that exist between CGUs, each of which is a sub-component of Pl. Valves that are connected at the output ports of the plant are operated jointly with the CGU-inventory manager. The CGU-inventory manager is responsible for operating valves that are connected at the input or output port of any CGU but are not connected at the input port of the plant to which the CGU belongs. Similar to the plant-inventory manager, the CGU-inventory manager operates recycle lines that are internal to each CGU. Information about the operation performer that executed the operation is recorded in order to coordinate the operation of the valves. In addition, the operations performed by the plant-inventory manager receive a higher priority than those performed by the CGU-inventory manager. This is necessary to avoid conflictive operations that try to manipulate the same valve. In cases when the operation of a control valve is conditioned by the current state of certain process variables, a valve cannot be operated until the process constraints are satisfied. In addition, the actual operation of the valve depends on the hold-up state of the CGU that is connected upstream of the valve, and the hold-up state of the plant. Consequently it is necessary to coordinate the decisions taken locally by CGUs about operating their valves with those taken globally by the plant. For this reason, performers are classified in four categories:

1. plant-inventory manager,
2. controlled group unit upstream of the valve,
3. controlled group unit downstream of the valve, and
4. controlled group unit for which the valve is internal.

Whenever a valve is operated, information about the operation category of the performer that operates the valve should be registered so that a performer that intends to operate the same valve can determine

```
algorithm generate-cgu-topology()
begin
     T ← find-equipment-path();
     repeat
          P = pop(T);
          m = the_first_element_in_the_path(P); // m is a valve
          n = the_last_element_in_the_path(P); // n is a valve
          o = P.cgu_name; // o is the name of the CGU
          e = (m, o); // e is an edge that connects m with o
          CGU_G ← push(CGU_G, e); // CGU_G is the CGU topology
               f = (o, n); // f is an edge that connects o to n
               CGU_G ← push(CGU_G, f);
     until P={}
end

procedure find-equipment-path
// Procedure find-equipment-path is used to generate all paths that
// links two control valves (only the first and the last element of
// the path are allowed to be control valves).  A path is defined in
// the graph ontological model as an ordered list of consecutive nodes
// in a graph. The resulting paths are stored in T and those paths
// with common equipment items are located and the topology is created
// with the first and last element of the paths and the name of the
// CGU to which they belong.
begin
     G ← {e = (m,n)|m ≠ n}
     C ← a list of control valves

     for {(c, d)| c ≠ d, c ∈ C, d ∈ C}
          U ← mbfs(c, d); // mbfs does a breadth-first-search
                          // that produces a set of paths U of
                          // the form:
                          // U = { P1, P2, …, Pn-1, Pn } where
                          // Pi = { (p1, p2, …, pm-1, pm) |
                          //        p1 = c, pm = d, p2,…,pm-1 ∉ C}
                          // where c is a control valve
          S ← push(S, U); // combines the sets of paths by
                           //pushing U onto S
     endfor
     Q ← quick_sort(S); // organizes the paths in decreasing
                        // order of length
     repeat
          Pn ← pop(Q);
          R ← push(R, P1);
          if P1.cgu_name = NIL then P1.cgu_name ← CGU_NAME();
          T ← push(T, P1);
          S ← find_subpaths(Q, P1);
          repeat
               P2 ← pop(S);
               P2.cgu_name ← P1.cgu_name;
               T ← push(T, P2);
          until S = {}
     until Q = {}
     return T
end
```

Figure 28 Generation of the CGU topology

whether it can actually operate that valve. When the process constraints associated to a control valve are not satisfied, a CGU-inventory manager uses an alternative path if such a path exists. It is assumed that these alternative paths are auxiliary lines that are added after a preliminary evaluation of the startup or shutdown of the plant.

4.5.1.1 Execution-model representation. In this approach, the resulting operating procedure is a conditional plan where each plan step is conditioned on observations of the plant and process state prior to their execution. This approach is similar to that of Peot and Smith (1992). The structure of the plan step is shown in Figure 31.

The *design rationale* of the plan step describes the arguments that support the plan step. Performer information describes the *agent* responsible for executing the action or the actual performer that executes the plan step. The *state* of the plan step takes the values of running, holding, paused, complete and stopped. A plan step is running only when its precondition is satisfied and the state of the plant and process is changing towards the *goal* of the plan step. The holding value applies when the state of the world is deliberately put into an intermediate state between the state before the application of the activity and the state resulting from the completion of the activity (e.g. a stationary state of a plant created between a shutdown state and a steady state of a chemical process). An activity in the paused

```
Algorithm inventory-control
INPUT_VALVE_LIST;
begin
     LIST_OF_INPUT_VALVES = the input_valves of CGU;
     LIST_OF_PATHS = the paths between boundary valves of CGU;
     while LIST_OF_INPUT_VALVES is not empty do
          VIN = pop the first valve from LIST_OF_INPUT_VALVES;
          while LIST_OF_PATHS is not empty do
               P = pop the path in which the first element is
               valve VIN from LIST_OF_PATHS;
               VOUT = the last valve of P;

               if VIN is a control_valve and VOUT is a
                    control_valve then begin
                    HOLD_UP_IN_PATH = check_hold_up_in_path(P);

                    if the operation_state of the first
                         actionable_equipment_item upstream
                         of VIN = on then begin
                         if the last operation on VIN was not
                         executed by the plant_inventory_manager
                         then
                            operate_equipment(VIN, on, CGU);
                         if HOLD_UP_IN_PATH is false and
                            the operation_state of VIN is on then
                            operate_devices_in_path(P, on, CGU,);
                         if HOLD_UP_IN_PATH is true then begin
                              operate_output_valve(VOUT, on, P,
                              CGU, PL);
                              if output_valve cannot be operated
                              then operate_equipment(VIN, 0, CGU);
                         endif
                    endif {if he first actionable_equipment_item…}
               endif
          end
     end
end
```

