Computers chem. Engng, Vol. 15, No. 2, pp. 73–84, 1991
Printed in Great Britain.

0098-1354/91 $3.00 + 0.00

RELATION BETWEEN MILP MODELLING AND LOGICAL INFERENCE FOR CHEMICAL PROCESS SYNTHESIS

R. RAMAN and I. E. GROSSMANN†
Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A.

(Received 30 January 1990; final revision received 25 June 1990; received for publication 15 October 1990)

Abstract—The objective of this paper is to show that qualitative knowledge in process synthesis that can be expressed in propositional logic form has an equivalent representation as linear equations and inequalities. Recent contributions in operations research are reviewed for this purpose including inference problems that can be formulated as MILP problems that can be solved mostly as relaxed LPs. It will also be shown how some of these ideas can be applied to chemical process synthesis. Also, comparisons with production systems are presented, as well as the application of the propositional logic-based approach to the systematic modelling of integer constraints that commonly arise in synthesis problems. Several examples are presented to illustrate the ideas.

INTRODUCTION

Researchers in process synthesis currently rely on two basic approaches, mathematical programming and artificial intelligence (AI)—to handle problems in this area. AI techniques rely on qualitative information (e.g. heuristics) to make design decisions. They do not require a detailed analysis and therefore are fast. On the other hand, AI techniques do not search for optimal solutions since they are concerned mainly with finding one or several feasible solutions. Traditionally, most synthesis decisions have been made based on qualitative considerations. Therefore, even today, preliminary design is thought to be ideally suited for qualitative analysis. Siriola *et al.* (1971), Mahalec and Motard (1977) and more recently Douglas (1985), Lien *et al.* (1987), Beltramini *et al.* (1989) and Stephanapoulos *et al.* (1987) have met with certain success in capturing the design procedure through qualitative approaches.

Mathematical programming approaches, on the other hand, are starting to receive increased attention because they have several advantages over the qualitative approach. For instance, they can capture interactions between variables very effectively, which the qualitative approach cannot and this is one of its major shortcomings. Furthermore, the design model is more accurate and the search more rigorous, so that the resulting design is at least optimum within the specified assumptions and alternatives that are considered for the search. Mathematical programming techniques, however, can require considerable computational expense and effort, although this has become less of a problem with the development of new algorithms and parallel computing. In this approach, the initial work was based on NLP techniques (Umeda *et al.*, 1972), then on MILP techniques (Papoulias and Grossmann, 1983) and more recently, has evolved to the use of MINLP techniques that allows the use of nonlinear systems with discrete and continuous variables (Grossmann, 1989).

At the start of any design process, one has available a certain amount of quantitative and qualitative information. In chemical process synthesis, the quantitative information may be in the form of rate equations, models of various process units and correlations for thermodynamic properties. The qualitative information is in the form of heuristics and prior knowledge on decisions that are likely to yield an acceptable design.

Ideally, one would like to be able to use both the qualitative and quantitative information available about the problem at hand in order to obtain an optimal solution quickly. In order to do that effectively, one must be in a position to process both kinds of information from within the same framework. Expressing qualitative knowledge mathematically would serve as a first step towards integrating the two forms of knowledge.

The purpose of this paper is to show how one can express mathematically logical expressions that make up qualitative knowledge in a systematic manner by reviewing some important recent contributions in the area, and by showing the relevance of these ideas in the domain of chemical process synthesis. Post (1987) and Cavalier and Soyster (1987) have shown how propositional logic expressions can be expressed mathematically as linear constraints and also how to make inferences from this type of model. Specifically, logical variables are replaced by integer binary variables, and reasoning is shown to be equivalent to solving a mixed-integer linear programming (MILP) model which in fact can

†To whom all correspondence should be addressed.

be solved as a linear programming problem in many cases (Hooker, 1988). Uncertain knowledge, such as heuristics, can also be easily handled.

Aside from illustrating the potential application of MILP models for logical inference in chemical process synthesis, this paper will present some comparisons with the production systems approach for handling qualitative information. In addition, it will be shown that modelling of integer constraints in mixed-integer programming formulations that arise in design optimization problems can be performed in a systematic manner using the logic-based framework. An approach to model constraints that contain both integer and continuous variables is discussed. Further extending this line of thought one can also formalize the modelling of discontinuous and nondifferentiable functions by using integer variables. Several examples will be presented to illustrate the basic concepts.

MOTIVATING EXAMPLE

In order to show how qualitative information can be expressed in an equivalent mathematical manner through linear constraints, consider the following simple example in the synthesis of separation systems where an important heuristic rule is "Remove the most plentiful component first". One can express this rule for component A in a multicomponent mixture as a logical expression by:

$$\text{PLENTY\_OF\_A} \Rightarrow \text{SEPARATE\_A}, \qquad (1)$$

where PLENTY_OF_A and SEPARATE_A are logical variables denoting whether A is the most plentiful component and whether the component A is to be separated from the mixture, which is the design decision. We can associate the binary variable y_1 with the logical variable PLENTY_OF_A and the binary variable y_2 with the logical variable SEPARATE_A. Let a value of TRUE for the logical variables correspond to a value of 1 for the binary variables and a value of FALSE for the logical variables correspond to a value of 0 for the binary variables. Then the logical relation can be expressed as

$$y_1 - y_2 \leqslant 0, \quad y_1, y_2 = \{0, 1\}. \qquad (2)$$

In this way, if $y_1 = 1$ (PLENTY_OF_A is TRUE), then the only way that the above inequality can be satisfied is by letting $y_2 = 1$ (SEPARATE_A is TRUE) which is exactly what the logical expression represents. Therefore, the inequality is a precise representation of the logical expression. Also it should be noted that for the case when the heuristic is allowed to be violated, the above inequality can be modified by introducing a nonnegative slack variable, v such that:

$$y_1 - y_2 \leqslant v, \qquad (3)$$

where a penalty can be associated with the violation of the inequality. Note that if $v = 0$, the heuristic is satisfied, while $v = 1$ implies that the heuristic is violated.

Since qualitative information is often expressed in the form of more complex logical relationships than in this simple example, it is not always trivial to intuitively develop a corresponding mathematical representation. The procedure to systematically convert logical expressions into their equivalent mathematical representation is discussed in the next section. The procedure will then be applied to the modelling of inference problems and to the modelling of discrete constraints that commonly arise in synthesis problems.

MATHEMATICAL REPRESENTATION OF LOGICAL RELATIONSHIPS

In order to obtain an equivalent mathematical representation for any propositional logic expression, one must first consider basic logical operators to determine how each can be transformed into an equivalent representation in the form of an equation or inequality. These transformations are then used to convert general logical expressions into an equivalent mathematical representation (Cavalier and Soyster, 1987).

The basic unit of propositional logic expression, which can correspond to a state or to an action, is called a literal which is a single variable that can assume either of two values, true or false. Associated with each literal P, there is another literal NOT P ($\neg P$) such that either P or ($\neg P$) is always true. A clause is a set of literals separated by OR operators and is also called a disjunction. A proposition is any logical expression and consists of a set of clauses P_i, $i = 1, \ldots, r$ that are related by the logical operators OR, AND, IMPLICATION.

