sitional logic can be used to help us to model these constraints. In the next chapters, we will actually make use of LP, MILP, NLP, and MINLP formulations for modeling synthesis problems. However, we will keep in mind the above guidelines when developing these models.

## 15.7 MODELING OF LOGIC CONSTRAINTS AND LOGIC INFERENCE

Because a large part of the next chapters will deal with the development of mixed-integer optimization methods, we will present in this section a framework that should be helpful for deriving constraints involving 0–1 variables. Some of these constraints are quite straightforward, but some are not. For instance, specifying that exactly only one reactor be selected among a set of candidate reactors $i \in R$ is simply expressed as,

$$\sum_{i \in R} y_i = 1 \tag{15.14}$$

On the other hand, consider representing the constraint: "if the absorber to recover the product is selected or the membrane separator is selected, then do not use cryogenic separation". We could by intuition and trial and error arrive at the following constraint,

$$y_A + y_M + 2y_{CS} \leq 2 \tag{15.15}$$

where $y_A$, $y_M$, and $y_{CS}$ represent 0–1 variables for selecting the corresponding units (absorber, membrane, cryogenic separation). Note that if $y_A = 1$ and/or $y_M = 1$ (Eq. 15.15) forces $y_{CS} = 0$. We will see, however, that we can systematically arrive at the alternative constraints,

$$\begin{aligned} y_A + y_{CS} &\leq 1 \\ y_M + y_{CS} &\leq 1 \end{aligned} \tag{15.16}$$

which are not only equivalent to Eq. (15.15) but also more efficient in the sense that they are "tighter" because they constrain more the feasible region (see exercise 7).

In order to systematically derive constraints involving 0–1 variables, it is useful to first think of the corresponding propositional logic expression that we are trying to model as described in Raman and Grossmann (1991). For this we first must 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; Williams, 1985).

To each literal $P_i$ that represents a selection or action, 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 15.1. From this table, it is easy to verify, for instance, that the logical proposition in $y_1 \vee y_2$ reduces to the inequality in Eq. (15.13).

**TABLE 15.1 Constraint Representation of Logic Propositions and Operators**

| Logical Relation | Comments | Boolean Expression | Representation as Linear Inequalities |
|---|---|---|---|
| Logical OR | | $P_1 \vee P_2 \vee .. \vee P_r$ | $y_1 + y_2 + .. + y_r \geq 1$ |
| Logical AND | | $P_1 \wedge P_2 \wedge .. \wedge P_r$ | $y_1 \geq 1$<br>$y_2 \geq 1$<br>..<br>$y_r \geq 1$ |
| Implication | $P_1 \Rightarrow P_2$ | $\neg P_1 \vee P_2$ | $1 - y_1 + y_2 \geq 1$ |
| Equivalence | $P_1$ if and only if $P_2$<br>$(P_1 \Rightarrow P_2) \wedge (P_2 \Rightarrow P_1)$ | $(\neg P_1 \vee P_2) \wedge (\neg P_2 \vee P_1)$ | $y_1 = y_2$ |
| Exclusive OR | Exactly one of the variables is true | $P_1 \veebar P_2 \veebar .. \veebar P_r$ | $y_1 + y_2 + .. + y_r = 1$ |

With the basic equivalent relations given in Table 15.1 (e.g., see Williams, 1985), 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 (Raman and Grossmann, 1991). The conjunctive normal form is a conjunction of clauses, $Q_1 \wedge Q_2 \wedge ... \wedge Q_s$ (i.e., connected by AND operators $\wedge$). 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, $P_1 \vee P_2 \vee ... \vee P_r$ (i.e., connected by OR operators $\vee$), it can be expressed in the linear mathematical form as the inequality.

$$y_1 + y_2 + ..... + y_r \geq 1 \tag{15.17}$$

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,

$$P_1 \Rightarrow P_2 \quad \Leftrightarrow \quad \neg P_1 \vee P_2 \tag{15.18}$$

