Automatic Synthesis of Batch Plant Procedures: Process-Oriented Approach

G. E. Rotstein, R. Lavie, and D. R. Lewin
Dept. of Chemical Engineering, Technion, Haifa 32000, Israel

A strategy for the automatic synthesis of plant operating procedures, through the integration of artificial intelligence and mathematical modeling tools is discussed. This "predicative/numerical" modeling paradigm integrates qualitative and quantitative aspects of the problem in a same framework. It uses a "process-centered" qualitative modeling approach together with MILP optimization techniques to automatically generate operating strategies. The operation procedures network (OP-Net), formulated in terms of processes from a generic library, describes abstract (hardware-independent) processing paths leading to the desired operational goals. In this approach, the model assumptions, conservation expressions and data are described at the same level of abstraction, using the same formal representation. This permits the direct, automatic generation and modification of operation procedures in situations involving changes in basic assumptions and initial scenarios. Thus, flexibility, an essential feature of batch processing, is safeguarded.

Introduction

The chemical process industry is increasingly interested in relatively small-scale, high-value, short-life-cycle products. This shift in industry perspective, justifies a fresh look at the way processing equipment is designed, programmed, and operated in order to ensure its flexibility. Flexibility of operation allows for the manufacture of different products using the same basic equipment. It also prolongs the useful life of a production facility beyond the short life cycle of a particular product. Exploiting the inherent flexibility of such plants gives rise to difficult operation problems. Although programmable logic controllers provide effective automation and sequencing of tasks, they are "hardwired" in the sense that they must be reprogrammed for each new production cycle. This raises the cost of flexibility and limits their effective capacity to adjust to plant upsets. In addition, as the size of the plant increases, the programming of these controllers become more involved. The automatic generation of batch plant procedures has therefore emerged as an important issue.

The analysis and synthesis of process operations require models of the considered problem. The models developed for the operation of flexible production facilities should be flexible themselves (that is, they should be easily modified in order to mirror changes in the plant or manufacturing conditions). Traditional modeling techniques are not explicit enough to meet this requirement, while those based on artificial intelligence show promise (Stephanopoulos et al., 1990; Piela et al., 1991). The need for explicit methods becomes apparent when the size of mathematical models grows and makes their development and understanding difficult. This is the case when we must deal with models consisting of thousands of variables and constraints.

Recently, there has been interest in the integration of artificial intelligence (AI) and operations research (OR) techniques for the solution of MI(N)LP [mixed integer (non)linear programming] models. OR methods are useful for the efficient solution and analysis of numerical optimization problems. However, AI methods may provide two main avenues of potential benefit: (1) they provide modeling methodologies to simplify the generation, correction, and reuse of the process OR models; (2) they may reduce the computational load by systematically exploiting the problems logical structure and the accumulated experience. Previous works were based on propositional logic and concentrated on aspects related with the second avenue. Two examples are the generation of propositional logic models from the design superstructures considered, or the use of the logic relationships to aid the branch and bound search in the solution of MI(N)LP problems (Raman and Grossmann, 1991, 1992). However, propositional logic is not a satisfactory language for the development of explicit process models. In this work, we shall discuss its limitations

Correspondence concerning this article should be addressed to R. Lavie.

and the advantages that are gained when predicative logic is used. The formal framework proposed aims to facilitate the integration of tasks from the generation of the model to the systematic accumulation of experience. The approach is implemented for the synthesis of operation procedures.

The automatic generation of operation procedures was first considered by Rivas, Rudd, and Kelly (1974). AI tools were first employed by Fusillo and Powers (1987), who introduced the means-ends planning paradigm, the use of functional operators, and the concept of "stationary states." A framework for the automatic synthesis of operating procedures was discussed by Stephanopoulos and Lakshmanan (1988a, b), based on a general object-oriented modeling structure and a nonlinear planning approach. Recently, a methodology for the evaluation of operating strategies, relying on the stability of the "stationary states" has been also proposed (Aelion and Powers, 1992). Crooks and Macchietto (1992) were the first to consider the specific problems associated with the synthesis of batch plant procedures. Their approach combines logic and the so-called state task network (STN) representation for scheduling (Kondilis et al., 1993). MILP is successfully employed as a subgoaling strategy (that is, when and where each task must be activated), while the synthesis of detailed control sequences for each subgoal relies on specific rules and logic inference. The hierarchical strategy adopted appears to be very effective. However, this approach relies on an arbitrary assignment of tasks to processing units (that is, unit suitability). This is detrimental to model maintenance and reusability.

In this article, a formal framework for the integration of artificial intelligence and mathematical modeling techniques is presented, and we present the theoretical background. Reasoning methods and their relation to optimization techniques are described. The limitations of propositional logic as a modeling language are then discussed and qualitative process theory (QPT) (Forbus, 1984) is proposed as an adequate representation technique. Here, QPT is used as a representation language and not as a tool for qualitative simulation as originally suggested by Forbus. Then, the integration of different tasks such as diagnosis (Grantham and Ungar, 1990), qualitative simulation (Crawford et al., 1990), numerical simulation (Forbus and Falkenhainer, 1990), and operation planning in a same framework using a common database is possible by applying different aspects of the QPT structure for the various objectives.

The approach is implemented on the automatic synthesis of batch plant operating procedures. We present an overview of the approach, and the desired properties for process operation models are discussed. A new "predicative/numerical" modeling paradigm is introduced with the potential of fulfilling operation synthesis needs. The proposed extended QPT library is used to model batch recipes, the symbolic/parametric reasoning strategy is discussed, and the MILP formulation employed for the modeling of batch operation problems is summarized. Finally, the software implementation and a case study are described.

Background

Parametric and symbolic reasoning

Reasoning is the process by which conclusions are obtained from statements about a problem under consideration. These statements describing the system being considered constitute the *model* of the problem. Two levels of reasoning can be distinguished according to the domain of the variables involved (Post and Sage, 1990a): *parametric* and *symbolic*.

Parametric Reasoning. In parametric models, the domain of variables consist of numbers or truth values. The variables represent the system state, such as the truth value of a proposition in a set of logical formulae, the temperature in a chemical reactor, or the activation state of a task in a scheduling production problem. All parameters and formulae must be predefined because no mechanism for defining new ones is supported. The interpretation of the parameters is inaccessible because tokens are only related by arithmetic or logical relationships. For example, given the propositional tokens Path_U1_U2 and Path_U2_U3, both having logical values of TRUE (or 1), one cannot as a consequence infer that there exists a path between U1 and U3.

The work we are about to present employs a mapping from propositional logic models to those suitable for input to integer programming procedures. A (MI)LP optimization problem can thus be defined by expressing propositional formulae mathematically as sets of linear constraints (Cavalier and Soyster, 1987) and adding a cost function. This can provide a tool for automatic theorem proving (Hooker, 1988), or for minimizing contradictions in expert systems decisions and default reasoning (Post and Sage, 1990b). In addition, in MIP numerical problems, one can systematically add logical data and heuristics both for the problem formulation and to guide its solution, as is necessary in process synthesis (Raman and Grossman, 1991, 1992).

Symbolic Reasoning. Predicative calculus uses propositional functions. A propositional function is expressed by means of a predicate symbol which specifies a relation between its arguments, a list of undetermined (generic) *individual variables*. The domain of these variables consists of all the objects in the modeled system. When a propositional function is instantiated, its generic values are set to represent specific objects. Then, it becomes a proposition. If the proposition is true, then the objects satisfy the corresponding relation.

In symbolic reasoning, the semantic content of a formula is accessible. Thus, an automatic reasoning procedure can infer a connection between a unit U1 and other units by finding all the elements ?u for which the proposition path (U1, ?u) is satisfied. Symbolic reasoning permits the introduction of quantified variables in the atomic formula. First-order predicative calculus utilizes the universal ($\forall$) and existential ($\in$) quantifying symbols allowing a single formula to stand for several propositions. Those are instantiated in a particular scenario by appropriate symbol substitution. For example, from the formula:

$$\forall(x, y, z)\text{Path}(?x, ?y)\wedge\text{Path}(?y, ?z)\Rightarrow\text{Path}(?x, ?z)$$

and a scenario consisting of the atomic formulas:

Path(U1, U2)
Path(U2, U3),

a new proposition can be inferred: Path(U1, U3).

A symbolic reasoning procedure can be divided into two steps (Post and Sage, 1990a). The first step is *unification*, a

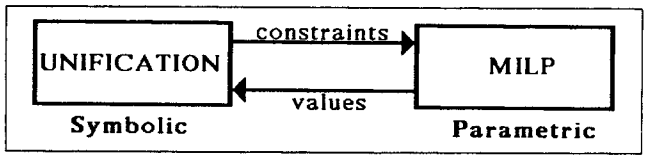

Figure 1. Symbolic and parametric reasoning steps.

pattern matching procedure for the instantiation of general formulae in the scenario of interest. This transforms the generic propositional functions into specific propositions. The second step is a parametric solution procedure. These steps can be applied recursively as new formulae are generated (see Figure 1).

In summary, the approach benefits from the strengths of each of the reasoning steps. On the one hand, working with symbolic models allows the explicit declaration of assumptions concerning the participating objects while also avoiding the declaration of all formulae and parameters beforehand. On the other hand, the transformation to a parametric model allows the integration of logical (qualitative) and numerical considerations in the same framework.