Figure 29 Inventory control algorithm

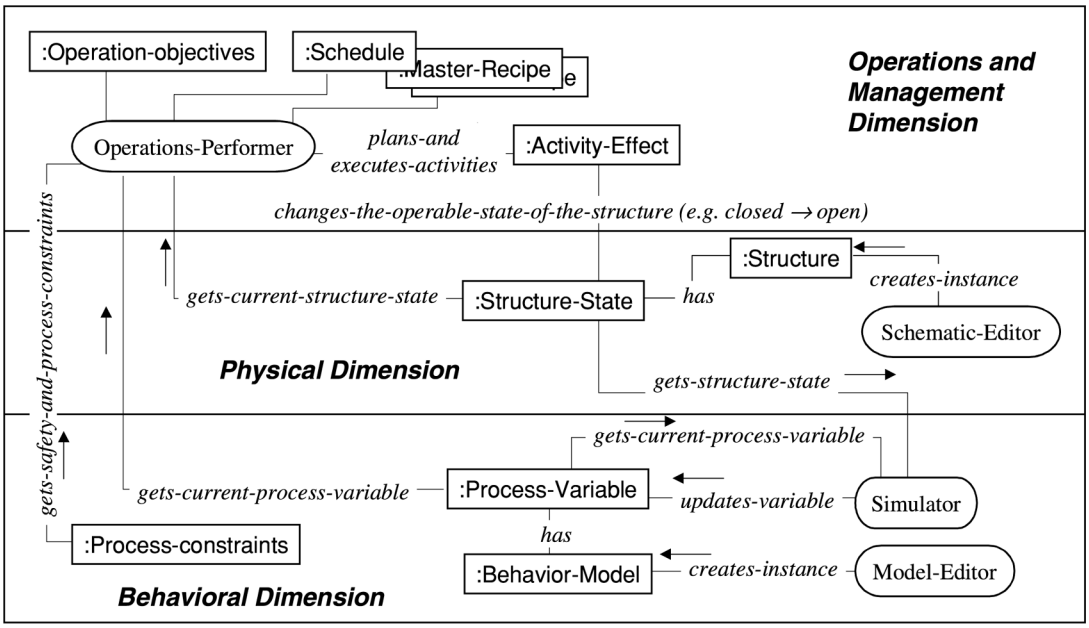

Figure 30 Information flow between the CGU and plant operation performers

state stops briefly and then is ready to change to a running state. The complete state is obtained when the activity purpose is satisfied. Action effects define the manipulation of an equipment item or the execution of other plan steps.

As an extension of the control recipe for batch processes defined in the ISA/S88 standard (ANSI/ISA, 1995), an operating procedure is represented as a multi-level hierarchy of procedural plan steps. On the top of the hierarchy, the *procedure* is intended to operate the plant or part of it. For continuous processes, a procedure describes the startup, shutdown, emergency and maintenance procedures. For batch plants, a procedure defines how to carry out a production process; an example of such a procedure is "make PVC". A procedure is made up of one or more *unit procedures* that are assigned for a group of equipment or for the whole plant. Examples of unit procedures are "prepare" and "react" in "make PVC". A unit procedure is made up of *operations* that are defined to manipulate one or more equipment items such that processing can be safely suspended. In the lower level of the hierarchy,