2. Move the negation inward by applying DeMorgan's Theorem:

$$\neg (P_1 \wedge P_2) \quad \Leftrightarrow \quad \neg P_1 \vee \neg P_2 \tag{15.19}$$

$$\neg (P_1 \vee P_2) \quad \Leftrightarrow \quad \neg P_1 \wedge \neg P_2 \tag{15.20}$$

3. Recursively distribute the "OR" over the "AND", by using the following equivalence:

$$(P_1 \wedge P_2) \vee P_3 \quad \Leftrightarrow \quad (P_1 \vee P_3) \wedge (P_2 \vee P_3) \tag{15.21}$$

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

The following two examples illustrate the procedure for converting logical expressions into inequalities.

**EXAMPLE 15.4**

Consider the logic condition we gave above "if the absorber to recover the product is selected or the membrane separator is selected, then do not use cryogenic separation". Assigning the boolean literals to each action $P_A$ = select absorber, $P_M$ = select membrane separator, $P_{CS}$ = select cryogenic separation, the logic expression is given by:

$$P_A \vee P_M \Rightarrow \neg P_{CS} \tag{15.22}$$

Removing the implication, as in (15.18), yields,

$$\neg (P_A \vee P_M) \vee \neg P_{CS} \tag{15.23}$$

Applying De Morgan's Theorem, as in Eq. (15.20), leads to,

$$(\neg P_A \wedge \neg P_M) \vee \neg P_{CS} \tag{15.24}$$

Distributing the OR over the AND gives,

$$(\neg P_A \vee \neg P_{CS}) \wedge (\neg P_M \vee \neg P_{CS}) \tag{15.25}$$

Assigning the corresponding 0–1 variables to each term in the above conjunction, and using Eq. (15.17),

$$\begin{aligned} 1 - y_A + 1 - y_{CS} &\geq 1 \\ 1 - y_M + 1 - y_{CS} &\geq 1 \end{aligned} \tag{15.26}$$

which can be rearranged to the two inequalities in Eq. (15.16),

$$\begin{aligned} y_A + y_{CS} &\leq 1 \\ y_M + y_{CS} &\leq 1 \end{aligned} \tag{15.27}$$

**EXAMPLE 15.5**

Consider the proposition

$$(P_1 \wedge P_2) \vee P_3 \Rightarrow (P_4 \vee P_5) \tag{15.28}$$

By removing the implication, the above proposition yields from Eq. (15.18),

$$\neg [ (P_1 \wedge P_2) \vee P_3 ] \vee P_4 \vee P_5 \tag{15.29}$$

Further, from Eqs. (15.19) and (15.20), moving the negation inwards leads to the following two steps,

$$[\neg (P_1 \wedge P_2) \wedge \neg P_3] \vee P_4 \vee P_5 \tag{15.30}$$

$$[(\neg P_1 \vee \neg P_2) \wedge \neg P_3] \vee P_4 \vee P_5 \tag{15.31}$$

Recursively distributing the "OR" over the "AND" as in Eq. (15.21) the expression becomes

$$(\neg P_1 \vee \neg P_2 \vee P_4 \vee P_5) \wedge (\neg P_3 \vee P_4 \vee P_5) \tag{15.32}$$

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,

$$\begin{aligned} y_1 + y_2 - y_4 - y_5 &\le 1 \\ y_3 \quad\; - y_4 - y_5 &\le 0 \end{aligned} \tag{15.33}$$

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 that make 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.

It should be noted that the one exception where applying the above procedure becomes cumbersome is when dealing with constraints that limit choices, for example, select no more than one reactor. In that case it is easier to directly write the constraint and not go through the above formalism.

As an application of the material above, let us consider logic inference problems in which given the validity of a set of propositions, we have to prove the truth or the validity of a conclusion that may be either a literal or a proposition. The logic inference problem can be expressed as:

$$\begin{aligned} &\text{Prove} \quad Q_u \\ &st \qquad B(Q_1, Q_2 ... Q_s) \end{aligned} \tag{15.34}$$

where $Q_u$ is the clause or proposition expressing the conclusion to be proved and $B$ is the set of clauses $Q_i$, $i = 1,2,..,s$.