To each proposition P_i, a binary variable y_i is assigned. Then the negation or complement of $P_i(\neg P_i)$ is given by $1 - y_i$. The logical value of true corresponds to the binary value of 1 and false corresponds to the binary value of 0. The basic operators used in propositional logic and the representation of their relationships are shown in Table 1. From this table, it is easy to verify, for instance, that the logical implication in (1) reduces to the inequality in (2).

With the basic equivalent relations given in Table 1 (e.g. see Williams, 1988), one can systematically model an arbitrary propositional logic expression that is given in terms of OR, AND, IMPLICATION operators, as a set of linear equality and inequality constraints. One approach is to systematically convert the logical expression into its equivalent *conjunctive normal form* representation which involves the application of pure logical operations. The conjunctive normal form is a conjunction of clauses, $Q_1 \wedge Q_2 \wedge \cdots \wedge Q_s$. Hence, for the conjunctive normal form to be true, each clause Q_i must be true independent of the others. Also since a clause Q_i is just a disjunction of literals, $P_1 \vee P_2 \vee \cdots \vee P_r$, it can be

Table 1. Representation of logical relations with linear inequalities

Logical relation	Comments	Logical expression	Representation as linear inequalities
Logical "OR"		$P_1 \vee P_2 \vee \cdots \vee P_r$	$y_1 + y_2 + \cdots + y_r \geqslant 1$
Logical "AND"		$P_1 \wedge P_2 \wedge \cdots \wedge P_r$	$y_1 \geqslant 1; y_2 \geqslant 1; \ldots; y_r \geqslant 1$
Implication	$P_1 \Rightarrow P_2$ is logically equivalent to $\neg P_1 \vee P_2$	$\neg P_1 \vee P_2$	$1 - y_1 + y_2 \geqslant 1$ or $y_1 - y_2 \leqslant 0$
Equivalence	P_1 if and only if P_2 $(P_1 \Rightarrow P_2) \wedge (P_2 \Rightarrow P_1)$	$(\neg P_1 \vee P_2) \wedge (\neg P_2 \vee P_1)$	$y_1 - y_2 \leqslant 0; y_2 - y_1 \leqslant 0$ or $y_1 = y_2$
Exclusive "OR" (EOR)	Exactly one of the variables is true	$P_1 \oplus P_2 \oplus \cdots \oplus P_r$	$y_1 + y_2 + \cdots + y_r = 1$
Classification	$Q = \{P_1, P_2, \ldots, P_r\}$ Q is true if any of the variables inside the brackets are true		$y_q = y_1 + \cdots + y_r$

expressed in the linear mathematical form as the inequality:

$$y_1 + y_2 + \cdots + y_r \geqslant 1. \tag{4}$$

The procedure to convert a logical expression into its corresponding conjunctive normal form was formalized by Clocksin and Mellish (1981). The systematic procedure consists of applying the following three steps to each logical proposition:

(1) replace the implication by its equivalent disjunction:

$$P_1 \Rightarrow P_2 \Leftrightarrow \neg P_1 \vee P_2; \tag{5}$$

(2) move the negation inward by applying DeMorgan's Theorem:

$$\neg(P_1 \wedge P_2) \Leftrightarrow \neg P_1 \vee \neg P_2, \tag{6}$$

$$\neg(P_1 \vee P_2) \Leftrightarrow \neg P_1 \wedge \neg P_2; \tag{7}$$

(3) recursively distribute the "OR" over the "AND" by using the following equivalence:

$$(P_1 \wedge P_2) \vee P_3 \Leftrightarrow (P_1 \vee P_3) \wedge (P_2 \vee P_3). \tag{8}$$

Having converted each logical proposition into its conjunctive normal form representation, $Q_1 \wedge Q_2 \wedge \cdots \wedge Q_s$, it can then be easily expressed as a set of linear equality and inequality constraints.

The following example illustrates the procedure for converting logical expressions into inequalities.

Example 1

Consider the proposition

$$(P_1 \wedge P_2) \vee P_3 \Rightarrow P_4 \vee P_5. \tag{9}$$

By removing the implication, the above yields from (5):

$$\neg[(P_1 \wedge P_2) \vee P_3] \vee P_4 \vee P_5. \tag{10}$$

Further, from (6) and (7), moving the negation inwards leads to the following two steps:

$$[\neg(P_1 \wedge P_2) \wedge \neg P_3] \vee P_4 \vee P_5, \tag{11}$$

$$[(\neg P_1 \vee \neg P_2) \wedge \neg P_3] \vee P_4 \vee P_5. \tag{12}$$

Recursively distributing the "OR" over the "AND" as in (8) the expression becomes:

$$(\neg P_1 \vee \neg P_2 \vee P_4 \vee P_5) \wedge (\neg P_3 \vee P_4 \vee P_5), \tag{13}$$

which is the conjunctive normal form of the proposition involving two clauses. Translating each clause into its equivalent mathematical linear form, the proposition is then equivalent to the two constraints:

$$\begin{aligned} y_1 + y_2 - y_4 - y_5 &\leqslant 1, \\ y_3 - y_4 - y_5 &\leqslant 0. \end{aligned} \tag{14}$$

LOGICAL INFERENCE

From the above example it can be seen that logical expressions can be represented by a set of inequalities. An integer solution that satisfies all the constraints will then determine a set of values for all the literals which makes the logical system consistent. This is a logical inference problem where given a set of n logical propositions, one would like to prove whether a certain clause is always true. The problem can be stated as:

$$\begin{aligned} &\text{prove} && P_u \\ &\text{s.t.} && B(P_1, P_2, \ldots, P_q), \end{aligned} \tag{15}$$

where P_u is the clause to be proved and B is the set of logical propositions P_i, $i = 1, 2, \ldots, q$ that must hold.

Given that all the logical propositions have been converted to a set of linear inequalities, the inference problem can be formulated as the following MILP (Cavalier and Soyster, 1987):

$$\begin{aligned} Z = \text{minimize} &\sum_{i \in I(u)} c_i y_i, \\ \text{s.t.} \quad & Ay \geqslant a, \\ & y \in \{0, 1\}^n, \end{aligned} \tag{16}$$

where $Ay \geqslant a$ is the set of inequalities obtained by translating $B(P_1, P_2, \ldots, P_q)$ into their linear mathematical form, and the objective function is obtained by also converting the clause P_u that is to be proved into its equivalent mathematical form. Here, $I(u)$ corresponds to the index set of the binary variables associated with the clause P_u. This clause is always true if $Z = 1$ on minimizing the objective function as an integer programming problem. If $Z = 0$ for the optimal integer solution, this establishes an instance where the clause is false. Therefore, in this case, the clause is not always true.

