•

AI Planning in a Chemical Plant Domain

R.S.Aylett[1], J.Soutter[2], G.Petley[1], P.W.H. Chung[3], A.Rushton[3]

Abstract This paper discusses the issues involved in applying a generic AI planner to the problem of generating plant operating procedures for chemical process plant. It considers the problem of providing the correct planning facilities, concentrating on dealing with flow of chemicals, and proposes the use and integration of a special purpose planner for valve sequencing. It goes on to consider the knowledge engineering issues using a sample chemical plant to illustrate the solutions adopted.

1 INTRODUCTION

AI Planning has been actively researched since the 1970s and STRIPS [8], yet, unlike other knowledge-based systems, has a small number of commercial successes and is still not thought of outside the AI Planning Community as a technology ripe for real-world application. In this paper we discuss the application of hierarchical non-linear planning to a real-world domain not previously considered by the AI Planning Community (though tackled by several groups in the Chemical Engineering Community [1, 6, 9, 10, 11, 13, 14]). This domain is the generation of operating procedures for continuous process plant in the chemical industry.

We argue that two areas must be tackled in order to apply a planner to a real-world domain, one related to specificity and one to generality. Firstly, there must be a match between the facilities offered by the planner and the key characteristics of the domain - the planner must be adequately specific. Failing this it will either be impossible or take far too long to solve problems within the domain.

Secondly, the knowledge engineering issues must be dealt with so that it is possible to reapply the system to a number of distinct problems within the domain - the planner must be adequately generic. Failing this, the effort required may outweigh the benefits of applying it [4].

We briefly describe the nature of plant operating procedures. We then characterise the domain by looking at the key planning problems within it. We discuss how CEP - the Chemical Engineering Planner - deals with these problems, concentrating on issues relating to flow. Finally we look at the application of CEP to a particular plant - the Double Effect Evaporator (DEE) - and discuss the knowledge engineering lessons learned.

2 PLANT OPERATING PROCEDURES

All industrial plants require an extensive set of operating procedures which define the steps required - for example - to start the plant up, to shut the plant down, to isolate pieces of equipment for maintenance or to deal with emergency situations. Steps may be carried out manually by human operators, or some of them may be embodied in the plant control system, depending on the level of automation. It is clearly vital for reasons both of safety and efficiency that procedures are of a high quality.

In the chemical process industry, a multi-disciplinary commissioning team is normally responsible for defining sets of procedures, taking of the order of two man-years of effort. If operability problems are uncovered during this work, late changes to the design of the plant may result, sometimes while the plant is actually being constructed. These are the motivations for the development of computer-based tools to aid in the authoring of operating procedures.

There is an obvious match between a sequence of operating procedure steps and the output of an AI planner. In the INT-OP (INTegrating OPerability) project we apply state-of-the-art AI hierarchical non-linear planning techniques to the generation of such procedures. To do this successfully, it has been necessary to think carefully about the characteristics of the process plant domain.

3 CHARACTERISING THE DOMAIN

Little work has been carried out in characterising planning domains, especially compared to other task types such as diagnosis. However in [2], we make a start by characterising planning domains in terms of TASK, AGENT and ENVIRONMENT together with the relationships between them, as shown in Figure 1.

Of the significant attributes and relationships discussed in [2], this domain has four. The first is that a TASK cannot be entirely characterised by the desired end-state; intermediate states are also important. In a process plant it is not only important that the plant arrives at, say, its start-up state, it is equally important that it does not pass through any unsafe states along the way. For example, a 'successful' start-up which vented poisonous gases to the atmosphere or contaminated an expensive catalyst with steam would be unacceptable. A planner must allow unsafe states to be excluded from valid plans.