The predicative formalization of knowledge requires a statement of the objects, either abstract or physical, that we are considering and their interrelationships. Qualitative process theory (QPT) appears to be a suitable modeling method for our problem on both accounts.

Qualitative physics (QP) and qualitative process theory

Qualitative physics models the world in qualitative terms on the basis of fundamental physical laws. A review on the subject has been recently published (Ungar and Venkatasubramanian, 1990). Two fundamental approaches can be distinguished in QP: "device-oriented" and "process-oriented." The former assumes that physical systems can be represented by interconnections between devices whose output represents the system behavior. The latter concentrates on processes as the agents for change in the world of interest and is the basis for QPT.

The world of interest is formally described in QPT as a set of physical and abstract objects such as substances, hardware (for example, a vessel), relevant physical variables (for example, a vessel temperature), and their derivatives along with the processes that are taking place. QPT formalizes knowledge usually left implicit in conventional modeling, thereby adding flexibility in diagnosis, hazard operational analysis, process operation, and design.

Quantities—Describing Parameters Values. In QPT, a quantity is made up of an amount and a rate of change. Both are numbers in a *quantity space* subdivided into intervals by relevant landmarks. Quantities exist either within an interval or on a landmark. For example, the temperature, T, of a substance may be: $T < T_{melt}$, $T = T_{melt}$, $T_{melt} < T < T_{boil}$, $T = T_{boil}$ or $T > T_{boil}$. Processes may be initiated or stopped whenever a changing variable quantity crosses a landmark.

Describing the Physical Situation. Relations in a dynamic environment may change in time as a function of the situation. For example, the concept of a liquid level is only valid as long as a liquid is contained in a vessel. In QPT, a situation is described by the activation state of *individual views* and *processes*. *Individual views* (IV) represent relations between world objects, whose activation state is determined by three groups of conditions: (a) individuals: objects that constitute the world of interest; (b) preconditions: nonquantitative conditions expressed as predicates; (c) quantity conditions: ordering relations in the quantity space. An *individual view* is considered to be *active* when all the individuals, preconditions, and quantity conditions are consistent with our world description (scenario). Then, the relations between variables specified in its structure become true. *Processes* are similarly described. Variables change as the result of perturbation or by propagation. Perturbations are described by *influences*, which are relations that arise only as the result of an active process. Propagations are the effect of change by one variable on another. They are modeled by *qualitative proportionality relations*. The description of the world of interest is based on "words" taken from a QPT vocabulary. A rich language enhances reasoning. An advantage of QPT is its focusing of our attention on vocabulary details and structures. A basic "chemical plant" vocabulary has been presented by Catino et al. (1991).

Approach Overview

Detailed planning of an operation can be extremely costly in terms of time and computational load. Hierarchical planning strategies (Sacerdoti, 1974) attempt to reduce this load by dividing the decision space into a hierarchy of abstraction spaces. Planning is then conducted from higher to lower levels of abstraction, delaying the consideration of concrete actions as late as possible. First, the structure- and hardware-related considerations are taken into account to determine the physical phenomena that can potentially take place. For example, if a pump is needed for the transport of material between two vessels, but does not exist in the defined scenario, then for all practical purposes that process cannot take place. Next, decisions are made concerning the requirements of the process and their rate of activation. Data such as the maximum pumping rate of the pump involved are taken into account. Finally, the plan of concrete operators required to implement the required transitions is elaborated. For example, the need to activate the pump is identified.

The strategy adopted for the synthesis of operation procedures is summarized in Figure 2. The operation procedures network (OPNet) is introduced as a modeling tool for the batch processing steps, where nodes in the network are described by

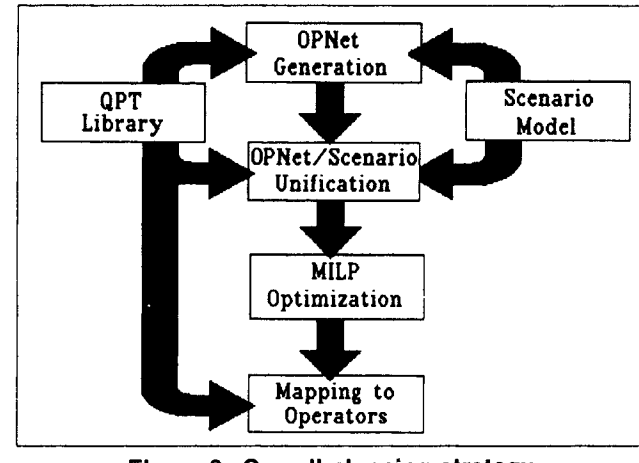

Figure 2. Overall planning strategy.

means of QPT processes and phenomena. The OPNet is discussed in more detail later.

Step 1: OPNet generation

Whenever a production recipe is considered, its description in terms of the developed QPT vocabulary (that is, processes and IVs) is necessary. Alternatively, given a production goal expressed at a high level of abstraction (for example, production of substance B) it is required to find the physical processes (abstract operators) necessary to achieve it. The automatic synthesis of an OPNet still needs to be developed. However, this task was assigned a low priority because while being difficult to automate, the manual generation of an OPNet is quite straightforward (Rotstein et al., 1992).

Step 2: symbolic/parametric mapping

At this stage, the predicative model is mapped into a parametric equivalent. The unification of the OPNet with the plant scenario is implemented. This involves a pattern matching procedure that finds all the possible instances of the desired processes in the scenario. Step 2 corresponds to the translation from a symbolic to a parametric representation. After all the instances have been identified, a MILP multiperiod model of the system is obtained. In the resulting model, the logical constraints (with some preprocessing for efficiency) are combined with the numerical constraints that arise from logical decisions. Additional information (such as the capacity of the units involved and the duration of processes that modify composition) is required at this stage.

Step 3: MILP optimization

The MILP model is now solved. An advantage of the approach is that if some part of the plant equipment is taken off-line, the corresponding instances are easily eliminated from the model with no further reference to the OPNet. Heuristic information, preferences, or partial ordering characteristics of the OPNet structure can be added to reduce the computational load (Rotstein et al., 1994).

Step 4: mapping from the OPNet to concrete operators

The implementation of an operation procedure requires that it be detailed in terms of concrete operator actions. A partially ordered set of concrete operators can be derived from the OPNet and the results of step 3 by extracting the preconditions in every activated instance at the various levels.

The development of adequate models is a central issue in the automation of any process operation task. Before going into further details on the synthesis procedures, we shall now discuss in some detail the modeling strategy here proposed.

Process Modeling

Desired properties

The lack of adequate development tools and methodologies makes modeling an expensive and time-consuming task (Stephanopoulos et al., 1990). Badly selected model characteristics often impede the success in achieving the desired goals. We shall now enumerate and define model attributes which facilitate the development and maintenance of process models:

(*a*) *Declarative.* Assumptions about the described system are stated explicitly as an integral part of the model, and separately from the procedural aspects of problem solving (Stephanopoulos et al., 1990; Piela et al., 1991). This should include different classes of knowledge, such as plant structure, the expected physical processes, or the user preferences. This property is needed not only to simplify model updating, but also to facilitate the analysis of the results obtained.

(*b*) *Flexible.* Allows the model to be easily modified to account for changes in the processing routes or in the physical plant where they are implemented. In general, there is a strong correlation between model declarativeness and its flexibility.

(*c*) *Reusable.* Models that are being developed can be used as a basis for the description of new plants or processes (Piela et al., 1991).

(*d*) *Multipurpose.* Models for different operating tasks, such as the simulation, diagnosis, scheduling or control of the process, and sharing the same database. Thus, information can be easily exchanged between the tasks.

(*e*) *Process-Oriented.* A distinction is made between the description of the physical phenomena and that of the physical equipment in which they take place. This enhances model flexibility and its adaptability to serve different objectives. For example, planning would involve a search for the physical phenomena that should be implemented (Rotstein et al., 1992), while fault diagnosis would involve searching for unexpected phenomena and the corresponding structural changes that could explain an abnormal behavior (Grantham and Ungar, 1990).

Predicative/numerical modeling approach

Traditionally, the process modeling paradigm has relied on a parametric approach. However, parametric models suffer on the one hand from the need to explicitly state all formulae *a priori* and, on the other hand, the model assumptions are left implicit and hidden. They obviously lack in the desired properties listed above. In general, parametric models are tailored to a particular problem and their modification is therefore more involved. Furthermore, the creation of high level declarative languages requires symbolic manipulation which cannot be achieved through parametric programming alone.

Setting the Stage. An alternative "Predicative/Numerical" paradigm for process operation models is proposed in order to avoid the limitations of the parametric approach. The model has initially the form:

$$\begin{aligned} \text{Model\_1:}\quad & p_{\text{lib}}(x, v, c) \\ & h_{\text{lib}}(x, v, c) \\ & p_s(V) \\ & h_s(x, v, c, X, V, C_s) \\ & f_s(X, C_s) = 0 \\ & g_s(X, C_s) \le 0 \\ & C_s(V) = 0 \end{aligned} \tag{1}$$

where P_{lib} is a library of predicative description of the processes that can take place, the conditions required for their activation and their effect. Here, an extended version of QPT is adopted, in which each process structure includes a description of the constraints associated with it. These constraints stand for the

modifications or additions to a parametric model of the system whenever a given process can potentially be activated. h_{lib} is a library of generic heuristics valid for the whole class of considered problems, p_s is a predicative description of the scenario structure and hardware, h_s represents scenario (problem specific) heuristics, f_s represents scenario equality constraints, g_s represents scenario inequality constraints, and v, x, and c are generic individual variables, generic process state variables, and generic process parameters, respectively. V represents individual constants representing plant hardware and structure, X represents scenario instantiated state variables, and C_s represents scenario parameters.