In many instances, the optimal integer solution to problem (16) will be obtained by solving its linear programming relaxation (Hooker, 1988). Even if no integer solution is obtained, it may be possible to reach conclusions from the relaxed LP problem if the solution is one of the following types (Cavalier and Soyster, 1987):

1. $Z_{\text{relaxed}} > 0$: The clause is always true even if $Z_{\text{relaxed}} < 1$. Since Z is a lower bound to the solution of the integer programming problem, this implies that no integer solution with $Z = 0$ exists. Thus the integer solution will be $Z = 1$.
2. $Z_{\text{relaxed}} = 0$, and the solution is fractional and unique: The clause is always true because there is no integer solution with $Z = 0$.

For the case when $Z_{\text{relaxed}} = 0$ and the solution is fractional but it is not unique one cannot reach any conclusions from the solution of the relaxed LP. The reason is that there may be other integer valued solutions to the same problem with $Z_{\text{relaxed}} = 0$.

In this way, just by solving the relaxed linear programming problem in (16), one might be able to make inferences. The following example will illustrate a simple application in process synthesis.

Example 2

Reaction path synthesis involves the selection of a route for the production of the required products starting from the available raw materials. All chemical reactions can be expressed in the form of propositional logic and can therefore be represented by linear mathematical relations. The specific example problem is to investigate the possibility of producing H_2CO_3 given that certain raw materials are available and the possible reactions. This example problem was presented by Mahalec and Motard (1977) who solved it using the resolution principle.

The chemical reactions are given by:

$$H_2O + CO_2 \rightarrow H_2CO_3,$$
$$C + O_2 \rightarrow CO_2, \tag{17}$$

assuming that H_2O, C and O_2 are available.

Expressing the reactions in logical form yields:

$$H_2O \wedge CO_2 \Rightarrow H_2CO_3,$$
$$C \wedge O_2 \Rightarrow CO_2. \tag{18}$$

The objective is to prove whether H_2CO_3 can be formed given that H_2O, C, O_2 are available. Define binary variables corresponding to each of C, O_2, CO_2, H_2O and H_2CO_3. Translating the above logical expressions into linear inequalities, the inference problem in (16) becomes the following MILP problem:

$$Z = \min y_{H_2CO_3}$$
$$\text{s.t. } y_{H_2O} + y_{CO_2} - y_{H_2CO_3} \leq 1,$$
$$y_C + y_{O_2} - y_{CO_2} \leq 1,$$
$$y_{H_2O} = 1,$$
$$y_C = 1,$$
$$y_{O_2} = 1,$$
$$y_C, y_{O_2}, y_{CO_2}, y_{H_2O}, y_{H_2CO_3} \in \{0, 1\}^5. \tag{19}$$

The objective involves the minimization of $y_{H_2CO_3}$ because the objective is to prove H_2CO_3. Solving the relaxed LP problem yields an integer solution with $Z = 1$ and $y_{H_2CO_3} = y_{CO_2} = 1$. This solution is then interpreted as "H_2CO_3 can always be produced from H_2O, C and O_2 given the above reactions."

LOGICAL INFERENCE WITH UNCERTAIN KNOWLEDGE

The problem most often encountered in design and process synthesis is to select the best flowsheet/design for producing the required product starting with the available raw materials. In order to obtain a "good" design (not necessarily optimal), it must satisfy as much as possible, the qualitative knowledge about the system.

The qualitative knowledge available about the design of a system can be classified as one of the following two types—hard logical facts or uncertain heuristics. Hard, logical facts are never violated—for example, the reaction $NaOH + HCl \rightarrow NaCl + H_2O$ holds from basic chemical principles. Qualitative knowledge in the form of heuristics on the other hand are just rules of thumb which may not always hold. Therefore all the knowledge for synthesizing a design may not be consistent since the heuristics may contradict one another; for example, a rule that suggests to use higher temperatures to increase yield may conflict with a rule that suggests to use lower temperature to increase selectivity. Resolution of conflicts is an important part of reasoning. In general, one must violate a weaker (more uncertain) set of rules in order to satisfy stronger ones. Therefore, it becomes necessary to model the violation of heuristics, which is done as follows (Post, 1987):

$$\text{Clause or V}, \tag{20}$$

where either the clause is true or the clause being violated (V) is true. Since the clause is also a disjunction, the conversion of (20) into the mathematical linear form is simple—the V just adds on to the constraint. For example,

$$[\neg P_1 \vee \neg P_2 \vee P_3 \vee P_4] \vee \text{V}, \tag{21}$$

yields

$$1 - y_1 + 1 - y_2 + y_3 + y_4 + v \geq 1, \tag{22}$$

where v can also be interpreted as a slack variable that allows the violation of the inequality. The variable v, which can be treated as a continuous variable, will take on values of 0, 1 only due to the logical condition in (20) since the clause itself takes on only integer values.

In order to discriminate between weak and strong rules, penalties are associated with the violation v_i of each heuristic rule, $i = 1, \ldots, m$. The penalty w_i is a nonnegative number which reflects the uncertainty of the corresponding logical expression. The more uncertain the rule, the lower the penalty for its violation. The value of the penalty can be supplied by the designer based on his/her experience and confidence in the heuristic rule. This could make the selection of the penalties w_i somewhat subjective. However, the penalties can be determined systematically if a quantitative optimization model is available for the same problem as discussed in Raman and Grossmann (1990). The total weighted penalty for the qualitative model can be associated with the equation:

$$Z = w^T v. \tag{23}$$

In this way, the logical inference problem with uncertain knowledge can be formulated as an MILP problem where the objective is to obtain a solution that satisfies all the logical relationships (i.e. $Z = 0$), and if that is not possible, to obtain a solution with the least total penalty for violation of the heuristics. This leads to the following MILP problem:

$$\begin{aligned} &\min Z = w^T v, \\ &\text{s.t.} \quad Ay + v \geq a: \quad \text{heuristics}, \\ &\qquad By \geq b: \quad \text{logical facts}, \\ &\qquad y \in \{0, 1\}^n, \quad v \geq 0. \end{aligned} \tag{24}$$

Note that no violations are assigned to the inequalities $By \geq b$, since these correspond to hard logical facts that always have to be satisfied. The solution to (24) will then determine the design that best satisfies the possibly conflicting qualitative knowledge about the system.

Example 3

One of the problems most extensively studied in process synthesis is distillation column sequencing. Seader and Westerberg (1977) have identified rules which when followed, often lead to a good solution. Three of those rules will be used to demonstrate the concepts discussed above. The rules are:

1. Remove most plentiful component first.
2. Avoid difficult separations.
3. Separate into equal sized fractions.

Also, the following rule is imposed as a hard constraint:

4. Perform sharp splits.

Assume that, in rules 1–3, the penalities $w_1 = 2$, $w_2 = 2$, $w_3 = 1$, have been assigned to reflect the experience of a designer. The following logical expressions correspond to the heuristic rules for an N-component system (ABCD . . .) for which only sharp splits are considered:

1. "A" most plentiful (PLENTY_A) ⇒ Separate A and B (SPLIT_AT_AB)
   "B" most plentiful (PLENTY_B) ⇒ Separate A and B (SPLIT_AT_AB) OR
   Separate B and C (SPLIT_AT_BC)

. . .

2. AB Separation toughest(TOUGH_AB) ⇒ Don't separate A and B(—SPLIT_AT_AB)
   BC Separation toughest(TOUGH_BC) ⇒ Don't separate B and C(—SPLIT_AT_BC)

. . .