A second domain characteristic is the highly interconnected character of its ENVIRONMENT. In a robot blocks world, removing one block normally has no effect on the other blocks in the domain (as long as blocks are only taken from the top of piles). In a process plant, the significant effect of opening or shutting a valve is not that the state of the valve changes, but that, depending on the state of the rest of the plant at the time, one or more chemical flows may be started or stopped. The interconnectedness of the domain is reflected in the particular

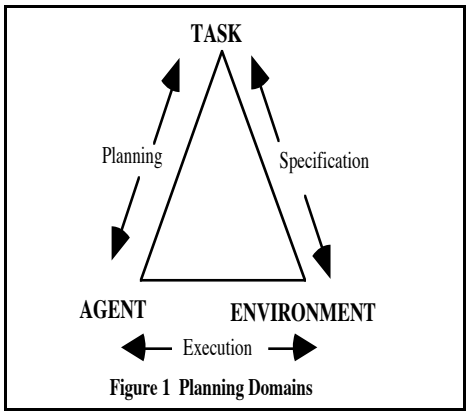

Figure 1 Planning Domains

1. IT Institute, University of Salford
2. BG Technology Centre, Loughborough
3. Chem. Eng. Dept, Loughborough University

properties of flow.

Flow plays as fundamental a role in a chemical plant domain as movement does in a robotic domain. Just as robot letter delivery can be decomposed into a sequence of MOVE operations, so could the Haber Process for making ammonia be decomposed into FLOW of chemicals. Does this mean that FLOW can be handled in much the same way as robot route planning?

In fact FLOW differs from robot route planning in many significant ways. Firstly, while a MOVE action can be thought of as directly moving the robot along a portion of its route, FLOW is produced as a side-effect of valve (and pump) operations. Secondly, while a robot moves only along the route planned for it, flow will occur at all junctions off the chosen route not specifically blocked. Thirdly, a flow will continue once it is started until it is explicitly stopped. Fourthly, more than one chemical may pass down the same route simultaneously - for example in the Haber process hydrogen and nitrogen are merged into one flow. Finally, a flow route is contaminated with the flowing chemical even after the flow has stopped. These differences are expressions of interconnectedness and must be handled if planning is to be successful.

A further domain characteristic is that the time to complete a TASK and the time taken by each action given to the AGENT for execution is very different. Starting up a plant takes hours - sometimes more than a day. An individual valve operation may take seconds. This difference in granularity indicates that hierarchical planning rather than goal reduction is needed.

A final characteristic indicating a need for hierarchical planning is the size of ENVIRONMENT. A real-world process plant contains hundreds to thousands of components, and without an abstraction hierarchy planner performance is likely to be unacceptably low. We note that current manual generation of plant operating procedures also makes substantial use of decomposition hierarchies.

4 THE CHEMICAL ENGINEERING PLANNER

The Chemical Engineering Planner, CEP, has been developed over the last five years as a tool for operating procedure synthesis (OPS) [16]. It divides the tasks involved in OPS into three areas: planning using operators, the handling of safety considerations and valve sequencing.

4.1 Operators

CEP provides three types of operators: expandable operators; primitive operators; and macro operators. In an expandable operator, like this operator to start up a vacuum pump:

```
operator StartVacuumPump
{    vacpump ?vac;
     inlet ?source;
     gas ?gas;
     chemical ?chem;
pair supplyChem, ?source : ?gas;
     expand
          active(?vac);
     using
          flowToMix(?source, out, ?vac, in, ?chem, ?gas, fill);
end}
```

quantities beginning with `?` are variables to be instantiated during planning, and act like a form of inference rule, allowing the `expand` goal to be rewritten in the form of the `using` goal(s).

Primitives, like this operator to operate a hand valve:

```
operator  OperateHandValve
{    aperture ?state1;
     aperture ?state2'
     hand ?h;
?state1 != ?state2
achieve
     aperture of ?h is ?state2
using
     aperture of ?h is ?state1;
end;
print(?n) [name of ?state2 is ?n] ' valve ' ?h;}
```

are essentially STRIPS operators, in which a print statement produces an actual step in the final operating procedures.

Macro operators [15] were introduced into CEP for two main reasons. Firstly, they allow the user to contribute knowledge about the order in which things should be done where this is known in advance. Secondly, they allow the representation of facts which should be protected over time in a way which STRIPS operators cannot. For example, a process might require a reaction vessel to be cooled through several steps involving the adding of a number of chemicals. A macro operator guarantees that cooling cannot be interrupted as the other steps occur. Macro operators also proved vital in our solution to the problems posed by flow, discussed in Section 5, where more detail of their use is given.