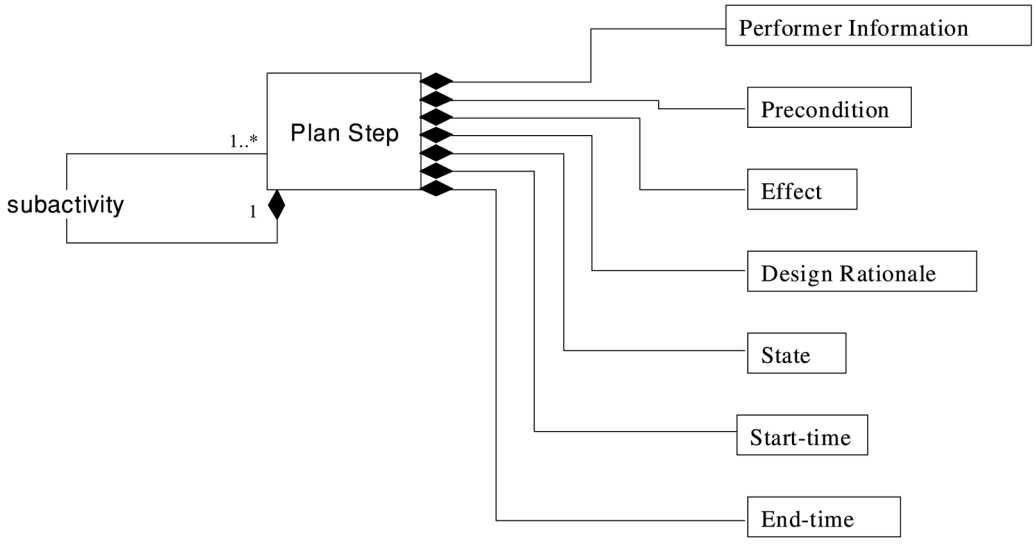

Figure 31 Plan step representation

phases act on specific process or control devices. A phase can modify the set point of a controller, send commands to the basic control system to operate a valve, or send a command to other phases. Examples of phase are "activate controller LIC-101", "open valve V-101". Another example is a phase that modifies the state of a multi-stage compressor from off to on. The execution of this phase results in a series of other phases that switch on each stage of the compressor.

4.5.2 Simulation-based planning with quantitative constraints

As has been mentioned, classical work on OPS draws on developments in the artificial intelligence field. Either linear planning (Fusillo & Powers, 1987; 1988) or non-linear planning (Lakshmanan & Stephanopoulos, 1988a; 1988b; 1990) approaches have been applied to chemical engineering problems. Given the fact that AI search mechanisms are well known to suffer from combinatorial explosion in larger-sized problems, the non-linear planning approach is conceptually more appealing, as it attempts to mitigate the combinatorial aspect of the problem. However, these past approaches are fundamentally limited in their application because they only handle *qualitative* constraints on the path between system states (e.g. "chemicals A and B cannot be mixed in the unit together"). In addition, as pointed out by Li *et al*. (1997), operating procedure synthesis in the late 1980s focused on plan feasibility without due consideration of optimising time or cost. Only recently have researchers been focusing on incorporating such *quantitative* constraints as "chemicals A and B should be mixed in such a way so as to avoid the flammability region of concentrations" (Li *et al*., 1997; Lee *et al*., 1993; Galán & Barton, 1997).

Li *et al*. (1997) introduced a Non-Linear Programming (NLP) approach to search for an optimal process operating path. However, in this attempt, the time of application of action templates (or the operator-time constant as defined in this paper) is not explicitly accounted for. In addition, the time horizon was not properly discretised to allow the NLP routine to guide the process along a possibly time-optimal and feasible path. Furthermore, solving the problem in the "continuous" domain may produce unrealistic actions. For example, at some point during the optimisation, the NLP routine may require one valve to be open 0.05% for 0.3s – an action that is not physically realisable. Mathematical representation of quantitative safety constraints was not discussed by Li *et al*. (1997).

In more recent work, Galán and Barton (1997) approach the OPS problem using a mixed-integer hybrid discrete/continuous framework, as an optimal control or dynamic optimisation problem requiring using control vector parameterisation (also known as an MIDO problem; see Bansal, 2000). However, as they are quick to point out, such an approach cannot guarantee a globally optimal solution, having to settle for plan feasibility only. Galán and Barton (1997) present an example application of their methodology using a mixture control problem very similar to the one presented here, involving a tank change-over operation from oxygen to methane by using nitrogen as a diluent.

Asprey *et al*. (1999) took an approach to explicitly account for time optimality, while maintaining realisable actions. For solving the problem, they discretised the time dimension, however not in an overall a-priori fashion – the actual number of time steps are not known a priori. Discretising the problem allowed representation of the system using evolving "states", each represented by a state vector. As actions are applied to the system, new state vectors describing the system evolving through time are calculated using dynamic simulation. A nested optimisation scheme was proposed, consisting of two distinct layers:

(i) Layer 1 – a layer that optimises the overall operations time by adjusting the "operator time constant" – the time duration for which an action is applied to the system. Thus the following optimisation problem is solved at this stage:

$$\min_{\Delta t} t_f \tag{2}$$

subject to

$$\Delta t > \varepsilon \tag{3}$$

where t_f is the total operational time to take the system from the initial state to the goal state, Δt is the operator time constant, and ε is a pre-determined realisable lower limit for the system operators, and is a function of the performance capabilities of the control valves used in practice.

(ii) Layer 2 – a layer in which a path-constrained optimisation minimises the difference between the current state vector and the goal state vector by executing actions. In this fashion, the two objective functions are optimised in a sequential, but iterative manner. The following optimisation problem is solved at this stage:

$$\min_{\delta_i(n)} \| p_{j_0} - p_{j_{GOAL}} \| \tag{4}$$

subject to

$$\phi(p_1(n), p_3(n)) \leq 0 \tag{5}$$

$$\delta_i(n) \in \{0, 1\}$$

where p_j and $\delta_i(n)$ are state variables of the system, and $\phi(\,)$ represents a functional form of a flammability constraint. The path constraint introduces non-convexity into the objective function, thus a globally optimal AI search mechanism was used.