3. A forms half of total feed (HALF_A) ⇒ Separate A and B(SPLIT_AT_AB)
   A and B form half of total feed (HALF_AB) ⇒ Separate B and C (SPLIT_AT_BC)
4. EOR (SPLIT_AT_AB, SPLIT_AT_BC, . . .)

These rules can be converted into their conjunctive normal form and further translated into their mathematical representation also allowing for the violation of each of these rules. The problem of deciding on how to split a multicomponent mixture (ABC . . .) becomes an MILP of the form:

$$\begin{aligned} Z = \min\ & 2(v_{11} + v_{12} + \cdots) + 2(v_{21} + v_{22} + \cdots) + (v_{31} + v_{32} + \cdots), \\ \text{s.t.}\ & \\ & \text{PLENTY\_A} - \text{SPLIT\_AT\_AB} - v_{11} \leq 0, \\ & \text{PLENTY\_B} - \text{SPLIT\_AT\_AB} - \text{SPLIT\_AT\_BC} - v_{12} \leq 0, \\ & \cdots \\ & \text{TOUGH\_AB} + \text{SPLIT\_AT\_AB} - v_{21} \leq 1, \\ & \text{TOUGH\_BC} + \text{SPLIT\_AT\_BC} - v_{22} \leq 1, \\ & \cdots \\ & \text{HALF\_A} - \text{SPLIT\_AT\_AB} - v_{31} \leq 0, \\ & \text{HALF\_AB} - \text{SPLIT\_AT\_BC} - v_{32} \leq 0, \\ & \cdots \\ & \text{SPLIT\_AT\_AB} + \text{SPLIT\_AT\_BC} + \cdots = 1, \\ & \text{SPLIT\_AT\_AB}, \text{SPLIT\_AT\_BC}, \ldots \in \{0, 1\}^N, \\ & \qquad v_{ij} \geq 0 \quad \text{for all } i, j, \end{aligned} \tag{25}$$

where SPLIT_AT_AB, SPLIT_AT_BC, . . . are the design decisions and v_{ij} are the violations of the heuristics.

Table 2. Data for Example 4 (five-component system—ABCDE)

Component	Flow rate	Relative volatility
A	1	1.25
B	2	1.0
C	4	0.95
D	2	0.7
E	1	0.45

Table 3. Data for Example 4 (three-component system—ABC)

Component	Flow rate	Relative volatility
A	1	1.25
B	2	1.0
C	4	0.95

Given the data in Table 2 for the five-component mixture (ABCDE), the values of the variables PLENTY_ . . . , TOUGH_ . . . , HALF_ . . . can be established *a priori*. The resulting MILP involves four binary variables, 13 continuous variables and 13 constraints. The solution, which was obtained as a relaxed LP (0.22 s, SCICONIC/GAMS on VAX 6320 and 0.36 s, ZOOM/GAMS on Microvax II) yields SPLIT_AT_CD = 1 with no penalty ($Z = 0$). This means that no rules are violated when the split is performed between components C and D. Further, consider the resulting system ABC. Solving the problem with the data given in Table 3 yields SPLIT_AT_BC = 1. In this case, $Z = 2$ because there is a conflict between rules 1 and 2. Rules 1 and 3 override rule 2, so B and C are split although it is the most difficult separation because of the presence of relatively large amounts of B and C in the system.

The sequential application of the minimization of heuristics for this example would determine the separation sequence that is shown in Fig. 1. However, in general there is no guarantee that the sequence corresponds to the one that minimizes violations over all possible sequences. In order to address this problem, the formulation in (42) can easily be extended by considering the initial feed and all the intermediate mixtures that are shown in Table 4. In addition logical constraints are added to ensure a feasible separation sequence.

While considering the synthesis problem, it is necessary to define the variables as follows:

ASYSTEM(i, j) denotes the existence of the intermediate (i, j) given in Table 4,

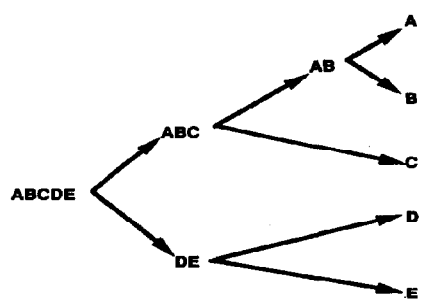

Fig. 1. Solution to Example 3.

Table 4. Intermediates (i, j) for the initial feed consisting of ABCDE

Row/column	1	2	3	4	5
1	ABCDE	ABCD	ABC	AB	A
2		BCDE	BCD	BC	B
3			CDE	CD	C
4				DE	D
5					E

SPLIT(i, j, k) denotes the existence of a split at position k (defined in Table 5) in intermediate (i, j),

PLENTY(i, j, k) existence of relatively large quantities of component k in intermediate (i, j),

TOUGH(i, j, k) existence of a difficult separation at position k in intermediate (i, j),

HALF(i, j, k) all components upto k form approximately half of the feed of intermediate (i, j).

The heuristic rules mentioned earlier would now be in the following form:

$$\begin{aligned}
&\text{ASYSTEM}(i,j) \wedge \text{PLENTY}(i,j,k) \\
&\quad \Rightarrow \text{CUT}(i,j,k-1) \vee \text{CUT}(i,j,k)\ \forall i,j,k, \\
&\text{ASYSTEM}(i,j) \wedge \text{TOUGH}(i,j,k) \\
&\quad \Rightarrow \neg\text{CUT}(i,j,k)\ \forall i,j,k, \\
&\text{ASYSTEM}(i,j) \wedge \text{HALF}(i,j,k) \\
&\quad \Rightarrow \text{CUT}(i,j,k)\ \forall i,j,k. \qquad (26)
\end{aligned}$$

The violations associated with each of the three rules is denoted by $v_1(i, j, k)$, $v_2(i, j, k)$, $v_3(i, j, k)$, resepectively. The weights associated with the three rules are the same as mentioned earlier. Equations (26) transforms to the following set of mathematical constraints:

$$\begin{aligned}
&-\text{ASYSTEM}(i,j) - \text{PLENTY}(i,j,k) + \text{CUT}(i,j,k) \\
&\quad + \text{CUT}(i,j,k-1) + v_1(i,j,k) \geq -1, \\
&-\text{ASYSTEM}(i,j) - \text{TOUGH}(i,j,k) - \text{CUT}(i,j,k) \\
&\quad + v_2(i,j,k) \geq -2, \\
&-\text{ASYSTEM}(i,j) - \text{HALF}(i,j,k) + \text{CUT}(i,j,k) \\
&\quad + v_3(i,j,k) \geq -1, \quad \forall i,j,k. \qquad (27)
\end{aligned}$$

Table 5. Definition of index k in Example 3

k	Split at
1	
2	AB
3	BC
4	CD
5	DE
...	...
N	between $(N-1)$th, Nth component

The hard logical constraints that are required to ensure a feasible separation sequence are the following:

1. If an intermediate exists, it must be split and at only one position. This is defined by the classification relationship:

$$\text{ASYSTEM}(i,j) = \{\text{CUT}(i,j,k);\ \forall k\}\ \forall i,j. \quad (28)$$

2. One split should be made at each position in order to separate all the components. This is expressed as:

$$\text{EOR}[\text{CUT}(i,j,k);\ \forall ij]\ \forall k. \quad (29)$$

3. If the intermediate (i,j) is split at position k, then the next two intermediates to exist are $(i, N+1-k+i)$ and $(k, j+k-i)$ where N is the total number of components in the initial feed. For example, consider the five-component system in this example where $N=5$. If the intermediate system ABCD exists $(i=1, j=2)$ and was split at position CD (i.e. the separation produced the intermediates ABC and D) for which $k=4$ then the new systems to exist would correspond to (1, 3) and (4, 5) which correspond to ABC and D, respectively, as can be verified from Table 4. This is expressed as:

$$\text{CUT}(i,j,k) \Rightarrow \text{ASYSTEM}(i, N+1-k+i) \wedge \text{ASYSTEM}(k, j+k-i)\ \forall i,j,k. \quad (30)$$

The hard logical constraints in (28–30) are transformed into their mathematical linear form which is:

$$\begin{aligned}
&\text{ASYSTEM}(i,j) - \sum_k \text{CUT}(i,j,k) = 0\ \forall i,j,\\
&\sum_i \sum_j \text{CUT}(i,j,k) = 1\ \forall k,\\
&\text{ASYSTEM}(i, N+1-k+i) - \text{CUT}(i,j,k) \geq 0\ \forall i,j,k,\\
&\text{ASYSTEM}(k, j+k-i) - \text{CUT}(i,j,k) \geq 0\ \forall i,j,k.
\end{aligned} \quad (31)$$

The objective function is:

$$2\sum_i \sum_j \sum_k v_1(i,j,k) + 2\sum_i \sum_j \sum_k v_2(i,j,k) + \sum_i \sum_j \sum_k v_3(i,j,k). \quad (32)$$

The MILP consists of minimizing the objective function in (32) under the constraints (27), (31) along with the additional requirements that:

$$\begin{aligned}
&v_1(i,j,k), v_2(i,j,k), v_3(i,j,k) \geq 0,\\
&\text{ASYSTEM}(i,j) \in \{0,1\}^{N^2},\\
&\text{CUT}(i,j,k) \in \{0,1\}^{N^3}.
\end{aligned} \quad (33)$$

The resulting MILP then involves 70 binary variables, 234 continuous variables and 275 constraints. Despite the fact that the MILP does not have the Horn claue structure, it was solved almost as a relaxed LP. The branch and bound method had to examine only two nodes requiring 1.35 s using the solver SCICONIC/GAMS (Brooke *et al.*, 1988) on a VAX 6320. Using ZOOM/GAMS on a Microvax II, 18 nodes were examined requiring 73.2 CPU s. The MILP also leads to the sequence shown in Fig. 1 which has an overall violation of $Z=2$. In this case, the sequential and simultaneous optimization approaches lead to the same sequence of separations. Note that if the penalty of the second heuristic involving difficult separations were much higher ($w_2 = 10$), then the sequential design would lead to the separation sequence shown in Fig. 2 with $Z=13$ while the simultaneous approach would still lead to the separation sequence of Fig. 1 which has a lower total penalty ($Z=10$). In general, the simultaneous approach will lead to a sequence that has a smaller weighted violation of heuristics than the sequential approach.

COMPARISON BETWEEN MILP APPROACH AND PRODUCTION SYSTEM APPROACH FOR QUALITATIVE KNOWLEDGE

Having presented in the previous section an MILP formulation for qualitative knowledge that includes heuristics, it is instructive to compare this approach with the use of production systems (e.g. expert systems) which have been extensively used in a number of different areas.

In order to preserve the modular nature of the inference procedure, production systems require that rules be written as Horn clauses. Horn clauses are disjunctions with not more than one nonnegated term (e.g. $A \vee \neg B \vee \neg C \vee \neg D$). In terms of production systems rules, a Horn clause is defined as a rule where there are no OR operators on the right-hand side of the implication ($B \vee C \vee D \Rightarrow A$). The reason for constraining the structure of rules to this form in production systems is that when the rule fires, it is possible to fix the value of the variable on the right side of the implication if the left side of the implication is true. This requirement of modularity is one of the main advantages of the production system in that it decomposes the solution of an inference problem. On the other hand, the drawback is that if

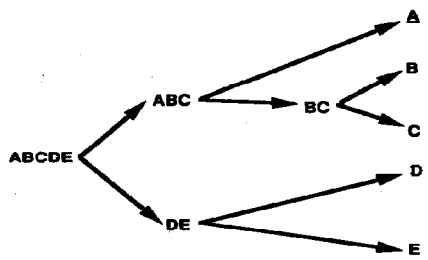

Fig. 2. Sequential solution to Example 3 with weights $w_1 = 2$, $w_2 = 10$, $w_3 = 1$.

the rule is not a Horn clause (e.g. $A \vee B \Rightarrow C \vee D$), the production system is not able to decide which of the variables on the right side of the implication to make true if the left side of the implication is true. The MILP approach, on the other hand, imposes no restrictions on the form of the logical expression.

Furthermore, with production systems, it is difficult to systematically model relationships like "Exclusive OR", "Classification" and "At most one" using propositional logic. Although it is simple for an expression with just two variables, it gets surprisingly complicated when the number of variables is increased. For example, A EOR B is equivalent to $(A \vee B) \wedge (\neg A \vee \neg B)$, while "at most one of A, B" is equivalent to $\neg A \vee \neg B$. Mathematically, they are very simple to express as the former is expressed as:

$$y_A + y_B = 1, \tag{34}$$

while the latter is expressed as:

$$y_A + y_B \leqslant 1 \tag{35}$$

and the complexity does not increase when the number of variables increases.

Although it is clear that the MILP approach offers more flexibility to represent the qualitative knowledge than a production system, the potential limitation is the computational efficiency. Both the logical inference and the MILP problems are classified as NP-complete which means that in the worst case, the solution time would grow exponentially with the problem size. In recent years though, large subclasses of logical inference problems have been identified as problems that can be solved as a *relaxed LP* in linear time. Chandrasekharan (1984) has shown how to round off the solution to the relaxed LP problem to obtain optimal integer solutions if the system is a Horn clause system. Furthermore, Chandru and Hooker (1989) have identified a structure termed hidden Horn clause system that can also be solved using Chandrasekharan's procedure in linear time. This implies that by just rewriting the logical expression as Horn clauses, one can greatly reduce the computational time. Trial runs by Hooker (1988) also seem to indicate that many of the logical inference problems can be solved as a relaxed LP even though they are not Horn clauses. Although at this point it is not clear whether an MILP-based approach could effectively replace the current search strategies used in production systems, the fact that these problems can often be solved as linear programs would suggest that this is an avenue worth exploring.