4.2 Safety

We summarise CEP's handling of safety here - more detail can be found in [17]. CEP's approach is based on 'goals of prevention', similar in concept to the 'don't disturb' goals of [19] and to a number of other like suggestions in the literature. However only two discussions of planning with this concept have been found in [19] and [10]. A simple method of handling safety would be to qualify planner operators with the goals they must not achieve. However CEP does not modify operators, rather stating goals which are not to be violated during planning separately as domain knowledge. For example:

```
restrictions
{    prevent
       state of HeatExchanger1 is started;
       state of GlassCooler1 is stopped;
     end}
```

prevents energy entering the DEE plant - because a heater is on - before there is any way of it leaving the plant through the cooler.

From a knowledge-engineering perspective, one should keep all safety-related knowledge together, rather than scattering it between actions. Furthermore, as discussed below, incorporating restrictions into actions makes it hard to define generic planner operators since the safety issues relating to a physical component are likely to depend on the plant into which it is incorporated.

When an action is added to the plan, CEP examines it (an approach first suggested in [14]) to see if it violates a goal of prevention and if so either adds new preconditions or adds constraints to the action variables. Thus goals of prevention are monitored, and modify actions only when a violation occurs.

5 DEALING WITH FLOW

As argued above, flow - achieved by sequencing open and close operations on valves - is fundamental to continuous process plant domains. Previous OPS work in the Chemical Engineering Community either dealt with valve sequencing

alone [11, 13], or dealt with the planning of the operation of reaction vessels to the exclusion of valve sequencing [10]. CEP is the first OPS system to combine both.

Several approaches are possible. Firstly, flow may be handled as a post-processing step to the allocation of reaction vessels. Secondly, flow may be dealt with like other aspects of planning through planner operators. Thirdly, flow may be handled via a sub-planner integrated into the planner. CEP has adopted this third solution.

5.1 Flow as post-processing

The first approach was adopted in [6] who viewed OPS as a resource allocation rather than a planning problem. Their objective was the production of given quantities of given products, achieved by choosing a process route (sequence of reactions) to produce each of the required products and then allocating resources in the form of reaction vessels to each. In a second phase, steps were added to the procedure to create the necessary flow paths between the reaction vessels used.

The problem with this approach is that the pipes which carry a flow are themselves resources. If flow paths are chosen in isolation then two flow paths may end up sharing the same pipes mixing of chemicals in dangerous or undesirable ways. These flows need not overlap temporally: a flow can contaminate a pipe with a chemical which may have to be removed before a second flow can be created with another chemical. Removal of a chemical often involves washing a pipe out with some neutral substance like water and in order to get this to the required location in the plant, sometimes the substance will have to flow through a vessel. This causes problems because the system is designed to allocate vessels to tasks only in the first phase of procedure creation. This indicates that flow cannot be treated as a post-processing step.

5.2 Flow using operators

We have indicated above why flow is hard to deal with using the operator approach. A `MOVE` operator with pre-conditions `at(?Robot,?X),next-to(?X,?Y)` and post-condition `at(?Robot,?Y)` can be used to find a route between two distant locations. This will not do for flow since not only must valves along the chosen flow-route be opened, valves off the flow-route must also be closed to stop flow into other parts of the plant. With no explicit representation of the flow-route, it is hard to close the correct valves, though the use of axioms, as in UCPOP [12], has not yet been investigated.

Modelling flow through operators for opening and closing valves is also problematic. Since flow is a side-effect which depends on the configuration of the plant, the STRIPS assumption that all effects of an action are declared in the operator is very hard to meet. For example, in Figure 3, if valves V2 and V4 are shut, then opening valve V3 will not result in a flow to the drain.

There are three alternatives. First, a separate operator can be used to describe the operation of each valve in each interesting plant state. Second, the operator representation can be enhanced to allow the effects of opening a valve to be a function of the state of the plant, for example using conditional effects. Third, operators can be used to represent the opening of each interesting flow route rather than representing each individual valve operation.