Optimisation techniques for Layer 1 include simply bounded *global* NLP algorithms (operator time constant ≥ a physically realisable valve time). For this continuous simulated annealing was used. Optimisation techniques for Layer 2 include AI search algorithms – including brute-force methods such as depth-first or breadth-first searches and more elegant methods such as *A**, iterative-deepening *A** and depth-first branch-and-bound (Korf, 1998; McAllester, 1994). The overall algorithm is as follows:

1. Choose an initial value for Δt in Layer 1.
2. Pass Δt to Layer 2, and attain the goal state by applying a sequence of actions to the system determined by the *A** search technique.
3. Once the goal state has been reached, compute the overall operations time (t_f) from the sequence of actions.
4. Pass the overall operations time (t_f) to Layer 1 for the objective function value.
5. Change Δt via the simulated annealing method, and return to step (2).
6. Iterate steps (2) to (5) to convergence.

A good example of OPS in the presence of quantitative safety constraints is the synthesis of a sequence of valve operations to control the concentrations of chemical species, while avoiding a flammability limit during a change-over or startup/shutdown process. Such an example can be found in Galán and Barton (1997) for a $N_2/O_2/CH_4$ mixture change-over, and also in CCPS (1995) for an air/steam/propylene mixture during reactor shutdown for acrylic acid. In either case, the mixture control set-up can be depicted as in Figure 32. Asprey *et al*. (1999) used the CCPS example, for the case of a process startup. Feed-lines for air, steam and propylene were present, each with its own control valve (assumed to behave ideally). The initial conditions for startup were atmospheric pressure, with the vessel filled with air.

For illustrative purposes, the quantitative safety path constraint and a fictitious optimisation solution scenario are shown in Figure 33. The percent by volume of air within the vessel is given by (100 − %steam − %propylene).

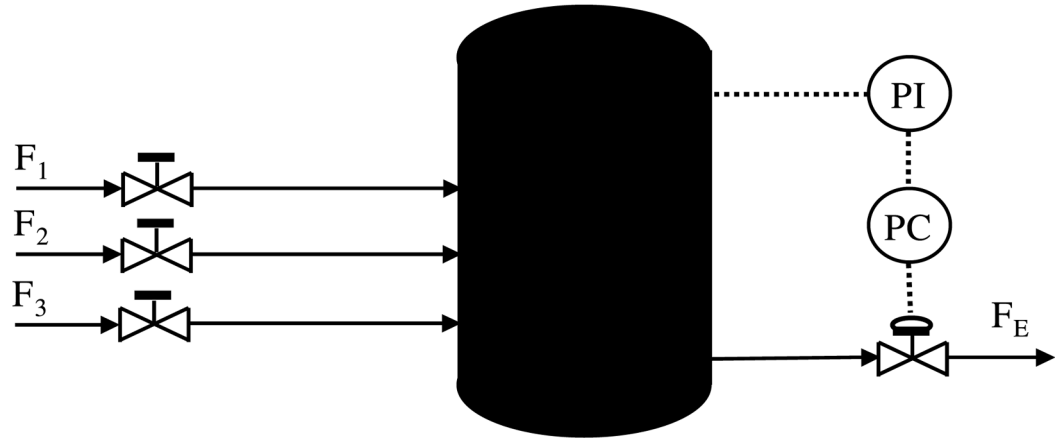

Figure 32 The Mixture Control Problem – showing a pressure vessel, three feed-lines with valves, and an exit line with valve and an on/off pressure controller

The describing equations for dynamic simulation can be written in terms of partial pressures of the individual components as:

$$\frac{dp_j}{dt} = \frac{RT}{V}(\delta_i N_i - \delta_E N_E) \tag{6}$$

$$\sum_j p_j = P_T \tag{7}$$

$$\delta_i,\ \delta_E \in \{0,\ 1\} \tag{8}$$

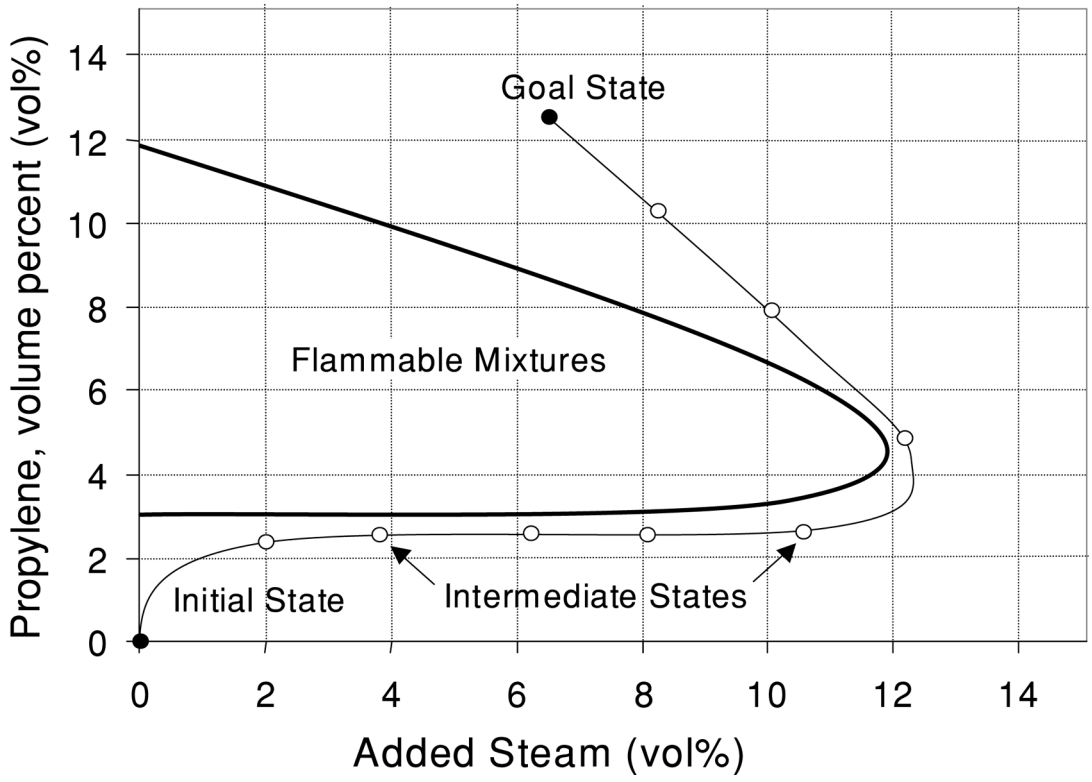

Figure 33 Planning in the presence of quantitative safety constraints. Shown here is the flammability path constraint, and an illustrative optimisation solution path

Table 2 The Mixture Control System Values

System Variable	Value
T	500 K
V	50.0 m^3
N_i	0.5 mol/s

where p_j represents the partial pressure of component j, P_T is the total system pressure, R is the universal gas constant, T is the temperature of the tank, N_i is the molar flow rate of the i^{th} feed-line (N_E for the tank exit line), and δ_i is the binary integer representing the state of valve i. In this model, we have assumed isothermal conditions, ideal gas mixtures, and that the control valves can be operated in a bang-bang fashion. The vessel pressure is controlled at a pre-specified set-point using δ_E by a simple on/off controller, while feed-line pressures are assumed to be high enough such that no back-flow through the system occurs. A slightly more detailed model for the $N_2/O_2/CH_4$ mixture change-over system, derived in terms of mole fractions,

$$y_j = \frac{p_j}{P_T} \tag{9}$$

can be found in Galán and Barton (1997), where non-ideal valve responses (opening and closing speeds) are also accounted for. Table 2 shows the system values used in the Asprey *et al*. (1999) study.

In this framework, the OPS problem is most easily represented in a state-space (Lakshmanan & Stephanopoulos, 1988a; Fusillo & Powers, 1987) or problem-space. The vector of component partial pressures represents a state. A state-space operator acting on the system represents a manipulation of a valve. In a state- or problem-space graph, the states of the space are represented by nodes of the graph, and the operators by (directed) edges between nodes. The sequence of operations for establishing desired process behaviour is then represented as a path between the initial and goal states. A graphical representation of the valve action time profiles for the time-optimal solution is shown in Figure 34.

4.5.3 Representation of quantitative safety constraints – the least-squares spline

Quantitative safety constraints in chemical processing systems are formed on the basis of experimental data, collected under small-scale, well-controlled environments. Mathematical representation of these constraints to a satisfactory degree of accuracy can be difficult, especially in the case of noisy experimental data.

Galán and Barton (1997) express the flammability limit or path constraint using a simplified logical proposition. However, it should be pointed out that such a representation tends to be too simplified. From the problem description depicted in Figure 28, it is obvious that an optimal OPS solution will occur when the mixture within the holding tank is kept close to the path constraint. Thus the more accurately the boundary is mathematically represented, the closer to true optimality the solution will be.

In Asprey *et al*. (1999) the authors represented the quantitative safety constraints by using a least-squares *B*-spline representation based on the work of Dierckx (1982). Least-squares *B*-splines retain the underlying, non-linear trend of the experimental data, without succumbing to the experimental noise. Subsequently these spline representations have continuous derivatives, making them ideal for use within an optimisation framework. As outlined by Dierckx (1982), all spline-based methods require the specification of the degree of polynomial, as well as the smoothing factor to control the trade-off between closeness of fit and smoothness of fit. If nothing is known about the variance of the data to be approximated, the smoothing factor must be determined by trial and error. Adaptive