Furthermore, many search strategies used by expert systems can be shown to be specialized forms of MILP search. Resolution has been shown to be equivalent to a specialized first rank cutting plane (Hooker, 1988). Similarly backtracking search is simply a branch and bound search on the space of alternatives. So specialized search strategies depending on the structure of the model, that are performed in production systems, can also be performed on the MILP model.

Finally production systems evaluate rules sequentially using forward or backward chaining strategies (Lien *et al.*, 1987), which makes the ordering of rules a critical issue. Design engineers try to use this to their advantage by ordering rules in such a manner as to control the search. But in case of conflicts, this approach is not guaranteed to minimize the violation of heuristics. The MILP approach, on the other hand, can search the rules simultaneously so it does not restrict the formulation or ordering of rules (also see Example 3). The following example illustrates the comparison between the production system and the MILP approach.

Example 4

Consider the following three heuristic rules which are specified in decreasing order of priority, for a given system whose components are to be separated through distillation:

1. If the component is present in low concentration, then do not remove it first, although it is easy to separate and it can be removed at the top of the distillation column.
2. If the component is easy to separate, then it can be removed at the top and should be removed first.
3. If the component is the most volatile, then it is easy to separate.

   It will be assumed that the first rule having the highest priority has a penalty for violation $w_1 = 100$, the second rule has a penalty of violation $w_2 = 10$, while the last has $w_3 = 1$.

In order to express these rules through a logic formulation, the following literals are introduced:

P_1 = low concentration of component,

P_2 = most volatile component,

P_3 = can be removed at the top of the column,

P_4 = component is easy to separate,

P_5 = remove first.

The rules are then expressed as:

$$P_1 \wedge P_3 \wedge P_4 \Rightarrow \neg P_5\text{: V1},$$
$$P_4 \Rightarrow P_3 \wedge P_5\text{: V2},$$
$$P_2 \Rightarrow P_4\text{: V3}, \tag{36}$$

where V1, V2, V3 are the potential violations of the three rules. Note that the rules are not consistent with each other. The first and second rules predict contradictory values for P_5 with the same value of P_4.

Assume that one is trying to determine whether a given component can be removed given that this component is present in low concentrations (P_1 = True) and that it is the most volatile

(P_2 = True). If one were to apply forward chaining and assuming that the rules are processed in the order that is given, then the first rule that is fired is rule 3 yielding P_4 = True since P_2 = True is known. The next rule that would be fired is rule 2 yielding $P_3 = P_5$ = True since P_4 = True is known. Since at this stage, all the literals have been determined, the search would stop. Note, however, that rule 1, which is the one with highest priority, is actually violated.

Similarly, if one were to apply backward chaining, the inference engine would first search the right-hand side of all the rules until it reaches the first occurrence of P_5, which is rule 2. This rule requires that P_4 = True in order for P_5 = True. Then it searches the right-hand side of the rules until it reaches the first occurrence of P_4 which is rule 3. This rule requires that P_2 be determined in order to determine P_4. On further search for P_2, the inference engine discovers that P_2 is an input variable and that P_2 = True. So from rule 3, P_4 = True and therefore, from rule 2, P_5 = True. Note that, once again, rule 1 has not been fired and that the solution obtained violates rule 1.

In this way, forward or backward chaining on the system of rules would both yield the same solution, P_3 = True, P_4 = True, P_5 = True (i.e. remove the component first). In reaching this solution, the first and the most important of the rules is violated. In fact neither approach even fires the first rule.

Consider now the MILP approach. Associating the binary variables y_1, y_2, y_3, y_4, y_5 with literals P_1, P_2, P_3, P_4, P_5, respectively, and converting the logical expressions into their equivalent mathematical form, the inference problem yields the following MILP:

$$\begin{aligned} Z = \min\ & 100v_1 + 10(v_{2a} + v_{2b}) + v_3, \\ \text{s.t. } & y_1 + y_3 + y_4 + y_5 - v_1 \leqslant 3, \\ & y_4 - y_5 - v_{2a} \leqslant 0, \\ & y_4 - y_3 - v_{2b} \leqslant 0, \\ & y_2 - y_4 - v_3 \leqslant 0, \\ & y_1, y_2, y_3, y_4, y_5 \in \{0, 1\}, \\ & v_1, v_{2a}, v_{2b}, v_3 \geqslant 0. \end{aligned} \tag{37}$$

Note that rule 2 must be expressed through two inequalities and therefore each must be assigned its own violations, v_{2a}, v_{2b}.

Solving the MILP with the input $y_1 = 1, y_2 = 1$ (i.e. the component with low concentration and most volatile) yields, from the relaxed LP, the solution $\{y_3 = 0, y_4 = 0, y_5 = 0\}$ which is interpreted as "do not remove the component first". This solution only violated the third and the weakest of the three rules, and so is a better solution than the one obtained by simple forward and backward chaining approaches. The advantage of searching through the rules simultaneously, rather than sequentially should now be obvious.

It should be noted that one important assumption in the MILP modelling of qualitative knowledge that has been presented is that it does not involve any quantitative information. In synthesis applications, however, it is commonly the case that some qualitative decisions might be contingent on the calculated values of variables for the system to be synthesized. As an example, consider the case when the separation system in Example 4 is embedded as part of a flowsheet. Clearly, the relative amounts of the components present would depend on the operating conditions of the flowsheet.

As shown in Raman and Grossmann (1990), however, one can integrate the qualitative MILP model within a quantitative MINLP optimization model so as to explicitly model the dependency of certain qualitative rules on quantitative information. Since this problem is beyond the scope of this paper, it is discussed in detail in Raman and Grossmann (1990). Instead, the next section will consider as an additional application, the modelling of integer and mixed integer constraints.

MODELLING WITH INTEGER VARIABLES

As an additional application of the relationship between logical expressions and MILPs, consider the problem of formulating integer or mixed-integer programming problems for design and synthesis problems. Here constraints are commonly written intuitively by modellers. While this may work for simple cases, this approach can become difficult for more complex situations. It is therefore desirable to have a procedure where given the qualitative requirements for a mixed-integer formulation, the integer constraints can be formulated in a systematic manner. The approach described in the earlier sections of using propositional logic can be used for this purpose. The requirements are first expressed as propositional logic, then converted to their conjunctive normal form expression and finally translated to their corresponding linear mathematical form. The following examples will illustrate this point.

Example 5

Vaselenak *et al.* (1986) in their paper on heat integration in batch processes require a variable z_{ij}^t to equal 1 if a match exists between hot tank i and cold tank j in time t and 0 otherwise. There is also a restriction of only one match between a hot tank and a cold tank per time period. The existence of heat exchange by hot tank i in period t is denoted by $y_i^t = 1$ and that of heat exchange by cold tank j in period t is denoted by $y_j^t = 1$. Define the logical variables, Z_{ij}^t, Y_i^t and Y_j^t, associated with the binary variables z_{ij}^t, y_i^t and y_j^t, respectively, where a value of True for the logical variables correspond to a value of 1 for the binary variable.