Each of these three strategies produces operators that are specific to a particular plant. They simply differ in the tradeoffs they make between complexity and brevity. In the first and third cases each operator is simple but a huge number are required to describe a complex plant. In the second case, each operator is very complex but only one is required for each valve in a plant. In all cases, the valve sequencing problem must be solved anew for each new plant.

In conclusion, it does not appear possible to create an operator model of opening a valve that is independent of the specific valve to be opened. Similarly, it does not appear possible to create an operator model to start a chemical flowing through a plant independent of the process plant that the chemical will flow though.

5.3 Flow with a subplanner

Domain-dependent subplanners go all the way back to [8] who pointed out that a robot route-finding algorithm could be represented as a subplanner designed to make `(inroom robot <- X)` true. Indeed handling motion planning like this is even more necessary for manipulators with many degrees of freedom where computational geometry rather than predicate calculus is the appropriate formalism. It is noticeable that all three of the AI Planning Systems currently used for real-world problems - OPLAN, SIPE, and PRODIGY - provide support for subplanners, suggesting that domain-dependent algorithms of this type are not unusual.

In this section we propose a domain dependent but plant independent mechanism for creating a flow using a specific flow reasoning algorithm. The implication of this work is that some difficult problems are best solved through the development of domain dependent modules that live within otherwise domain independent planners.

Incorporating flow handling via a subplanner requires an algorithm for establishing flow and an interface for incorporating this into the planning process. We consider each in turn.

The algorithm that is used by the OPS community to create a flow of chemical is based on work by [13]. This algorithm finds a flow route using a maze searching algorithm; the valves around this route are closed and then the valves along the route are opened. The advantage is that each flow of chemical has an easily determined effect - the contamination of all the units along the flow route by the

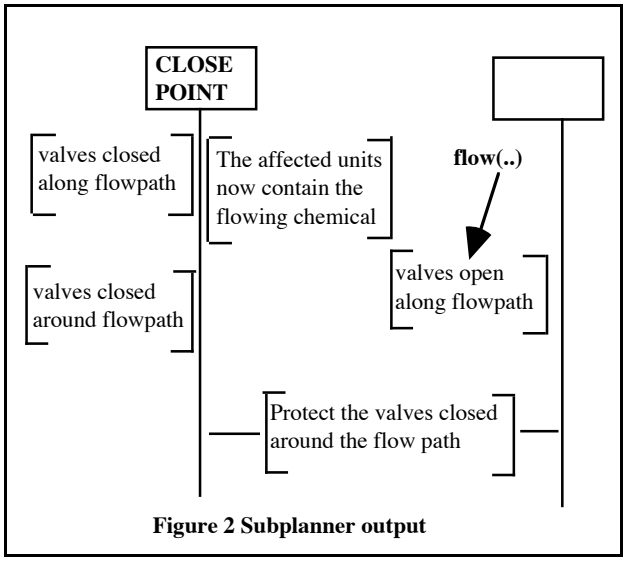

Figure 2 Subplanner output

chemical being transported.

Clearly the subplanner should be called when the planner has a goal to create a flow between two points. It will then find a flow route. At issue then is how the route should be communicated back to the planner.

The first strategy adopted in CEP was to have the subplanner directly manipulate CEP's data structures including CEP's current plan. Thus each time a flow goal was solved, the subplanner output was translated into a partial plan with the structure shown in Figure 2, where the blank box represents the CEP node which originally created the flow goal. A suitably instantiated version of this structure was added to the existing CEP plan.

However this strategy proved limiting. For example, when we decided on a third pass over the flow route so that pumps could be turned on strictly after a flow path was opened it was very difficult to make the change because the translation of the subplanner output into a partial plan was hard coded into the subplanner itself.

Looking at OPLAN, SIPE and PRODIGY, all three approach suplanning as a mechanism for performing mathematical reasoning during planning. Hence it is not suprising that their subplanners are constrained to behave as mathematical functions. Subplanners take a fixed length sequence of arguments as input and produce a result that is completely determined by these input parameters. The input parameters are planning objects, that is variables or constants. A planning variable is used to hold the return value.