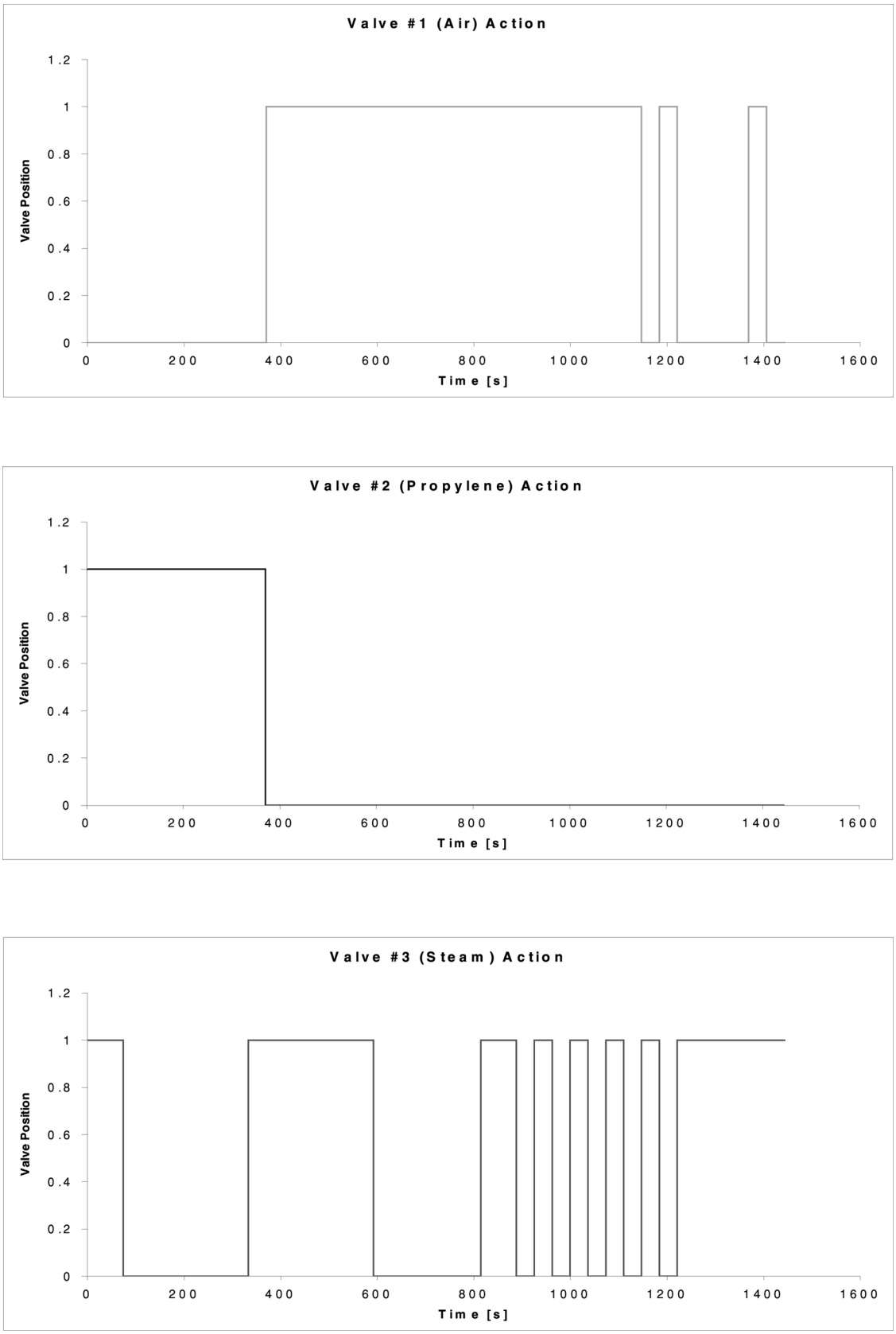

Figure 34 The Optimal Valve Action Time Profiles – showing the valve operations to give the time-optimal solution

degree polynomial filters have been devised, using various statistical measures of fit (i.e. the F test) as feedback in the determination of the polynomial degree (Barak, 1995) and smoothing factor. An additive safety margin to account for model inaccuracies could easily be incorporated into the final spline representation.

5 Conclusions

Most OPS systems are built on the assumption that operating procedures can be generated automatically. However, real-life problems have unexpected situations such as the unavailability of raw material when the system will fail as re-planning takes place to generate a plan that is no longer applicable. In addition, as in any kind of synthesis problem, there are assumptions that can be adjusted during the planning process. All this seems to indicate that research should be oriented towards the development of interactive OPS systems rather than fully automatic.

Most attempts to solve OPS problems have relied on simplified process behaviour models. The simulation-based planning approach makes use of detailed dynamic behaviour models of the process, and a mathematical representation of quantitative safety constraints embedded within a rigorous dynamic optimisation framework.

Most OPS approaches limit their results to an existing plant structure, while there is actually a trade-off between plant design and operating procedures synthesis.

Up to now, engineers and plant personnel have relied on textual descriptions of the operating procedures to operate the plant. The result has been that those documents contain ambiguous terms that are often neglected by the technical staff. Moreover, it is not uncommon to see that these documents become obsolete after modifications to the plant. Based on this, it is no wonder that plant personnel, suspicious about the accuracy of the procedures, end up relying on their experience to operate the plant. The success of coming generations of OPS systems relies on providing the plant personnel with enough information to understand not only when an action is executed but why and how.