The restriction of only one match each among all hot and cold tanks per time period is expressed as follows:

$$\text{EOR}(Y_i^t; i = 1, 2, \ldots, N\text{HOT}) \quad t = 1, 2, 3, \ldots, T,$$
$$\text{EOR}(Y_j^t; j = 1, 2, \ldots, N\text{COLD}) \quad t = 1, 2, 3, \ldots, T. \tag{38}$$

The relation betwen Z_{ij}^t and Y_i^t, and Y_j^t is expressed as follows:

$$Y_i^t \wedge Y_j^t \Rightarrow Z_{ij}^t. \tag{39}$$

In the conjunctive normal form:

$$\neg Y_i^t \vee \neg Y_j^t \vee Z_{ij}^t. \tag{40}$$

Hence, in the mathematical linear form, the three expressions in (38) and (39) become:

$$\sum_{i=1}^{N\text{HOT}} y_i^t = 1 \quad t = 1, 2, \ldots, T,$$
$$\sum_{j=1}^{N\text{COLD}} y_j^t = 1 \quad t = 1, 2, \ldots, T,$$
$$y_i^t + y_j^t - z_{ij}^t \leqslant 1 \quad \forall i, j, t. \tag{41}$$

Example 6

Consider the problem of deriving an integer cut that will prevent a certain integer point from being feasible. These integer cuts are used to generate second, third best solutions, or as part of the MILP master problem for an MINLP algorithm (see Kocis and Grossmann, 1989b). Suppose the integer point in question is:

$$y = \{y_i, i \in B; y_j, j \in N\}, \tag{42}$$

where y_i, $i \in B$ are the set of all binary variables that have a value of 1, and y_j, $j \in N$ are the set of all variables that have a value of 0. Since the purpose of the integer cut is to prevent the given combination values for y_i and y_j, the integer cut can logically be expressed as:

$$\neg[(\underset{i \in B}{\wedge} Y_i) \wedge (\underset{j \in N}{\wedge} \neg Y_j)]. \tag{43}$$

Application of DeMorgan's Theorem yields:

$$(\underset{i \in B}{\vee} \neg Y_i) \vee (\underset{j \in N}{\vee} Y_j). \tag{44}$$

Translating the above into the mathematical linear form:

$$\sum_{i \in B} (1 - y_i) + \sum_{j \in N} y_j \geqslant 1, \tag{45}$$

which finally yields the inequality (Balas and Jeroslow, 1972):

$$\sum_{i \in B} y_i - \sum_{j \in N} y_j \leqslant |B| - 1, \tag{46}$$

where $|B|$ is the cardinality of set B.

MODELLING WITH INTEGER AND CONTINUOUS VARIABLES

Another type of constraints that have been difficult to formulate systematically are the class of constraints wherein the satisfaction of a set of constraints implies that another set of constraints is also valid [e.g. "$f(x) \leqslant 0$" implies "$g(x) \geqslant 0$"]. These types of constraints arise when modelling the superstructure of a flowsheet (see Kocis and Grossmann, 1989a). One can associate a binary variable with each of the inequalities and equalities involved and then the relation between these is expressed in terms of propositional logic. The relationships between the various kinds of inequalities and equalities with the associated binary variables is handled as follows:

1. Associate a binary variable y_1 with the inequality $f(x) \leqslant 0$. Then the relationship between y_1 and the inequality is:
$$L_1 y_1 + \epsilon \leqslant f(x) \leqslant U_1(1 - y_1) \tag{47}$$
where L_1, U_1 are the upper and lower bounds for $f(x)$ in the problem considered and ϵ is a small positive tolerance. So if $f(x) \leqslant 0$, then $y_1 = 1$ in order to satisfy the inequality. Conversely, if $f(x) \geqslant \epsilon$, then $y_1 = 0$.
2. Associate a binary variable y_2 with the inequality $g(x) \geqslant 0$. Then the relationship between y_2 and the inequality is:
$$L_2(1 - y_2) \leqslant g(x) \leqslant U_2 y_2 - \epsilon, \tag{48}$$
where L_2, U_2 are the lower and upper bounds on the value of $g(x)$ for the problem considered and ϵ is a small small positive tolerance. So if $g(x) \geqslant 0$, then $y_2 = 1$ in order to satisfy the above inequality.
3. Treat $h(x) = 0$ as "$h(x) \leqslant 0$ AND $h(x) \geqslant 0$". Associate the binary variable z_1 with $h(x) \leqslant 0$, the binary variable z_2 with $h(x) \geqslant 0$ and the binary variable y_3 with $h(x) = 0$.

The relationship between the inequalities and the binary variables associated with them follows from equations (47) and (48):

$$L_3 z_1 + \epsilon \leqslant h(x) \leqslant U_3(1 - z_1), \tag{49}$$
$$L_3(1 - z_2) \leqslant h(x) \leqslant U_3 z_2 - \epsilon. \tag{50}$$

Also, from the definition of the binary variables:

$$Y_3 \Leftrightarrow Z_1 \wedge Z_2. \tag{51}$$

This is equivalent to the following three inequalities:

$$z_1 + z_2 - y_3 \leqslant 1, \tag{52}$$
$$z_1 - y_3 \geqslant 0, \tag{53}$$
$$z_2 - y_3 \geqslant 0, \tag{54}$$

where L_3, U_3 are the lower and upper bounds on the value of $h(x)$ and ϵ is a small positive tolerance.

Equations (49), (50) and (52–54) relate the binary variable y_3 with the equality "$h(x) = 0$". Note that when $y_3 = 1$, then $h(x) = 0$.

To illustrate the modelling procedure for a specific relationship, consider the following example.

Example 7

Consider the logical condition, "If $f(x) \leqslant 0$ and $h(x) = 0$, then $g(x) \geqslant 0$". Associate binary variables y_1, y_2, y_3 with the expressions $f(x) \leqslant 0$, $g(x) \geqslant 0$ and $h(x) = 0$, respectively. Then the expression can be written as:

$$Y_1 \wedge Y_3 \Rightarrow Y_2. \tag{55}$$

In the linear mathematical form:

$$y_1 + y_3 - y_2 \leqslant 1. \tag{56}$$

Also, from equations (47–50) and (52–54), the following inequalities relate the algebraic equality and inequalities with their associated binary variable:

$$\begin{aligned} L_1 y_1 + \epsilon &\leqslant f(x) \leqslant U_1(1 - y_1), \\ L_2(1 - y_2) &\leqslant g(x) \leqslant U_2 y_2 - \epsilon, \\ L_3 z_1 + \epsilon &\leqslant h(x) \leqslant U_3(1 - z_1), \\ L_3(1 - z_2) &\leqslant h(x) \leqslant U_3 z_2 - \epsilon, \\ z_1 + z_2 - z_3 &\leqslant 1, \\ z_1 - y_3 &\geqslant 0, \\ z_2 - y_3 &\geqslant 0. \end{aligned} \tag{57}$$

Thus the inequalities in (56) and (57) define the logical relationship for the inequalities and equations.

Example 8

Consider as a final example, the modelling of nondifferentiable functions using integer variables. This can be done with the ideas developed so far. Associate a binary variable with each discontinuity or point of nondifferentiability and model the inequalities and inequalities on either side of the point considered using the approach described in the previous example.