These interfaces cannot however be adopted for a flow subplanner which differs in significant ways from a straightforward mathematical function:

- Flow subplanners are non-deterministic. There may be many flow routes for a particular chemical and any suitable one may be chosen arbitrarily. All possible routes should be considered during backtracking to ensure completeness.
- Flow subplanners may implement partial functions. There may be no feasible flow paths between two points or all feasible paths may be blocked by other flows of chemical. Hence it may not be possible to find a route for a particular flow of chemical at a particular point in a plan.
- Unbound input parameters can be important. For example, if the destination point for a flow is unbound but constrained to the set of possible drains for a particular chemical, then the subplanner could oportunistically choose a suitable drain when looking for a flow route.
- Flow subplanners return partial plans, not a single variable. The interface must support their incorporation into planning.

A macro operator, as mentioned in section 4, was used to implement this interface, which is general enough to support other subplanners of like complexity and is therefore more powerful than the interfaces of SIPE, OPLAN and PRODIGY.

```
macro Flow
{
valve *?opened, *?closed;
unit *?contaminated, ?source, ?destination;

call     /* subplanner call */
      flow(?source, ?destination)
        [*?opened, *?closed, *?contaminated];

solve    /* Use this operator for */
      flow(?source, ?destination, ?chem);
nodes    /* Node definitions */
      1 instant close;
order    /* Temporal ordering of nodes */
      1, @;
require  /* preconditions */
      1, @          aperture of *?closed is closed;
      @             aperture of *?opened is open;
      @             aperture of *?pumps is open;
achieve /* effects */
      1 contains(*?unit, *?port, ?chem);
}
```

The `call` section of this macro specifies `flow` as the subplanner to be called from the CEP table relating names to available subplanners. The arguments in round brackets are input parameters: question marks show that they are variables rather than constraints. The arguments in square brackets are return parameters: the star in front of them indicates they represent a set of objects rather than a single object.

The `call` above will find a route between `?source` and `?destination` and then constrain the variables `*?opened`, `*?closed` and `*?contaminated` with the details of the flow route found. For example, `*?opened` will be constrained to the set of valves that are to be opened. The call may have the side effect of binding some of the input parameters - a particular flow must start from a particular source and end up at a particular destination.

The remainer of the macro describes the handling of the variables constrained by the call. For example, at some point in the partial plan being produced by the subplanner, the valves around the flow path must be closed. This "close point" is referred to in the macro as node1. These valves must remain closed until the point in the CEP plan which produced the flow goal in the first place, which we give the special symbol @. CEP can say then that the flow operator has the precondition shown, amounting to "the aperture of all the *?closed valves must be closed at node 1 and remain closed until @". Thus the macro now performs the translation of the subplanner output into a partial plan.

The following steps are required to apply this macro:

1) A new set of variables are created according to the type definitions at the macro head.

2) These variables are constrained so the `solve` section matches the goal to be solved.

3) The flow subplanner is called, further constraining the `opened, closed` and `contaminated` variables.

4) The domain of all starred variables is fixed.

5) New nodes are added as described in `nodes` and their order is constrained as in `order.`

6) Preconditions, effects and causal links are added as described in `achieve` and `require`.

7) On backtracking, steps 6, 5 and 4 are undone and the subplanner is asked for an alternative solution.

6 THE DOUBLE EFFECT EVAPORATOR

After successfully applying CEP to all the 'toy' problems in the literature, with the exception of those requiring numerical calculations, a Double Effect Evaporator (DEE) test rig constructed in the Chemical Engineering department at Loughborough University was tackled [3]. This was to examine scaling-up issues and the knowledge-engineering problems involved in applying CEP to a real-world domain. The target users for CEP are not experts in AI Planning but chemical engineers working at the design stage. If CEP is to be useful, it must be possible for such users to apply it to any chemical process plant - the planner must be sufficiently general.

Figure 3 shows a simplified schematic for part of the plant - the Engineering Line Diagram (ELD) giving a more accurate picture requires a whole sheet of A4 and cannot be shown here. The DEE set-up contains a larger number of components than in most previous domains - for example thirty plus hand valves - while the number of different types of equipment is also large, with valves, controllers, pumps, heaters, coolers, evaporators, feed tank, mixing tank and a barometric condenser.