The goals are expressed as: $P_{goal}(x, v, c, X, V, C_s)$ where P_{goal} is a predicative description. Variables in lowercase letters in Model__1 indicate generic (noninstantiated) variables, while those in capital letters represent specific ones. For example, "p" may indicate a generic pressure variable while "P__T1" may stand for the instantiation of p for container T1. One should recognize that the generic libraries in Model__1 include a statement of the technologies that will be considered in the eventual model.

Connecting to the Real World. The key step in the mapping from the symbolic to the parametric representation is the *unification* of the generic expressions with those specified in the given scenario. This results in a numerical/propositional model. The two-stage reasoning approach extends naturally to numerical functions and variables. For example, the expression:

$$\text{Container}(?x) \Rightarrow \text{Temperature}(?x)$$

indicates that given any vessel $?x$ there is a generic variable temperature($?x$), its temperature. Unification of this expression with some particular scenario will yield specific variables: temperature(A1), temperature(A2), ..., temperature(An), corresponding to the temperature at various plant locations.

The unification stage is critical, since subsequent to this step, the space of feasible solutions for the problem considered is completely defined. A typical example from the area of process design is the concept of the *superstructure* (Grossmann, 1990), a representation that aims to contain all the candidates that are to be considered for the optimal solution of the considered problem. At the unification stage, one has essentially two main options. One option is to include all the feasible instances at a considerable computational cost that will show up in consequent solution steps. Alternatively, one may apply problem specific knowledge and heuristics to trim the solution space. In the relatively small examples shown in this presentation, we choose to adopt the former approach and consider all options. The alternative approach is discussed elsewhere (Rotstein et al., 1994).

After unification, Model__1 will have been transformed into:

$$\begin{aligned}&\text{Model\_\_2:} \quad P_{goal}(X, V, C_s)\\ &\text{s.t.}\\ &\qquad p_{(lib \cup s)}(X, V, C_s)\\ &\qquad h_s(X, V, C_s)\\ &\qquad f_s(X, C_s) = 0\\ &\qquad g_s(X, C_s) \le 0\\ &\qquad C_s(V) = 0 \end{aligned} \tag{2}$$

Converting into a Standard Solvable Form. As a final stage, the propositions in Model__2 must be reduced to numerical form (Post and Sage, 1990b; Raman and Grossmann, 1991) obtaining the classical mathematical programming formulation, stated as:

$$\begin{aligned}&\text{Model\_\_3:} \quad \text{Find the extremum of:}\\ &\qquad f_{goal}(X, I_V, C_s, \mu)\\ &\text{s.t.}\\ &\qquad pc_{(lib \cup s)}(I_V) \le 0\\ &\qquad hc_{(lib \cup s)}(X, I_V, C_s) \le \mu\\ &\qquad f_{(lib \cup s)}(X, I_V, C_s) = 0\\ &\qquad g_{(lib \cup s)}(X, I_V, C_s) \le 0 \end{aligned} \tag{3}$$

where $pc_{(lib \cup s)}$ are constraints representing the propositional expressions and terms coming from the scenario and library descriptions, $hc_{(lib \cup s)}$ are constraints representing the heuristics coming from the scenario and library descriptions, I_V are variables representing the propositional terms coming from the scenario and library descriptions, μ is a penalty for the violation of the heuristic induced constraints, $f_{(lib \cup s)}$ are the LHS of a set of equations added by instantiation of library processes in the considered scenario, combined with the scenario specific constraints, and $g_{(lib \cup s)}$ are the LHS of a set of inequalities added by instantiation of library processes in the considered scenario, combined with the scenario specific constraints.

Combined qualitative/quantitative process library

In order to account for both the quantitative and qualitative aspects of a problem, the original QPT processes structure was extended to include quantitative constraints. These are the changes that are introduced to a parametric model of the system whenever a possible instantiation for a library process is found in the scenario of interest (see Appendix I, under constraints). The logical format of a process structure is:

$$\bigwedge_i a_i(x, v, c) \Rightarrow f_{lib}(x, v, c), \quad i = 1, \ldots, n \text{ predicates} \tag{4}$$

where a is a set of n predicates implied by the process and which describe the conditions under which the constraints f_{lib} are active. The convention $\bigwedge_i$ implies the logical AND operating on the predicates, as illustrated in the following example:

$$\text{If:} \quad \text{Substance}(?s) \wedge \text{Container}(?x) \Rightarrow \text{Mass}(?s, ?x) \le \text{Capacity}(?x) \tag{5}$$

and ($?s = A$, $?x = U1$) is an instantiation of the process in the scenario, then the constraint:

$$\text{Mass\_\_}A\text{\_\_}U1 \le \text{Capacity\_\_}U1$$

is obtained. Note that the convention used to name the state variables and parameters involved in f_{lib} is based on the names of the concrete objects in the scenario as well as the instances at which the process is instantiated. Also note that in the accepted formulation, the data is generated in the form of a matrix inequality ($Ax \le b$), expressing the constraints in Eq. 3. The constraints describe the potential effects of process activation on the instance state variables, as well as the process

activation requirements. Their form depends on the model structure. Here, we adopt a QPT compatible multiperiod formulation for the batch plant scheduling problem (see the next section). Other formulations are possible.

The generation and modification of the constraints, requires the definition of an appropriate compiling vocabulary that facilitates the modular, automatic construction of a parametric model. The vocabulary elements constitute the building blocks employed for the description of the process *constraints* sections. The following are examples of vocabulary elements generated:

(1) (*relax-mass-balance-constraints* ?substance ?container ?relative-rate). When a process is activated, it becomes possible to modify the amount of substance ?substance in vessel ?container by a quantity arising from ?relative-rate.

(2) (*Add-constraints* ?direction). This adds new constraints with the (in)equality sign ?direction ($1 \equiv \leq$, $g \equiv \geq$, $e \equiv =$).

(3) (*Add-parameters* ?variable-form ?parameter ?time-shift). This adds ?parameter to the corresponding ?variable-form matrix column and to the rows defined by the last *add-constraints* statement. The ?time-shift (in time units) specifies a lead or lag time between the task activation instant and the variable time.

A typical extended qualitative/quantitative process structure is shown in Figure 3. The *relations* and *influences* sections are omitted from the figure.

Up to this point, we have presented a new predicative/numerical paradigm for model formulation, and discussed its relation to classical mathematical programming. An extended version of QPT was defined and developed to permit its implementation. A complete account of the implementation issues is presented elsewhere (Rotstein, 1993). The potential of

```
(setf 1-component-pumping (make-process
 :form          '(1-component-pumping ?src ?dst ?pump)
 :I.V.s         '((contained-1-component-liquid ?s1 ?c))
 :individual-elements
                '((container ?dst)
                  (pump ?pump)
                  (can-pump ?s1 ?pump)
                  (can-contain ?s1 ?dst))
 :individual-conditions
                '((fluid-connected ?c ?pump)
                (fluid-connected ?pump ?dst))
 :preconditions '((connect ?c ?pump)
                  (connect ?pump ?dst))
 :constraints   '((relax-mass-constraints ?s1 ?c -1)
                  (relax-mass-constraints ?s1 ?dst 1)
                  (add-constraints 'g)
                  (add-parameters (mass-variable-form ?s1 ?c) 1 0)
                  (add-parameters (mass-variable-form 'all ?c) -1 0)
                  (add-parameters (task-variable-form) (- (parameter 'Cmax ?c)) 0)
                  (add-rhs (- (parameter 'Cmax ?c)))
                  (add-constraints 'l)
                  (add-parameters (rate-variable-form) 1 0)
                  (add-parameters (task-variable-form)(-(parameter 'RMax ?pump)) 0)
                  )))
```

Figure 3. Example of an extended QPT process description.

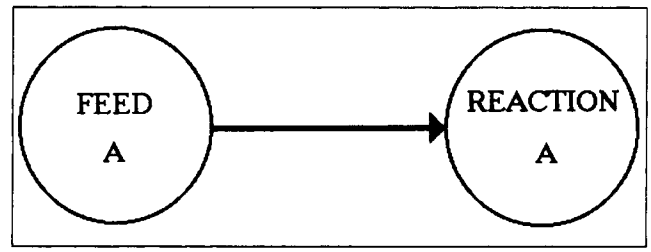

Figure 4. Simplified task network.

QPT as a tool for the modeling of batch processing recipes and for the planning of operations will now be discussed.