The idea is illustrated with the following example on the modelling of the $\max\{0, f(x)\}$ function that arises in the heat integration model by Duran and Grossmann (1986):

$$\phi = \max[0, f(x)]. \tag{58}$$

The function is nondifferentiable at $f(x) = 0$ since:

$$\phi = \begin{cases} f(x): & f(x) \geqslant 0, \\ 0: & f(x) < 0. \end{cases} \tag{59}$$

Associate the binary variable y_1 with $f(x) \geqslant 0$ and $(1 - y_1)$ with $f(x) \leqslant 0$ so that:

$$\begin{aligned} f(x) - M y_1 &\leqslant 0, \\ f(x) + L(1 - y_1) &\geqslant 0. \end{aligned} \tag{60}$$

Furthermore, we wish to impose the conditions $y_1 = 1 \Rightarrow \phi = f(x)$ and $y_1 = 0 \Rightarrow \phi = 0$. This can be accomplished with the inequalities:

$$\begin{aligned} L_1(1 - y_1) \leqslant \phi - f(x) \leqslant U_1(1 - y_1), \\ L_2 y_1 \leqslant \phi \leqslant U_2 y_1, \end{aligned} \tag{61}$$

where L_1, U_1 are the lower and upper bounds on $\phi - f(x)$, respectively, while L_2, U_2 are similar bounds on ϕ. Since $L_1 = L_2 = 0$, (60) and (61) can be simplified as

$$\begin{aligned} 0 \leqslant \phi - f(x) \leqslant U_1(1 - y_1) \\ 0 \leqslant \phi \leqslant U_2 y_1. \end{aligned} \tag{62}$$

Note that if $y_1 = 1$, $\phi = f(x) \geqslant 0$, and if $y_1 = 0$, $\phi = 0$. In this way (62) represents the max function in (58).

CONCLUSION

In this paper, an MILP approach has been presented for solving problems in process synthesis which require reasoning to make inferences from qualitative knowledge. The work done in this area has been reviewed and special emphasis placed on showing its relevance to process synthesis problems. These ideas have been illustrated with several examples, including the synthesis of a separation system. In this example, it was shown that the MILP approach provides a framework for simultaneously minimizing the weighted violation of heuristics.

A comparison between the production systems approach and the MILP approach has also been made with regard to representation of knowledge and control strategies. A simple example was presented to illustrate the fact that forward or backward chaining search schemes may lead to solutions that yield larger violation of heuristics when compared to the MILP approach. Although no firm conclusions can be drawn from the computational expense for solving the MILP for general logic structures, systems with Horn clauses and extended Horn clause systems can be solved in linear time, as has been proved by Chandrasekharan (1984) and Hooker (1988).

Finally, the advantage of using propositional logic to systematically model constraints for integer and mixed integer programming models in process synthesis has been shown. This concept has also been applied to handle the modelling of discontinuous and nondifferentiable functions using integer variables. The approach presented in this paper should provide a first step in integrating qualitative and quantitative knowledge effectively.

Acknowledgements—The authors would like to acknowledge financial support from the Engineering Design Research Center at Carnegie Mellon University and from the Tennessee Eastman Company.

REFERENCES

Balas E. and R. Jeroslow, Canonical cuts on the unit hypercube. *SIAM J. Appl. Math.* **23**, 61–79 (1972).

Beltramini L. J., C. M. Sheppard and R. L. Motard, Truth maintenance systems in process synthesis. Paper 31c, *National AIChE Meeting*, Houston (1989).

Brooke A., D. Kendrick and A. Meeraus, *GAMS—A Users' Guide*. Scientific Press, Palo Alto (1988).

Cavalier T. M. and A. L. Soyster, Logical deduction via linear programming. IMSE Working Paper 87-147, Department of Industrial and Management Systems Engineering, Pennsylvania State University (1987).

Chandrasekharan R., Integer programming problems for which a simple rounding type of algorithm works. *Progress in Combinatorial Optimization* (W. R. Pulleybank, Ed.), pp. 101–106. Academic Press, Canada (1984).

Chandru V. and J. N. Hooker, Extended Horn sets in propositional logic. Working Paper 88-89-39, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh (1989).

Clocksin W. F. and C. S. Mellish, *Programming in Prolog*. Springer-Verlag, New York (1981).

Douglas J. M., A hierarchical decision procedure of process synthesis. *AIChE Jl* **31**, 353–362 (1985).

Duran M. A. and I. E. Grossmann, Simultaneous optimization and heat integration of chemical processes. *AIChE Jl* **32**, 123–138 (1986).

Grossmann I. E., MINLP optimization strategies and algorithms for process synthesis. *FOCAPD Meeting Proceedings* (J. J. Siirola *et al.*, Eds), pp. 105–132. Elsevier, Amsterdam (1989).

Hooker J. N., Resolution vs cutting plane solution of inference problems: some computational experience. *Ops Res. Letts* **7**, 1 (1988).

Kocis G. R. and I. E. Grossmann, A modelling/decomposition strategy for MINLP optimization of process flowsheets. *Computers chem. Engng* **13**, 797–819 (1989a).

Kocis G. R. and I. E. Grossmann, Computational experience with DICOPT solving MINLP problems in process synthesis engineering. *Computers chem. Engng* **13**, 307–315 (1989b).

Lien K., G. Suzuki and A. W. Westerberg, The role of expert systems technology in design. *Chem. Engng Sci.* **42**, 1049 (1987).

Mahalec V. and R. L. Motard, Procedures for the initial design of chemical processing systems. *Computers chem. Engng* **1**, 57–68 (1977).

Papoulias S. A. and I. E. Grossmann, A structural optimization approach in process synthesis. Parts I, II and III. *Computers chem. Engng* **7**, 695–734 (1983).

Post S., Reasoning with incomplete and uncertain knowledge as an integer linear program. *Proc. Avignon 87: Expert Systems and their Applications*. Avignon, France (1987).

Raman R. and I. E. Grossmann, Integration of qualitative knowledge in MINLP optimization for process synthesis. Paper 26f, *A. AIChE Meeting*, Chicago (1990).

Seader J. D. and A. W. Westerberg, A combined heuristic and evolutionary strategy for synthesis of simple separation sequences. *AIChE Jl* **23**, 951 (1977).

Siirola J. J., G. J. Powers and D. F. Rudd, Synthesis of systems design III. Towards a process concept generator. *Ind. Engng Chem. Fundam.* **17**, 677 (1971).

Stephanapoulos G., J. Johnston, T. Kriticos, R. Lakshmanan, M. Mavrovouniotis and C. Siletti, DESIGN-KIT: an object-oriented environment for process engineering. *Computers chem. Engng* **11**, 655–674 (1987).

Umeda T., A. Hirai and A. Ichikawa, Synthesis of optimal processing systems by an integrated approach. *Chem. Engng Sci.* **27**, 795–804 (1972).

Vaselenak J. A., I. E. Grossmann and A. W. Westerberg, Heat integration in batch processes. *Ind. Engng Chem. Process Des. Dev.* **25**, 357 (1986).

Williams H. P., *Model Building in Mathematical Programming*. Wiley, Chichester (1988).