The DEE functions to remove water from a salt water solution (brine). It is called 'double effect' because the steam evaporated off in the first evaporation supplies the energy for the second evaporation. Because the test rig was used for teaching, the concentrated brine is returned to the starting point and mixed with water, returning the brine solution to its original concentration of salt and allowing the process to continue indefinitely.

Configuring CEP to a particular plant requires: a model of the plant in a suitable form; an appropriate set of planner operators; a set of safety restrictions; a description of the initial and desired states of the plant (the problem description). Knowledge acquisition for planning is little investigated [18] and it has been argued that its costs in time and effort form a major obstacle to the use of AI planning Systems [4]. As part of the DEE study, tools were developed to tackle knowledge acquisition.

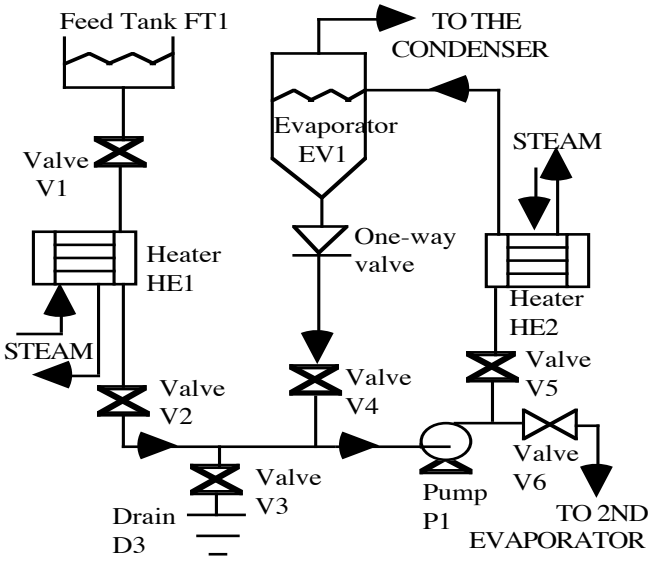

Figure 3 DEE schematic (part)

6.1 Acquiring the Plant Model

CEP requires a textual plant model described in terms of a hierarchy of components and the connections between them. A complete hierarchy for plant components is being developed and was substantially extended by the DEE domain. CEP uses a hierarchical frame-based description developed during earlier work at Loughborough [5].

Manual entry of the instances required for the DEE plant would be non-trivial: for an industrial scale plant, described in dozens (at least) of ELDs, inconceivable. An automatic system was therefore developed for producing the domain description. A popular package for design, AutoCAD, was adapted to provide the standard chemical engineering equipment symbols. When a new piece of equipment is added to the plant diagram, a text box appears prompting the user to add the name and connections for it. Thus on completion of the drawing, the necessary information has been collected to allow the automatic creation of a file that describes the plant to CEP. The plant model is generated automatically for CEP as part of the plant design process with little extra work for the user.

6.2 Planner Operators

The plant model expresses declarative knowledge about configuration and structure. Planner operators capture the behaviour of plant components - procedural knowledge - and if poorly defined can make planning grossly inefficient, incorrect or impossible. Creating a set of correct, consistent and efficient operators for a particular domain is taxing for experts in AI Planning and could not feasibly be left to CEP's target users.

Two alternatives exist: to provide a tool that allows non-expert users to define planner operators from scratch without the current pain, or to provide a library of planner operators that can be reused - or at worst slightly adapted - for a particular plant. The second alternative is currently being pursued. For it to succeed, CEP planner operators must be generic to the continuous process plant domain.

This is a plausible target: though each plant is a unique configuration, the same components are used in many different plants. One might hope to define generic operators for each piece of equipment at the leaves of the equipment hierarchy already mentioned.

In the DEE study we discovered that deficiencies in planner functionality can often be overcome by creating problem-specific operators. Our ability to make the operator set generic acted as a measure of the adequacy of CEP for the domain. An initial solution, reported in [3], used just under 20 operators to produce a successful start-up procedure containing 50 steps in under a second on a Sparc 5. However a number of these operators contained information specific to the DEE plant which could not be transferred to any other plant. After improvements in CEP, in particular the introduction of macro-operators and their use to model flow, a solution was developed with 13 completely generic operators: 6 ordinary ones and 7 macros.