Given that all the logical propositions have been converted to a set of linear inequalities, the inference problem in Eq. (15.34) can be formulated as the following MILP (Cavalier and Soyster, 1987):

$$\begin{aligned} &\text{Min} \quad Z = \sum_{i \in I(u)} c_i y_i \\ &st \qquad A\ y \ge a \\ &y \in \{0,1\}^n \end{aligned} \tag{15.35}$$

where $A\ y \ge a$ is the set of inequalities obtained by translating $B\ (Q_1, Q_2, .., Q_s)$ into their linear mathematical form, and the objective function is obtained by also converting the clause $Q_u$ 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 $Q_u$. 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 (15.35) 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 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 Eq. (15.35), one might be able to make inferences. The following example will illustrate a simple application in process synthesis.

**EXAMPLE 15.6**

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 clauses in 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.

The chemical reactions are given by

$$\begin{aligned} H_2O + CO_2 &\longrightarrow H_2CO_3 \\ C + O_2 &\longrightarrow CO_2 \end{aligned} \tag{15.36}$$

assuming that $H_2O$, C, and $O_2$ are available. Expressing the reactions in logical form yields

$$\begin{aligned} H_2O \wedge CO_2 &\Rightarrow H_2CO_3 \\ C \wedge O_2 &\Rightarrow CO_2 \end{aligned} \tag{15.37}$$

The objective is to prove whether $H_2CO_3$ can be formed given that $H_2O$, C, and $O_2$ 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 Eq. (15.35) becomes the following MILP problem,

$$\begin{array}{lll} Z = \text{Min} & y_{H2CO3} & \\ st & y_{H2O} + y_{CO2} - y_{H2CO3} & \leq 1 \\ & y_C + y_{O2} - y_{CO2} & \leq 1 \\ & y_{H2O} & = 1 \\ & y_C & = 1 \\ & y_{O2} & = 1 \\ & y_C, y_{O2}, y_{CO2}, y_{H2O}, y_{H2CO3} \in \{0,1\} & \end{array} \tag{15.38}$$

The objective involves the minimization of $y_{H2CO3}$ because the objective is to prove whether $H_2CO_3$ can be found. Solving the relaxed LP problem yields an integer solution with $Z = 1$ and $y_{H2CO3} = y_{CO2} = 1$. This solution is then interpreted as "H2CO3 can always be produced from $H_2O$ , C, and $O_2$ given the above reactions".

Finally, it should be noted that the MILP in Eq. (15.35) can easily be extended for handling heuristic rules that may be violated (Raman and Grossmann, 1991). To model the potential violation of heuristics, the following logic relation is considerded,

$$\text{Clause OR } v \tag{15.39}$$

where either the clause is true or it is being violated ($v$). In order to discriminate between weak and strong rules, penalties are associated with the violation $v_i$ of each heuristic rule, $i = 1,..,m$. The penalty $w_i$ is a non-negative number that reflects the uncertainty of the corresponding logical expression. The more uncertain the rule, the lower the penalty for its violation. 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:

$$\begin{array}{lll} \text{Min} & Z = w^T v & \\ st & A\,y \quad \geq a & : \text{ Logical facts} \\ & B\,y + v \geq b & : \text{ Heuristics} \\ & y \in \{0,1\}^n,\ v \geq 0 & \end{array} \tag{15.40}$$

Note that no violations are assigned to the inequalities $Ay \geq a$ since these correspond to hard logical facts that always have to be satisfied. In this way Eq. (15.40) can be used to solve inference problems involving logic relations and heuristics. Clearly, if the solution is $Z = 0$, it means that it is possible to find a solution without violating heuristics. In general, the solution to Eq. (15.40) will determine a design that best satisfies the possibly conflicting qualitative knowledge about the system.