OPNet Representation

In batch processing, a raw material undergoes a series of transformations in order to yield a desired product. There may exist several alternative paths leading from an initial (or intermediate) state to a goal (or subgoal). Each path is initiated by physical actions causing changes in the activation status of processes. In conventional batch operation terminology (Bristol, 1985), transformations are grouped into *procedures*, consisting of one or more *phases*, which in turn are formed by a collection of simple *steps*. *Phases* are delimited by points where processing may be held for external intervention or where there are changes in the required processes or resources. A common abstract representation for batch procedures is a directed task network, where the task nodes denote phases. A simple example is presented in Figure 4 where the objective is to produce B starting from material A through the catalytic reaction: $A \xrightarrow{Z} B$. Traditionally, batch phases have been modeled as an activation sequence operating on the processing units. This requires predetermination of the production paths in the particular plant considered. In addition, the effects of the activation of different phases are described employing inflexible subsets of (in)equalities. The "*process-oriented*" approach adopted here focuses on the various physical phenomena that need to take place and their partial ordering in time. The phenomena are modeled using the QPT vocabulary. Phases are described as combinations of QPT processes (processes are the source of change) and IVs, grouped into a detailed node-arc structure, termed the *operation procedures network* (OPNet). Each process adds its particular set of constraints to the phase description, whose mathematical structure is therefore flexible. The relationship between the OPNet and conventional batch operation terminology is shown in Figure 5.

The OPNet representation is divided into two hierarchical levels of abstraction. The higher level is a digraph, whose nodes (*task nodes*) represent a set of physical phenomena lumped into a single batch phase, and whose arcs indicate their required order of execution. Next, each task node is further detailed as a new digraph, expanding the task to expose its *processes* and *individual views* (OPNet nodes). Those are connected by directed arcs which specify partial time ordering (Rotstein et al.,

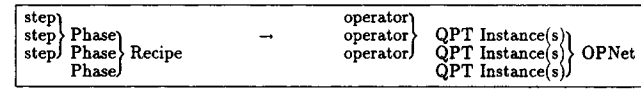

Figure 5. Vocabulary used.

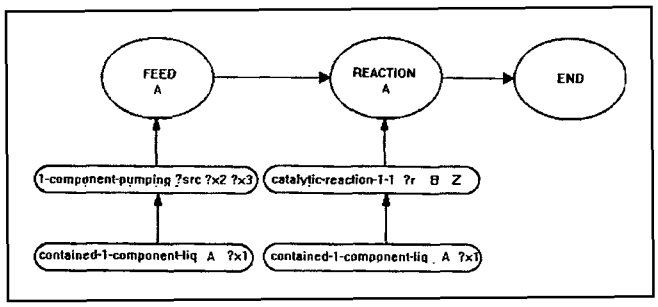

Figure 6. OPNet example.

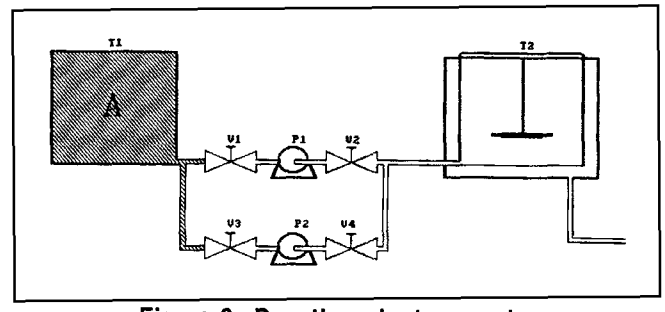

Figure 8. Reaction plant example.

1992). The two-level network model has been implemented using (object-oriented) LISP syntax (Rotstein, 1993). This approach maintains flexibility in the level of abstraction; the individual elements appearing as arguments of processes or IVs may be specified as generic or specific, thus permitting the association of a task with particular hardware. The activation of a phase requires *all* nodes within it to be active.

At this stage, nothing is stated with respect to *unification*, that is, the association of all task nodes to particular plant hardware items. This will be done at a later stage. Clearly, the size of the optimization problem that will be eventually generated depends on the generality of the OPNet formulation. One may reduce the solution space as well as avoid unnecessary or incorrect task instantiation by implanting case-dependent information in a task node. This is accomplished in two ways: (a) by stating task-specific ontological conditions or preconditions, or (b) by replacing generic variables with plant-specific items. However, this must be done with care to avoid missing promising production strategies.

Figure 6 shows the expanded OPNet for the task-network presented in Figure 4. It provides an abstract, process-oriented description of the procedure without relating it to a particular plant scenario or to hardware availability. The network describes a "physical phenomena sequence" rather than a sequence of processing units. At this level, issues such as unit connectivity or suitability to processing tasks are disregarded. These are modeled explicitly within each phase, through the individuals specified in the QPT library for the relevant *processes* and *individual views*. For example, the individuals required by the feeding phase retrieved from the library are summarized in the next subsection discussing unification.

A case that highlights the modular nature of our modeling approach is the representation of a distillation task. The separation between two hypothetical components G and H is modeled in Figure 7 using five QPT structures. Three processes must take place simultaneously: the distillation itself, and heat-transfer processes in the reboiler and the condenser.

An attempt can now be made to implement the OPNet on the example plant illustrated in Figure 8 and modeled in association with the predicative/numerical scenario given in Figure 9.

Symbolic to Parametric Mapping

The mapping from the predicative to the parametric descriptions is implemented in three steps:

(1) The unification of the OPNet description of the processing steps with the processing plant (scenario);

(2) The compilation of the constraints listed in the library description of the processes used; and

(3) The inference of disjunction constraints $[\bigvee_i y_i \Rightarrow \sum_i y_i \leq 1, y_i \in (0, 1)]$ from the hardware requirements specified in the *individuals* section.

We shall now consider each of these three steps in detail.

Unification

This step corresponds to the mapping from Model_1 to Model_2. Operation phases are described by means of QPT processes. The various possible instances for each phase are found using the following algorithm:

(i) Select a new phase z.

(ii) Extract its individual elements and individual conditions from P_{lib}, the library description of the processes and IVs involved. The set of predicates $a_z(v_z)$, representing the resources required by the phase z is obtained. Add the task specific conditions, if defined.

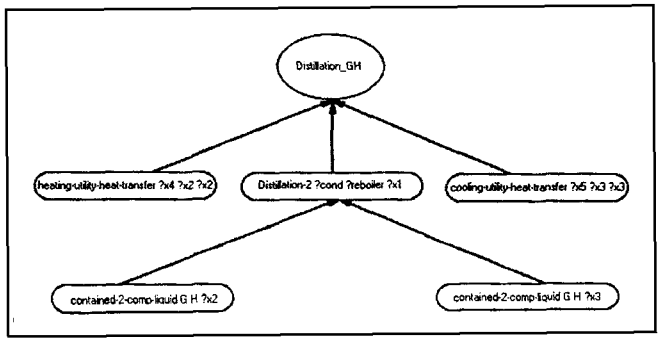

Figure 7. OPNet description for task: *Distillation* GH.

Individuals	Ontological Conditions	
(vessel T1)	(access-to Z T2)	(can-contain A T1)
(vessel T2)	(fluid-connect T1 P1)	(can-contain A T2)
(pump P1)	(fluid-connect P1 T2)	(can-contain B T2)
(pump P2)	(fluid-connect T1 P2)	(can-pump A P1)
(substance A)	(fluid-connect P2 T2)	(can-pump A P2)
(substance B)	(has-mixer T2)	
(catalyst Z1)	(catalyst-for Z A B)	

Initial mass of A	200 kg	Time Horizon, H,5 time units (arbitrary)	
Capacity(T1)	500 kg	Capacity(T2)	100 kg
Capacity(P1)	50 kg/time unit	Capacity(P2)	25 kg/time unit
Conversion	25 kg/time unit	Reaction time	2 time units

Figure 9. Predicative/numerical scenario for the reactor system.

(iii) Unify $a_z(v_z)$ with the corresponding scenario description $P_s(V)$. An instance of z is a set of values $V_z \subseteq V$ s.t. $a_z(V_z) \subseteq P_s(V)$.

(iv) Repeat steps i to iii for all phases.

In our simple example, the only process required to realize the task node "Feed_A" is the 1-component-pumping process (see Figure 6). On performing step ii of the algorithm, we extract the following list of individual-elements:

$$a_{Feed\_A}(v_{Feed\_A}) = \{(\text{container ?dst})\ (\text{pump ?pump})\ (\text{can-pump ?s1 ?pump})\ (\text{can-contain ?s1 ?dst})\ (\text{substance ?s1})\ (\text{can-contain ?s1 ?c})\ (\text{fluid-connect ?c ?pump})\ (\text{fluid-connect ?pump ?dst})\}. \quad (6)$$

Finally, in step iii, $a_{Feed\_A}(v_{Feed\_A})$ is unified with the scenario $P_s(V)$ described by Figure 9. Two instances are obtained:

$$\{?c = T1,\ ?dst = T2,\ ?pump = P1,\ ?s1 = A\} \quad (7a)$$

$$\{?c = T1,\ ?dst = T2,\ ?pump = P2,\ ?s1 = A\} \quad (7b)$$

Compilation of Constraints

Each time a phase instance, a_z, is obtained, a set of constraints (f_{lib}) is added to the parametric model. These are extracted from the *constraints* sections of the processes describing the task. For the instance (Eq. 7a) of the pumping process in our example, the following constraints are extracted (see also Figure 3):

```
(relax-mass-constraints A T1 -1)
(relax-mass-constraints A T2 1)
(add-constraints 'g)
 (add-parameters (mass-variable-form A T1) 1 0)
 (add-parameters (mass-variable-form 'all T1) -1 0)
 (add-parameters (task-variable-form) (- (parameter 'Cmax
 T1)) 0)
 (add-rhs (-(parameter 'Cmax T1)))
(add-constraints '1)
 (add-parameters (rate-variable-form) 1 0)
 (add-parameters (task-variable-form)(-(parameter 'RMax
 P1)) 0)
```

The first of these constraints states that by activating that particular instance one reduces the amount of A in T1 by an amount specified by the rate variable. The last three statements constrain the magnitude of this variable according to the pumping capacity of P1. In addition, task "Feed_A" must be activated before any change can take place. The eventual compiled constraint resulting here takes the form:

$$\text{Task-Rate(Feed\_A, t)} - \text{RMax\_P1} \times \text{Task-State(Feed\_A, t)} \le 0 \quad (8)$$

Inference of disjunction constraints

The assumption set of phase z, a_z, can be divided into two different subsets. One, a_z^0, consists of statements that represent ontological impositions on the plant scenario (for example, that a container is suitable to contain a particular substance). The other, a_z^a, declares availability demands (for example, that a unit cannot serve simultaneously two tasks). These requirements give rise to disjunction constraints in the parametric model.