References

Aelion, V and Powers, GJ, 1991 "A unified strategy for the retrofit synthesis of flowsheet structures for attaining or improving operating procedures" *Computers and Chemical Engineering* **15**(5) 349–360.

Aelion, V and Powers GJ, 1992, "Evaluation of operating procedures based on stationary-state stability" *Industrial and Engineering Chemistry Research* **11** 2532–2538.

Aelion, V and Powers GJ, 1993 "Risk reduction of operating procedures and process flowsheets" *Industrial and Engineering Chemistry Research* **32**(1) 82–90.

Alsop, N and Macchietto, S, 1995, "Scheduling and sequencing operations for the start-up of an evaporator plant" *IChemE Research Event*, Edinburgh, 125–127.

ANSI/ISA, 1995, "S88 01–1995 – Batch control – Part 1: Models and terminology".

Asprey, SP, Batres R, Fuchino T and Naka, Y, 1999, "Optimal simulation-based operations planning in the presence of quantitative safety constraints" *Industrial and Engineering Chemistry Research* **38**(6) 2364–2374.

Aylett, R, Petley, G, Chung, PW, Soutter, JK and Rushton, AG, 1997, "Planning and chemical plant operating procedure synthesis: a case study" in S Steel and R Alami (eds) *Recent Advances in AI Planning* Springer.

Aylett, RS, Petley, GJ, Chung, PWH, Chen, B, Edwards, DW, 2000, "AI Planning – Solutions for Real-World Problems" *Knowledge-Based Systems* **13**(2–3) 61–69.

Bansal, V, 2000, "Analysis, design and control optimisation of process systems under uncertainty", Ph.D. thesis, University of London.

Barak, P, 1995, "Smoothing and Differentiation by an Adaptive-Degree Polynomial Filter" *Analytical Chemistry* **67**(17) 2758–2762.

Benson, R and Perkins, J, 1997, "The future of process control – a UK perspective" in JC Kantor, CE Garcia and B Carnahan (eds) *AIChE Symposium Series* 316.

CCPS (Center for Chemical Process Safety), 1995, "Guidelines for safe process operations and maintenance" American Institute of Chemical Engineers.

Chapman, D, 1987, "Planning for conjunctive goals" *Artificial Intelligence* **32**(3) 333–377.

Chung, PWH, 1993, "Qualitative analysis of process plant behaviour" *Proceedings of the Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* 277–283.

Crooks, CA and Macchietto, S, 1992, "A combined MILP and logic-based approach to synthesis of operating procedures for batch plants" *Chemical Engineering Communications* **114** 117–144.

Dierckx, P, 1982, "A fast algorithm for smoothing data on a rectangular grid while using spline functions" *SIAM Journal on Numerical Analysis* **19**(6) 1286–1304.

Ernst, GW and Newell, A, 1969, *GPS: A Case Study in Generality and Problem Solving* Academic Press.

Fikes, R and Nilsson, NJ, 1971, "STRIPS: a New Approach to the Application of Theorem Proving to Problem Solving" Artificial Intelligence, **2** 189–208. Also in [GF Luger F (ed.), 1995, *Computation and Intelligence Collected Readings* AAAI Press/The MIT Press.

Foulkes, NR, Walton MJ, Andow PK and Galluzzo M, 1988, "Computer-aidedd synthesis of complex pump and valve operations" *Computers and Chemical Engineering* **12**(9/10) 1035–1044.

Fusillo, RH and Powers, GJ, 1987, "A synthesis method for chemical plant operating procedures" *Computers and Chemical Engineering* **11**(4) 369–382.

Fusillo, RH and Powers, GJ, 1988a, "Computer-aided planning of purge operations" *AIChE Journal*, **34**(4) 558–566.

Fusillo, RH and Powers, GJ, 1988b, "Operating procedure synthesis using local models and distributed goals" *Computers and Chemical Engineering* **12**(9/10) 1023–1034.

Galán, S and Barton, PI, 1997, "Dynamic optimization formulations for operating procedure synthesis" Paper presented at the Annual Meeting of the American Institute of Chemical Engineers.

Hwang, KS, Tomita, S and O'Shima, E, 1991, "Development of a computer based system for synthesizing the operating procedure of a chemical plant" *International Chemical Engineering*, **31**(1) 134–144.

Ivanov, VA, Kafarov, VV, Perov VL and Reznichenko, AA, 1980a, "Design principles for chemical production startup algorithms" *Automatic Remote Control* **41** 1023–1032.

Ivanov, VA, Kafarov VV, Perov VL and Reznichenko, AA, 1980b, "On algorithmization of the start-up of chemical productions" *Engineering Cybernetics* **18** 104–110.

Karnaugh, M, 1953, "The map method for synthesis of combinational logic circuits" *Transactions of the American Institute of Electrical Engineers* **72**(9) 593–599.