Demonstrating that the DEE domain could be solved with generic operators suggests that the idea of providing an operator library for users is feasible. Work is now going on to structure such a library round the equipment hierarchy and to provide an interface at plant design time for operator selection.

6.3 Safety Restrictions

It is possible to misuse safety restrictions to force an ordering onto planner operators for a particualar task - such as start-up say - within a particular plant. An example of this was discussed in [3] for the DEE plant. The introduction of macro operators into CEP reduced the DEE restrictions from 13 to 5 - all independent of the overall task being planned - as ordering information was put where it belonged in particular macros. An example of such a general restriction is that preventing a vacuum in the condensor by stating that the pump must be off if the evaporator is off. Once the evaporator is on, steam flows into the condensor giving the pump something to work on. We note however that this is a plant-specific instance of a more general safety consideration, that pumps should never be switched on if there is nothing to pump. There may be scope for a hierarchy of safety restrictions, which would allow the user to select those relevant to the components and process in their particular plant, and this will be investigated further.

6.4 Problem Description

Currently, the initial and final states of the plant are being entered by hand. Though less trying than a whole plant model, this is still undesirable. It is clearly possible capture the initial and final states of the plant from the plant model and this facility is being integrated into the system.

7 DISCUSSION

CEP has been used to produce operating procedures using AI planning for a domain more complex than any other previously attempted. Previous work in the Chemical Engineering community using state-graphs limited work to plants containing a handful of valves because of the number of states they generated: 20 valves each with 2 states produces 1,048,576 nodes in a state-graph. Other workers used larger plant [14] but only considered valves and not vessels. A real-world nuclear fuel processing plant was used in [6], but this work concentrated on optimising a hand-generated plan. Only [1] have seriously considered AI Planning technology but in this case a linear STRIPS engine only. Work has been carried out examining the links between planning and qualitative reasoning [7], but using execution monitoring as a way of dealing with problems rather than handling them within the planning process itself. CEP appears to be the only system to date which can produce procedures for large plant dealing with both vessels and flow.

Further work is however needed in a number of areas. For example, not all linearizations of a partial plan are equally acceptable as an operating procedure. It may be the case that some actions which are ordered in the plan really should be carried out one after another, without interpolation of other actions partially ordered with respect to them. For example, when brine enters a glass preheater, the steam passing through it starts to condense and requires the shutting of one valve and the opening of another at virtually the same time to change the flow to the trap drain. One would not wish other valve operations elsewhere in the plant along a different branch of the plan graph to be inserted into this sequence even though this linerarization is formally possible. Currently, the plan graph representation is not sufficiently rich to represent this information.

Secondly, CEP knows only the plant topology, which means that some knowledge available to human authors of operating procedures is being neglected. The DEE plant contains a glass pre-heater. Now it is possible to start up the plant without using this preheater, and accordingly CEP originally generated a procedure that did not use it. The reasons for using the preheater during start-up are: the temperature of the brine can be increased in stages - protecting the glass lined vessels, and the control of the temperature of the brine entering the first evaporator is easier with two heaters. An expert in operability, seeing that the design contained a glass preheater, would infer that it was there for the purpose of start-up and accordingly use it. This knowledge of design rationale does not appear to be representable within the confines of planner operators and we are currently examining the issue in more detail.

A further example of missing knowledge is the spatial organisation of the plant. For example, if two valves have to be opened manually, then these actions should be together if they are geographically next to each other in the plant. This information is not available from ELDs or the CEP plant model.

Finally, much work still remains to be carried out in making CEP usable by a non AI expert. In particular, we need to demonstrate in practice that planner operators can be selected from a library for a new plant and result in CEP producing correct operating procedures.

8 CONCLUSIONS

We argued in the introduction that AI Planners could only be successfully applied in the real world if they were sufficiently specific - that is, had the necessary functionality for the required domain - and were also sufficiently generic - that is, could easily be reapplied to new problems in the domain. We have discussed how these two areas have been tackled so far in our work and shown that real progress can be made in both cases.