Blind application of the two-stage reasoning strategy (that is, unification followed by solution) could lead to an unnecessarily large problem size. It would assign a binary variable to each statement in a_z at every time increment. However, one can recognize in the problem structure means for a considerable reduction of its size (See Appendix II). Specifically:

- Ontological conditions need be considered only at the unification step, when the plant scenario is checked. It is assumed that these conditions remain unchanged along the selected time horizon.
- The availability of units for the various task instances at any time instant, need not be expressed by independent binary variables. These variables can be lumped into one (binary) task instance variable representing the instance activation state along the desired horizon.

The unification step and the inference of disjunction constraints steps carry out the translation from Model_1 to Model_3. The MILP model thus obtained describes the physical changes that can take place when the OPNet is implemented in the particular scenario. The MILP optimization procedure will then yield the decisions concerning the activation of tasks in space and time.

MILP Multiperiod Formulation of the Batch Plant Operation Problem

To keep the problem tractable, we adopt a linear multiperiod approach, similar to the state-task network (STN) formulation used by Kondili et al. (1993). The time horizon of interest (that is, for which a plan is being developed) is divided into periods of equal duration. Three items are central in our formulation: the tasks to be activated, the hardware, manpower and energy resources available to implement those tasks, and the mass of the materials that are being processed in the plant. The state of these items must be considered at the boundary of each time period. These are achieved through the following variables: m_{cjt} is the mass of component c in unit j ($j \in U$) at time t, h_{ist} is the rate of mass change introduced by instance s of task i at time t, and I_{ist} is defined as the activation state of instance s of task i at time interval t (1 or 0). For a horizon of H time intervals, the time periods will be numbered from 0 to H. Task activation or deactivation is possible only at the start of those periods, allowing synchronization between the tasks. The value of mass variables at time $t+1$ is a result of the processes active between t and $t+1$.

In MILP modeling, logical considerations are represented by binary variables. Operational decisions require the modeling of constraints that become active subject to specific logical conditions. In addition, the quantitative effect of those decisions must also be described. These aims are achieved by expressing the relevant constraints in terms of a combination of real and binary variables. The model consists of:

- Component mass balances.
- Allocation constraints, which express conflicts in resource assignment.
- Container holdup limitations.
- Task rate limitations.
- Limitations on soft resources (for example, energy or manpower).
- Initial mass fraction required and expected conversion rates.
- Required task activation interval.
- Plant Initial conditions.

The mass balances constitute the skeleton of the proposed model formulation. The change on the mass contents of a component A at unit j can be stated as:

$$m_{Aj}(t+dt) - m_{Aj}(t) = \sum_i \sum_s I_{IS}(t) \int_t^{t+\Delta t} \dot{r}_{Ais}(x_{cj}, T_j, P_j, t)dt \quad (9)$$

where $\dot{r}_{Ais}$ is component A modification rate for instance s of task i as a function of the concentrations, temperature and pressure at unit j and time t, and $I_{is}(t)$ is the activation state for instance s of task i at time t.

These constrain the possible changes in the mass of each component in each container along the time horizon of interest. If no physical process take place in a particular unit, its mass contents remain unchanged. Activating a task in particular equipment items relaxes the corresponding constraints up to an amount dictated by the selected task instances. For example, the activation of the flow of a substance from one vessel to another using a given pump permits a change in the mass of the substance in both vessels up to an amount determined by the pump characteristics. Then, optimizing any cost function from $t=0$ to the time horizon H, subject to the mass balances shown above, gives rise to a MINLP problem. However, the multiperiod approach facilitates model linearization by lumping the effect of nonlinearities as global linear changes along the selected time period:

Assume $\begin{cases} P_j(t) \text{ and } T_j(t) \text{ are given for each task} \\ \text{Decisions only at discrete times (multiperiod model).} \end{cases}$

Then, the general form of the mass balances is:

$$m_{Ajt+\Delta t} - m_{Ajt} = \sum_i \sum_s I_{ist} R_{Ais}(x_{cj}) \quad (10)$$

where R_{Ais} is the maximum rate of change for component A by the activation of instance s of task i along Δt time units.

The instances s of tasks i are part of a set of task instances that can modify the mass contents of c in unit j. Each activation of an instance of *relax-mass-balance-constraints* (as defined earlier) for component c and unit j, will add an additional term to the sum on the right side of Eq. 10.

If R_{is} is not a function of the concentrations, x_{cj}, Eq. 10 is already linear. Otherwise, additional inequalities are added to specify the desired initial concentrations and rate profile when the task is executed. In the formulation proposed, the form and number of the model constraints vary from task to task, depending on the processes that are activated at each particular phase. It is therefore useful to have an automatic compiler to simplify the construction of the MILP data files.

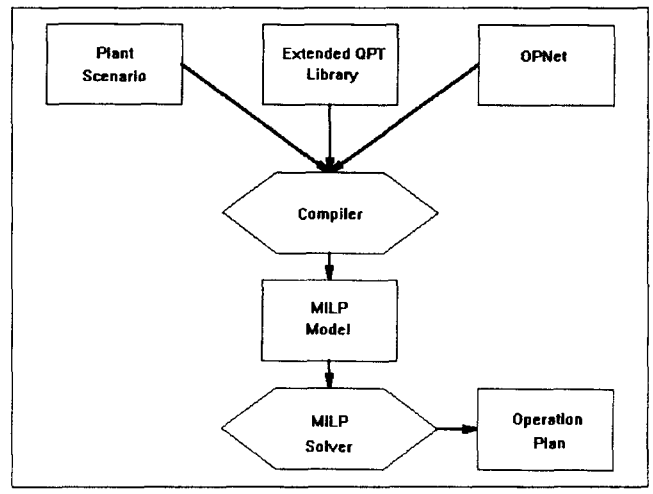

Figure 10. Structure of S2P.

Software Implementation and Case Study

The modeling and the symbolic-parametric mapping methods have been implemented as a CommonLisp program named S2P. The resulting MILP models are solved using CPLEX(1991), a commercial LP, and MILP optimization package. The software's data and procedural structure are described in Figure 10. The program compiles the Process Library, the plant scenario and the OPNet description into an MPS model of the system. The MPS format is a widely used standard for the definition of LP and MILP problems. A case study is presented below.

Case study

The example is a multiproduct problem in which three products can be obtained:

$$M1 \xrightarrow{Z12} M2$$

$$M1 + M3 \xrightarrow{\text{Exothermic}} M4 + M5$$

$$M2 + M3 \xrightarrow{\text{Blend}} M6$$

Here, one of the products, M2, can be recycled (blended) to obtain an additional salable material, M6. The first step in solving this problem according to the proposed strategy is to write down a task network. This is shown in Figure 11.

Next, the tasks must be modeled in terms of the library processes and individual views required to activate them (see Appendix I for a description of selected items in the extended qualitative/quantitative library). This results in the OPNet description of the processing steps. Task specific conditions and preconditions must also be included at this stage. Figure 12 illustrates a typical exothermic reaction task in the network.

Modeling tasks such as Store__M2 or Reaction__M1 requires a single process and individual view from the library. On the other hand, the representation of the task for the exothermic

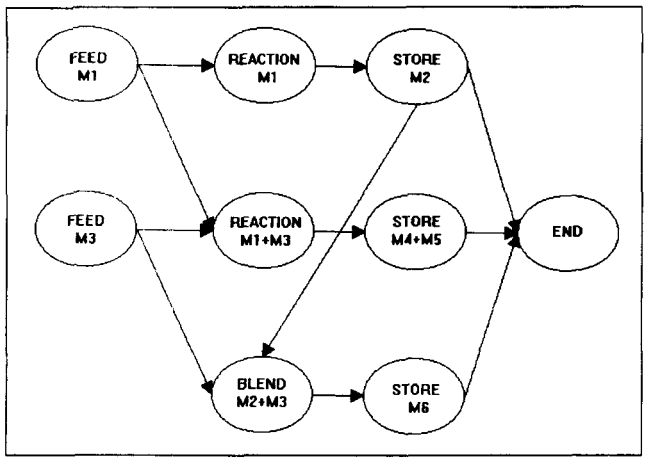

Figure 11. Task network for case study.

reaction of M1 and M3 demands a combination of two processes, one describing the reaction and the other the heat-transfer process required for cooling.