KHK (Koatsu Gas Hoan Kyokai), 1991, "Kombinato Jiko Jireishu" in *Japan: Compendium on Accidents and Their Statistics* The High Pressure Safety Institute.

Kinoshita, A, Umeda, T and O'Shima, E, 1982, "An approach for determination of operational procedure of chemical plants" *Proceedings of the International Symposium on Process Systems Engineering (PSE '82)* pp. 114–120.

Korf, RE, 1998, "Artificial intelligence search algorithms" in MJ Atallah (ed.) *The CRC Handbook of Algorithms and Theory of Computation* CRC Press.

Lakshmanan, R and Stephanopoulos, G, 1988a, "Synthesis of operating procedures for complete chemical plants – I. Hierarchical, structured modeling for nonlinear planning" *Computers and Chemical Engineering* **12**(9/10) 985–1002.

Lakshmanan, R and Stephanopoulos, G, 1988b, "Synthesis of operating procedures for complete chemical plants – II A nonlinear planning methodology" *Computers and Chemical Engineering* **12**(9/10) 1003–1021.

Lakshmanan, R and Stephanopoulos, G, 1990, "Synthesis of operating procedures for complete chemical plants – III Planning in the presence of qualitative mixing constraints" *Computers and Chemical Engineering* **14**(3) 301–317.

Lee, JJ, Norris, WD, II and Fishwick, PA, 1993, An object-oriented multimodel design for integrating simulation and planning tasks" *Journal of Systems Engineering* **3** 220–235.

Li, HS, Lu, ML and Naka, Y, 1997, "A two-tier methodology for synthesis of operating procedures" *Computers and Chemical Engineering* **21S** S899–S903.

McAllester, D, 1994, "Graph search and STRIPS planning" *Lecture Notes on Artificial Intelligence* available online at `http://www research att com/ dmac/notes/`.

McDermott, D, 1996, "A Heuristic Estimator for Means Ends Analysis in Planning" *Proceedings of the 3rd International Conference on Artificiaul Intelligence Planning Systems (AIPS-96)* 142–149.

Naka, Y, 1989, "Operational design" Process System Engineering Laboratory, Internal Report, Tokyo Institute of Technology.

Naka Y, Lu, ML and Takiyama H, 1997. "Operational design for start-up of chemical processes" *Computers and Chemical Engineering* **21**(9) 997–1007.

OSHA (Occupational Safety and Health Association, 1992, "Process safety management of highly hazardous chemicals" Regulation No. 29 CFR 1910 119 published by the Department of Labour, Occupational Safety and Health Administration, Washington, DC, February 24, Federal Register **57** (36) 6356–6417.

O'Shima, E, 1978, "Safety supervision of valve operations" *Journal of Chemical Engineering of Japan* **11**(5) 390–395.

Peck, AR, 1968, "An analytic approach to the start-up and shut-down of processes" Ph.D. thesis, Polytechnic Institut of Brooklyn, New York.

Penberthy, JS and Weld, DS, 1992, "UCPOP: A Sound, Complete, Partial Order Planner for ADL" *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning* 103–114.

Peot, M and Smith DE, 1992, "Conditional Nonlinear Planning" *Proceedings of the First International Conference on AI Planning Systems* 189–197.

Rivas, RJ, Rudd DF and Kelly, LR, 1974, "Computer-aided safety interlock systems" *American Institute of Chemical Engineers Journal* **20**(2) 311–319.

Rivas, JR and Rudd, DF, 1975, "Man-Machine Synthesis of Disaster-Resistant Operations" *Operations Research* **21**(1) 2–21.

Strimaitis, PJ, 1987, "Application of artificial intelligence techniques to the strategic control of chemical processes (the PROLOG synthesizer)" Honours project supervised by Roeach J R, Department of Chemical Engineering, University of Adelaide, Australia.

Roach, JR, O'Neill, BK and Strimaitis, PJ, 1990, "The use of artificial intelligence techniques for the generation of operating command sequences for the regulation of complex chemical processes" *Proceedings of the Eighteenth Australian Chemical Engineering Conference* 453–460.

Soutter, JK and Chung, PW, 1995 "Planning architectures for operating procedure synthesis" *Proceedings of the Eighth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* 221–229.

Soutter, J and Chung, PW, 1996, "Partial order planning with goals of prevention" *Proceedings of the Fifteenth Workshop of the UK Planning & Scheduling Special Interest Group, 2* 300–311.

Tomita, S, Hwang, KS and O'Shima, E, 1989, "On the development of an AI-based system for synthesizing plant operating procedure" in *Computer Integrated Process Engineering (CIPE '89) Symposium Series* (**114**) 303–310.

Waldinger, R, 1977, "Achieving several goals simultaneously" in EW Elcock and D Michie (eds) *Machine Intelligence 8* Ellis Horwood Publishers.

Weld, DS, 1994, "An introduction to least commitment planning" *AI Magazine* **15**(4) 27–61.

Wilkins, DE and Desimone, RV, 1994, "Applying an AI planner to military operations planning" in M Fox and M Zweben (eds) *Intelligent Scheduling*Morgan Kaufmann Publishers Inc.