REFERENCES

[1] Aelion, V. & Powers, G.J. (1991) A Unified Strategy for the Retrofit Synthesis of Flowsheet Structures for Attaining or Improving Operating Procedures. In: Computers and Chemical Engineering, vol. 15 no 5, pp349-360, Pergamon 1991

[2] Aylett, R.S. & Jones, S.D. (1996) Planner and Domain: Domain Configuration for a Task Planner. International Journal of Expert Systems v9 no2 pp279-318, JAI Press 1996

[3] Aylett, R.S; Petley, G.J; Chung, P.W.H; Soutter, J. & Rushton, A. (1997) 'Planning and chemical plant operating procedure synthesis: a case study'. Proceedings, 4th European Conference on Planning, Toulouse, 1997. Springer-Verlag Lecture Notes in Artificial Intelligence, to appear.

[4] Chien, S.A; Hill, R.W; Wang, X; Estlin, T; Fayyad, K.V. & Mortenson, H.B.(1996) Why Real-world Planning is Difficult: a Tale of Two Applications. In: New Directions in AI Planning, M.Ghallab & A.Milani, eds, IOS Press, Washington DC 1996 pp 287-98

[5] Chung, P.W.H. (1993) Qualitative Analysis of Process Plant Behaviour. Proceedings,Industrial and Engineering Applications of AI and Expert Systems,ed. P.W.H.Chung, G.Lovegraove & M.Ali, pp277-83 Gordon & Breach 1993

[6] Crooks, C.A. & Macchietto, S. (1992) A Combined MILP and Logic-Based Approach to the Synthesis of Operating Procedures for Batch Plants. Chemical Engineering Communications 114, pp117-144

[7] Drabble, B. (1993) Excalibur: a program for planning and reasoning with processes. Artificial Intelligence, v62 no1, pp1-40, Elsevier 1993

[8] Fikes, R.E; Hart, P.E. & Nilsson, N. (1972) "Learning and Executing Gneralised Robot Plans'. Artificial Intelligence, 3:251-288, 1872

[9] Foulkes, N.R.; Walton, M.J.; Andow, P.K. & Galluzo, M. (1988) Computer Aided Synthesis of Complex Pump and Valve Operations. Computers and Chemical Engineering, 12 pp1035-1044

[10] Fusillo, R.H. & Powers, G.J. (1987) A Synthesis Method for Chemical Plant Operating Procedures. In: Computers in Chemical Engineering, vol 11 no 4, pp 369-382, Pergamen, 1987

[11] Lakshmanan, R. & Stephanopolous, G. (1990) Synthesis of Operating Procedures for Complete Chemical Plants - 3. Planning in the Presence of Qualitative Mixing Constraints In: Computers in Chemical Engineering, vol 14 no 3, pp301-317, Pergamon 1990

[12] Penberthy, J.S. & Weld, D.S. (19920 'UCPOP: A Sound, Complete, Partial-order Planner for ADL'. Proceedings, KR-92, pp 324-32

[13] O'Shima, E. (1978) Safety Supervision of Valve Operations. Journal of Chemical Engineering of Japan. v5 pp390-5, 1978.

[14] Rivas, J.R. & Rudd, D.F. (1974) Synthesis of Failure-Safe Operations. In: AIChE Journal, vol 20 no 2, pp 320-325, March 1974.

[15] Sacerdoti, E. (1985) The Non-Linear Nature of Plans. Proceedings, IJCAI, pp206-14 1985

[16] Soutter, J. (1997) An Integrated Architecture for Operating Procedure Synthesis. PhD thesis, Loughborough University, Loughborough LE11 3TU, UK

[17] Soutter, J. & Chung, P.W.H. (1996) Partial Order Planning with Goals of Prevention. proceedings, 15th Workshop of the UK Planning and Scheduling SIG, vol 2 pp300-11, John Moores University, Liverpool, UK.

[18] Valente, A. (1995) Knowledge-level Analysis of Planning Systems. ACM SIGART Bulletin Special Issue, 6(1), Jan 1995

[19] Weld, D. & Etzioni, O. (1994) The first law of robotics (a call to arms). Proceedings, 12th National Conference on Artificial Intelligence, AAAI 94. pp1042-47.