Finally, the description of the batch recipe is completed by defining the process constraints' parameters and the predicates describing expected chemical materials and processes. After specifying the OPNet, the plant must be modeled. Both plant topology and process constraint parameters are specified in the predicative/numerical scenario data file, shown in Figure 13. Figure 14 illustrates the flowsheet of the plant considered.

The next step in the planning strategy demands the instantiation of the tasks and the generation of the corresponding model (in)equalities. In the example, the program recognizes that the task "reaction_M1" can only take place in T2, since this is the only vessel equipped to handle the catalyst Z12. Similarly, the exothermic "Reaction_M1 + M3" can only take place in T5, since it is the only vessel equipped with a cooling system. On the other hand, there are three different instantiations for the task "Store_M2." The individual conditions corresponding to the pumping process involved in storing M2 are satisfied by pump P3 (T2→T3 or T5→T3) and by pump P2 (T3→T5).

As mentioned previously, the number of feasible instantiations for a task can be reduced by sacrificing some of the OPNet's generality. In the example, one instance of the task "Store_M2" can be eliminated by specifying that M2 can *only* be stored in T3. This specification would prevent the activation of the Blending task, which would then remove M6 as one of the potential products of the plant.

The task instances are automatically found by S2P, using the algorithm described in the section on symbolic to parametric mapping resulting in a MILP model. In the present case study with a time horizon of 12 h, a model consisting of 91 binary variables, 771 constraints and 234 continuous variables was generated.

Before the optimization problem can be solved, the objectives (preferences and goals) must be stated. A methodology to facilitate this step is under development (Rotstein et al. 1994). Here, the case study is optimized on the basis of an arbitrary cost function that maximizes the production of all three products, with the price of M6 assumed to be three times that of M2 and of M4/M5. The branch and bound procedure

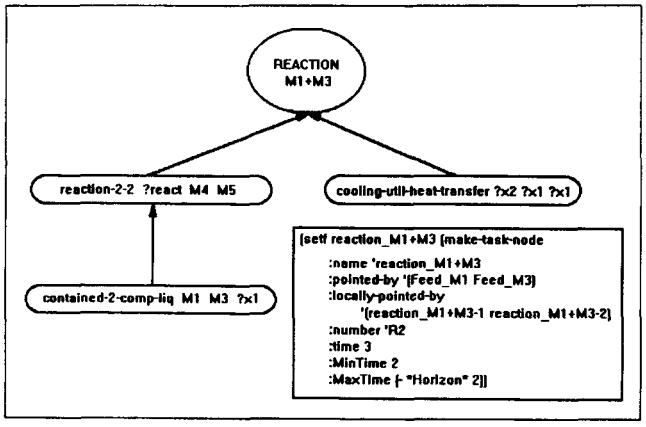

Figure 12. OPNet description for task node: *Reaction_M1 + M3.*

```
(container T1)          (fluid-connect T2 P3)   (container T6)
(can-contain M1 T1)     (fluid-connect P3 T3)   (can-contain M4 T6)
(container T3)          (fluid-connect P1 T5)   (can-contain M5 T6)
(can-contain M2 T3)     (fluid-connect T5 P3)   (pump P2)
(container T2)          (fluid-connect P3 T6)   (can-pump M2 P2)
(can-contain M1 T2)     (fluid-connect T4 P1)   (pump P3)
(can-contain M2 T2)     (fluid-connect T3 P2)   (can-pump M5 P3)
(pump P1)               (fluid-connect P2 T5)   (can-pump M4 P3)
(can-pump M3 P1)        (fluid-connect P3 T7)   (can-pump M2 P3)
(can-pump M1 P1)        (fluid-connect T1 P1)   (can-pump M6 P3)
(container T4)          (fluid-connect P1 T2)   (mixer-at T5)
(can-contain M3 T4)     (container T5)          (reaction-2-2 M1 M3 M4 M5)
(substance M1)          (can-contain M6 T5)     (cool-utility C1)
(substance M2)          (can-contain M2 T5)     (heat-connect T5 C1 T5)
(substance M3)          (can-contain M5 T5)     (Blend M2 M3 M6)
(substance M4)          (can-contain M3 T5)     (mixer-at T2)
(substance M5)          (can-contain M1 T5)     (catalyst-for Z12 M1 M2)
(substance M6)          (can-contain M4 T5)     (can-introduce Z12 T2)
(catalyst Z12)          (container T7)          (can-contain M6 T7)

(Cmax_T1 250)           (Coef_M1_Reaction_M1 1)   (Coef_M1_Reaction_M1M3 0.5)
(Cmax_T2 100)           (Coef_M2_Reaction_M1 1)   (Coef_M3_Reaction_M1M3 0.5)
(Cmax_T3 200)           (X_M1_Reaction_M1 1)      (Coef_M4_Reaction_M1M3 0.5)
(Cmax_T4 100)           (Phi_Reaction_M1_0 0.5)   (Coef_M5_Reaction_M1M3 0.5)
(Cmax_T5 100)           (Phi_Reaction_M1_1 0.5)   (X_M1_Reaction_M1M3 0.5)
(Cmax_T6 200)           (RMax_Reaction_M1 0.5)    (X_M3_Reaction_M1M3_0 0.5)
(Cmax_T7 200)           (X_M2_Blend_M2M3 0.5)     (Phi_Reaction_M1M3_0 0.5)
(Qmax_C1 500)           (X_M3_Blend_M2M3 0.5)     (Phi_Reaction_M1M3_1 0.3)
(Init_M1_T1 250)        (X_M6_Blend_M2M3 0.)      (Phi_Reaction_M1M3_2 0.2)
(Init_M3_T4 100)        (Phi_Blend_M2M3_0 0.5)    (RMax_Reaction_M1M3 0.5)
(RMax_P1 50)            (Phi_Blend_M2M3_1 0.5)    (CoolDem_Reaction_M1M3 2)
(RMax_P2 50)            (RMax_Blend_M2M3 0.5)
(RMax_P3 200)           (X_M4_Store_M4M5 0.5)     (*HORIZON* 12)
                        (X_M5_Store_M4M5 0.5)     (δ 0.25)
```

Figure 13. Predicative/numerical scenario for case study.

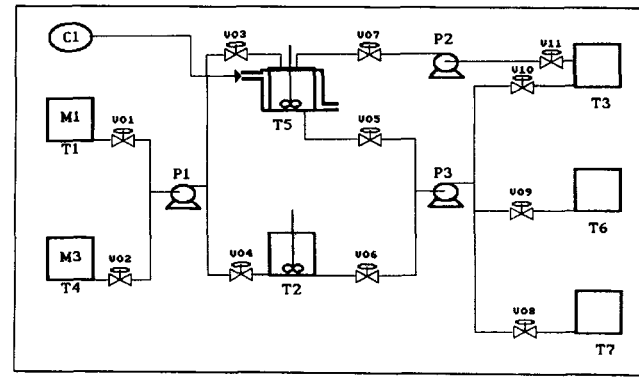

Figure 14. Case study batch plant.

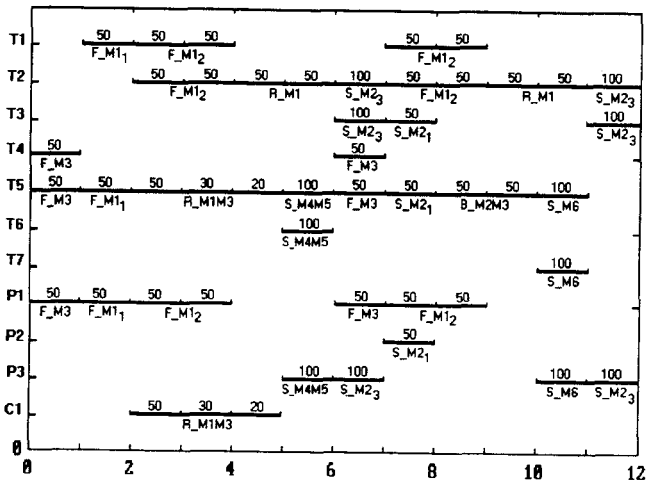

Figure 15. Gantt chart for case study.

required the solution of 111 LPs and took 71.9 s CPU time on a SPARC2 machine.

The results are summarized as a Gantt Chart (in Figure 15) and as a partially ordered plan of actions (in Figure 16). The Gantt chart describes the tasks activated at each time, the units and utilities involved and the amount of material being processed.

The MILP results can also be translated into a partially ordered plan of actions. This relies on the connection that has been established between the predicative and the numerical aspects of the process model (discussed earlier). The required actions are extracted from the library processes preconditions. The actions are qualified by the initiation and stopping times of the relevant processes. Figure 16 illustrates a plan for a simple example in which only M2 is produced. This example involved the examination of 221 nodes, and took 94.4 s.

The computational load could have been further reduced by eliminating all the constraints and variables associated with tasks that are irrelevant for the production of M2. This was done by recompiling the MILP model for the case in which only the task "Store_M2" is connected to the node "END" in the directed graph in Figure 11. These steps reduced the branch and bound tree to 24 nodes, whose optimization took 2.5 s.

Time=0
- Activate Feed_M1 — Process 100 units
  - (connect T1 P1)
  - (connect P1 T2)
  - (activate P1)

Time=2
- Stop Feed_M1
  - ~ (connect T1 P1)
  - ~ (connect P1 T2)
  - ~ (activate P1)
- Activate Reaction_M1 — Process 100 units
  - (connect-mixer T2)
  - (introduce-catalyst Z12 T2)

Time=4
- Stop Reaction_M1
  - ~ (connect-mixer T2)
- Activate Store_M2 — Process 100 units
  - (connect T2 P3)
  - (connect P3 T3)
  - (activate P3)

Time=5
- Stop Store_M2
  - ~ (connect T2 P3)
  - ~ (connect P3 T3)
  - ~ (activate P3)
- Activate Feed_M1 — Process 100 units
  - (connect T1 P1)
  - (connect P1 T2)
  - (activate P1)

Time=7
- Stop Feed_M1
  - ~ (connect T1 P1)
  - ~ (connect P1 T2)
  - ~ (activate P1)
- Activate Reaction_M1 — Process 100 units
  - (connect-mixer T2)
  - (introduce-catalyst Z12 T2)

Time=9
- Stop Reaction_M1
  - ~ (connect-mixer T2)
- Activate Store_M2 — Process 100 units
  - (connect T2 P3)
  - (connect P3 T3)
  - (activate P3)

Time=10
- Stop Store_M2
  - ~ (connect T2 P3)
  - ~ (connect P3 T3)
  - ~ (activate P3)

Figure 16. Partially ordered plan of actions.

Conclusions

The integration of artificial intelligence and optimization methods, relying on the "predicative/numerical" modeling paradigm, appears to be a powerful approach. Its application permits the generation of models that can be easily modified and reused. These properties are particularly important for the flexible synthesis of batch plant operation procedures. They facilitate the implementation of model modifications that can account for changes in the processing routes or in the physical hardware where they are implemented.

The *process-oriented approach* and the *OPNet description* introduced here facilitate the complete decoupling of the abstract processing steps required from any specific plant implementation. The conservation expressions, the basic assumptions in which they rely, and the actions required to set them active are all modeled by one homogeneous representation using an extended version of QPT. The particularities of process operations synthesis can then be combined with the modified QPT structure to define an adequate order of abstraction spaces. The overall method involves the generation of the OPNet, its transformation to a parametric MILP model, and its mapping to a sequence of concrete simple operators. The method has been realized in the program S2P, which automatically generates a MILP model of the problem. The program requires three data files: one for the "predicative/numerical" scenario, one describing the proposed OPNet, and one containing a generic quantitative/qualitative process library.

Some issues require additional attention. For one thing,

accounting for case-specific knowledge at the unification stage could relieve the computational load on the MILP solution step (by reducing the number of task instances). This can be achieved at the expense of increasing the load on the OPNet instantiation step and has the potential risk of losing promising alternative operational paths. There is clearly a tradeoff between the model generality and the tractability of the solution procedure. Another important issue is the need to deal with multiple objectives. Both of these topics are discussed in Rotstein et al. (1994).

Acknowledgment

This research was supported by the Basic Research Foundation administered by the Israeli Academy of Sciences and Humanities and by the Fund for the Promotion of Research at the Technion.

Notation

C_i = subset of components involved in task i
F_i = fixed resource task requirements
P_{goal} = description of the objectives

Subscripts

c = component
i = task number
j = unit number
s = task instance number
t = time instant

Literature Cited

Aelion, V., J. Kalagnaman, and G. J. Powers, "Evaluation of Operating Procedures Based on Stationary-State Stability," *Ind. Eng. Chem. Res.*, **31**, 2532 (1992).
Beale, E. L. M., "Branch and Bound Methods for Mathematical Programming Systems," *Ann. Discrete Math.*, **5**, 201 (1979).
Bristol, E. H., "A Design Tool Kit for Batch Process Control: Terminology and a Structural Model," *InTech*, 10, 47 (1985).
Catino, C., S. D. Grantham and L. H. Ungar, "Automatic Generation of Qualitative Models of Chemical Process Units," *Comp. & Chem. Eng.*, **15**, 583 (1991).
Cavalier, T. M., and A. L. Soyter, "Logical Deduction Via Linear Programming," IMSE Working Papers 87-147, Pennsylvania State Univ. (1987).
CPLEX, *CPLEX Manual*, Optimization Inc. (1993).
Crawford, J., A. Farquhar, and B. Kuipers, "QPC: A Compiler from Physical Models into Qualitative Differential Equations," *Proc. Conf. on Artificial Intelligence*, AAAI-90 (1990).
Crooks, C. A., and S. Macchietto, "A Combined MILP and Logic-based Approach to the Synthesis of Operating Procedures for Batch Plants," *Chem. Eng. Comm.*, **11**, 117 (1992).
Forbus, K. D., and B. Falkenhainer, "Self-Explanatory Simulations: An Integration of Qualitative and Quantitative Knowledge," *Proc. Conf. on Artificial Intelligence*, AAAI-90 (1990).
Forbus, K. D., "Qualitative Process Theory," *AI*, 7 (1984).
Fusillo, R. H., and G. J. Powers, "A Synthesis Method for Chemical Plant Operating Procedures," *Comp. & Chem. Eng.*, **11**, 369 (1987).
Grantham, S. D., and L. H. Ungar, "A First Principles Approach to Automated Trouble Shooting of Chemical Plants," *Comp. & Chem. Eng.*, **14**, 783 (1990).
Grossman, I., "MINLP Optimization Strategies and Algorithms for Process Synthesis," in *Foundations of Computer-Aided Design*, J. J. Siirola, I. E. Grossman, and G. Stephanopoulos, eds., Elsevier Science Publishers, Amsterdam (1990).
Hooker, J. N., "Resolution Vs. Cutting Plane Solution of Inference Problems: Some Computational Experience," *Oper. Res. Lett.*, **7**, 1 (1988).
Kondili, E., C. C. Pantelides, and R. W. H. Sargent, "A General Algorithm for Short Term Scheduling of Batch Operations: I. MILP Formulation," *Comp. & Chem. Eng.*, **17**, 211 (1993).
Lakshmanan, R., and G. Stephanopoulos, "Synthesis Of Operating Procedures for Complete Chemical Plants: I. Hierarchical Structural Modeling for Nonlinear Planning," *Comp. & Chem. Eng.*, **12**, 985 (1988a).
Lakshmanan, R., and G. Stephanopoulos, "Synthesis Of Operating Procedures for Complete Chemical Plants: II. A Nonlinear Planning Methodology," *Comp. & Chem. Eng.*, **12**, 1003 (1988b).
Piela, P. C., T. G. Epperly, K. M. Westerberg, and A. W. Westerberg, "ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis: The Modeling Language," *Comp. & Chem. Eng.*, **15**, 53 (1991).
Post, S., and A. P. Sage, "An Overview of Automated Reasoning," *IEEE Trans. on Sys., Man and Cyb.*, **20**, 202 (1990a).
Post, S., and A. P. Sage, "Default Reasoning Using Least Exception Logic," *Inf. and Decision Technol.*, **16**, 43 (1990b).
Raman, R., and I. E. Grossmann, "Integration of Logic and Heuristic Knowledge in MINLP Optimization for Process Synthesis," *Comp. & Chem. Eng.*, **16**, 155 (1992).
Raman, R., and I. E. Grossmann, "Relation between MILP Modeling and Logical Inference for Chemical Process Synthesis," *Comp. & Chem. Eng.*, **15**, 73 (1991).
Rivas, R. J., D. F. Rudd, and L. R. Kelly, "Computer-Aided Safety Interlock Systems," *AIChE J.*, **20**, 311 (1974).
Rotstein, G. E., R. Lavie, and D. R. Lewin, "Incorporating Shallow Knowledge in Batch Operations Models," *Comp. & Chem. Eng.*, **18**(Suppl.), S499 (1994).
Rotstein, G. E., "Operation of Chemical Processes using Reasoning," D.Sc. Thesis, Israel Institute of Technology, Haifa, Israel (1993).
Rotstein, G. E., R. Lavie, and D. R. Lewin, "A Qualitative Process-Oriented Approach for Chemical Plant Operations—The Generation of Feasible Operation Procedures," *Comp. & Chem. Eng.*, **16**(Suppl.), S337 (1992).
Sacerdoti, E. D., "Planning in a Hierarchy of Abstraction Spaces," *AI*, **5**, 115 (1974).
Stephanopoulos, G., G. Henning, and H. Leone, "MODEL.LA. A Modeling Language for Process Engineering—I. The Formal Framework," *Comp. & Chem. Eng.*, **14**, 813 (1990).
Ungar, L. H., and V. Venkatasubramanian, "Knowledge Representation," *Artificial Intelligence in Process System Engineering III*, G. Stephanopoulos and J. F. Davis, eds., CACHE Monograph Series, Austin, TX (1992).

Appendix 1: Sample Entries in Extended Qualitative/Quantitative Process Library

Processes

```
(setf 2-component-pumping
 (make-process
  :form          '(2-component-pumping ?src ?dst ?pump)
  :IVs           '((contained-2-component-liquid ?s1 ?s2 ?c))
  :ind-elements  '((Container ?dst)
                   (pump ?pump))
```

(continued)

Appendix 1: Sample Entries in Extended Qualitative/Quantitative Process Library (continued)

```
    :ind-conditions   '((fluid-connect ?c ?pump)
                      (fluid-connect ?pump ?dst)
                      (can-pump ?s1 ?pump)
                      (can-pump ?s2 ?pump)
                      (can-contain ?s1 ?dst)
                      (can-contain ?s2 ?dst))
    :preconditions    '(connect ?c ?pump)
                      (connect ?pump ?dst)
    :constraints      '((relax-mass-constraints ?s1 ?c(-(parameter 'x ?s1 *task*)))
                      (relax-mass-constraints ?s1 ?dst (parameter 'x ?s1 *task*))
                      (relax-mass-constraints ?s2 ?c (-(parameter 'x ?s2 *task*)))
                      (relax-mass-constraints ?s2 ?dst (parameter 'x ?s2 *task*))
                      (add-concentration-constraints ?s1 ?c 0)
                      (add-concentration-constraints ?s2 ?c 0)
                      (add-constraints '1)
                      (add-parameters (rate-variable-form) 1 0)
                      (add-parameters (task-variable-form) (-(parameter 'RMax ?pump)) 0)
                      (add-constraints 'g)
                      (add-parameters (rate-variable-form) 1 0)
                      (add-parameters (task-variable-form) (-(*(parameter 'RMax ?pump)*Eps*)) 0)))))
  (setf reaction-2-2
   (make-process
    :form             '(reaction-2-2 ?react ?prod1 ?prod2)
    :IVs              '((contained-2-component-liquid ?s1 ?s2 ?c))
    :ind-conditions   '((reaction-2-2 ?s1 ?s2 ?prod1 ?prod2)
                      (can-contain ?prod1 ?c)
                      (can-contain ?prod2 ?c)
                      (mixer-at ?c))
    :preconditions    '(connect-mixer ?c)
    constraints       '((relax-mass-constraints ?s1 ?c (-(parameter 'coef ?s1 *task*)))
                      (relax-mass-constraints ?s2 ?c (-(parameter 'coef ?s2 *task*)))
                      (relax-mass-constraints ?prod1 ?c (parameter 'coef ?prod1 *task*))
                      (relax-mass-constraints ?prod2 ?c (parameter 'coef ?prod2 *task*))
                      (add-task-initial-concentration-constraints ?s1 ?c)
                      (add-task-initial-concentration-constraints ?s2 ?c)
                      (add-task-rate-profile ?s1 ?c)
                      (add-constraints '1)
                      (add-parameters (rate-variable-form) 1 0)
                      (add-parameters (task-variable-form)(-(*(parameter 'RMax *task*)
                                (parameter 'CMax ?c))) 0)
                      (add-constraints 'g)
                      (add-parameters (rate-variable-form) 1 0)
                      (add-parameters (task-variable-form) (*(parameter 'RMax *task*)
                                (parameter 'CMax ?c))*Eps*)) 0))))
  (setf catalytic-reaction-1-1
   (make-process
    :form             '(catalytic-reaction-1-1 ?react ?prod1 ?cat)
    :IVs              '((contained-1-component-liquid ?s1 ?c))
    :ind-elements     '((catalyst ?Cat))
    :ind-conditions   '((catalyst-for ?cat ?s1 ?prod1)
                      (can-introduce ?cat ?c)
                      (can-contain ?prod1 ?c)
                      (mixer-at ?c))
    :preconditions    '(connect-mixer ?c)
                      (introduce-catalyst ?cat ?c)
    :constraints      ((relax-mass-constraints ?s1 ?c (-(parameter 'coef ?s1 *task*)))
                      (relax-mass-constraints ?prod1 ?c (parameter 'coef ?prod1 *task*))
```

(continued)

Appendix 1: Sample Entries in Extended Qualitative/Quantitative Process Library (continued)

```
                    (add-task-initial-concentration-constraints ?s1 ?c)
                    (add-task-rate-profile ?s1 ?c)
                    (add-constraints '1)
                    (add-parameters (rate-variable-form) 1 0)
                    (add-parameters (task-variable-form)(-(*(parameter 'RMax *task*)
                                (parameter 'CMax ?c))) 0)
                    (add-constraints 'g)
                    (add-parameters (rate-variable-form) 1 0)
                    (add-parameters (task-variable-form)
                                (-(*(*(parameter 'RMax *task*)
                                (parameter 'CMax ?c))*Eps*)) 0))))
(setf cooling-utility-heat-transfer
 (make-process
  :form          '(cooling-utility-heat-transfer ?cooler ?src ?dst)
  :ind-elements  '((cool-utility ?cooler)
                  (container ?src)
                  (container ?dst))
  :ind-conditions '((heat-connect ?src ?cooler ?dst))
  :preconditions '(connect ?src ?cooler)
                  (connect ?cooler ?dst)
  :constraints   '((add-relative-resource-demands ?cooler
                                                  (Parameter 'CoolDem *task*)))))
(setf Blending-2
 (make-process
  :form          '(Blending-2 ?components ?mixture)
  :IVs           '((contained-2-component-liquid ?s1 ?s2 ?c))
  :ind-conditions '((blend ?s1 ?s2 ?mixture)
                  (mixer-at ?c))
  :preconditions '(connect-mixer ?c)
  :constraints   '((relax-mass-constraints ?s1 ?c (-(parameter 'x ?s1 *task*)))
                  (relax-mass-constraints ?s2 ?c (-(parameter 'x ?s2 *task*)))
                  (relax-mass-constraints ?mixture ?c 1)
                  (add-task-initial-concentration-constraints ?s1 ?c)
                  (add-task-initial-concentration-constraints ?s2 ?c)
                  (add-task-rate-profile ?s1 ?c)
                  (add-constraints '1)
                  (add-parameters (rate-variable-form) 1 0)
                  (add-parameters (task-variable-form) (-(*(parameter 'RMax *task*)
                                (parameter 'CMax ?c))) 0)
                  (add-constraints 'g)
                  (add-parameters (rate-variable-form) 1 0)
                  (add-parameters (task-variable-form)
                                (-(*(*(parameter 'RMax *task*)
                  (parameter 'CMax ?c)) *Eps*)) 0))))
(setf distillation-2
 (make-process
  :form          '(distillation-2 ?cond ?reb ?column)
  :IVs           '((contained-2-component-liquid ?s1 ?s2 ?c1)
  :ind-elements  '((column ?column))
  :ind-conditions '((reboiler ?c1 ?column)
                  (condensor ?c2 ?column))
  :constraints   '((relax-mass-constraints ?s1 ?c1 (-(parameter 'coef ?s1 *task*)))
                  (relax-mass-constraints ?s2 ?c1 (-(parameter 'coef ?s2 *task*)))
                  (relax-mass-constraints ?s1 ?c2 (parameter 'coef ?s1 *task*))
                  (relax-mass-constraints ?s2 ?c2 (parameter 'coef ?s2 *task*))
                  (add-task-initial-concentration-constraints ?s1 ?c1)
                  (add-task-initial-concentration-constraints ?s2 ?c1)
```

(continued)

Appendix 1: Sample Entries in Extended Qualitative/Quantitative Process Library (continued)

```
(add-task-rate-profile ?s1 ?c1)
(add-constraints '1)
(add-parameters (rate-variables-form) 1 0)
(add-parameters (task-variable-form) (-(*(parameter 'RMax *task*)
                (parameter 'CMax ?c1))) 0)
(add-constraints 'g)
(add-parameters (rate-variable-form) 1 0)
(add-parameters (task-variable-form)
                (-(*(*(parameter 'RMax *task*)
                (parameter 'CMax ?c1))*Eps*)) 0))))
```

Individual views

```
(setf contained-3-component-liquid (make-I.V.
:form           '(contained-3-component-liquid ?s1 ?s2 ?s3 ?c)
:ind-elements   '((container ?c)
                (substance ?s1)
                (substance ?s2)
                (substance ?s3))
ind-conditions  '((can-contain ?s1 ?c)
                (can-contain ?s2 ?c)
                (can-contain ?s3 ?c))))
```

Appendix 2: Size Reduction of MILP Model

The complete propositional formulation for the activation of an instance of any task z at time t, $\text{Instance}_z(t)$ is:

$$I_z(t) \Rightarrow \text{can-activate}(\text{Instance}_z(t)) \qquad \text{(A1)}$$

$$\text{can-activate}(\text{Instance}_z(t)) \Leftrightarrow a_z^0 \wedge a_z^a(t) \qquad \text{(A2)}$$

$I_z(t)$: activation state of Instance z.

That is, in order to activate the Instance_z at any time t, it should be possible to activate it at that time. The instance can be activated if the instance assumptions are true at the instant considered. The truth value of the assumptions a_z^0 is checked at the unification step. If they are not satisfied by the scenario then the truth value of $I_z(t)$ will be false (that is, zero) for all the time horizon, and it is unnecessary to include a binary variable to indicate the activation state of that instance. On the other hand, if a_z^0 is found to be true, then only the $a_z^a(t)$ assumptions should be checked along the time horizon. Under this assumption we can simplify Eqs. A2 and A1 to:

$$I_z(t) \Rightarrow a_z^a(t) \qquad \text{(A3)}$$

Whenever $I_z(t)$ is activated, $a_z^a(t)$ will be given a truth value of one. We can therefore conclude that the binary variables representing the truth value of $a_z^a(t)$ at the disjunction constraints can be replaced by the binary variable $I_z(t)$ for the corresponding task instance.

Manuscript received Mar. 29, 1993, and revision received Oct. 4, 1993.