## 15.8 MODELING OF DISJUNCTIONS

In the previous section we presented a systematic framework based on logic for modeling constraints involving 0–1 variables. In a number of cases, however, we will have to deal with logic constraints that involve continuous variables. A good example is the following condition when selecting among two reactors:

*If select reactor 1, then pressure P must lie between 5 and 10 atmospheres.*
*If select reactor 2, then pressure P must lie between 20 and 30 atmospheres.*

To represent logic with continuous variables we will consider linear disjunctions of the form:

$$\vee_{i \in D}[A_i x \leq b_i] \tag{15.41}$$

where $\vee$ is the OR operator that applies to a set of disjunctive terms D. In the above example, Eq. (15.41) reduces to:

$$\begin{bmatrix} P \leq 10 \\ -P \leq -5 \end{bmatrix} \vee \begin{bmatrix} P \leq 30 \\ -P \leq -20 \end{bmatrix} \tag{15.42}$$

where the first term is associated to reactor 1 ($y_1$) and the second term to reactor 2 ($y_2$).

The simplest way to convert Eq. (15.41) into mixed-integer constraints is by using "big-M" constraints, which are given as follows:

$$\begin{aligned} & A_i x \leq b_i + M_i(1 - y_i) \quad i \in D \\ & \sum_{i \in D} y_i = 1 \\ & y_i = 0,1 \quad i \in D \end{aligned} \tag{15.43}$$

Note that the 0–1 variable $y_i$ is introduced to denote which disjunction $i$ in $D$ is true ($y_i = 1$). The second constraint in Eq. (15.43) only allows one choice of $y_i$. The first set of inequalities, $i \in D$, introduce on the right-hand side a big parameter $M_i$, which renders the inequality redundant if $y_i = 0$. Note that if $y_i = 1$, the inequality is enforced.

As applied to Eq. (15.42) the big-$M$ constraints yield:

$$\begin{aligned} & P \leq 10 + M_1\,(1 - y_1) \\ & -P \leq -5 + M_1\,(1 - y_1) \\ & P \leq 30 + M_2\,(1 - y_2) \\ & -P \leq 20 + M_2\,(1 - y_2) \\ & y_1 + y_1 = 1 \end{aligned} \tag{15.44}$$

Large values, such as $M_1 = 100$, $M_2 = 100$, are valid choices but produce weak "relaxations" or bounds for the objective function when the $y$'s are treated as continuous variables. This would be, for instance, the first step in the LP branch and bound method.

An alternative for avoiding the use of big-$M$ parameters in Eq. (15.43) is the use of the convex hull formulation, which requires disaggregating continuous variables. As shown in Balas (1985) and discussed in Turkay and Grossmann (1996), the convex hull model of Eq. (15.41) is given by:

$$\begin{aligned} & x = \sum_{i \in D} z_i \\ & A_i\, z_i \leq b_i\, y_i \qquad i \in D \\ & \sum_{i \in D} y_i = 1 \\ & 0 \leq z_i \leq U y_i \qquad i \in D \\ & y_i = 0,1 \end{aligned} \tag{15.45}$$

In the above $z_i$ are continuous variables disaggregated into as many new variables as there are terms for the disjunctions. The first equation simply equates the original variable $x$ to the disaggregated variables $z_i$. The second constraint corresponds to inequalities written in terms of the disaggregated variables $z_i$ and a 0–1 variable $y_i$. The third simply states that only one $y_i$ can be set to one. The fourth constraint is optional in that it is only included if $y_i = 0$ in the second inequality does not imply $z_i = 0$. The importance of the constraints in Eq. (15.45) is that they do not require the introduction of the big-$M$ parameter yielding a tight LP relaxation. The disadvantage is that it requires a larger number of variables and constraints.

Applied to Eq. (15.42), Eq. (15.45) yields,

$$\begin{gathered} P = P_1 + P_2 \\ P_1 \le 10\, y_1 \qquad P_2 \le 30\, y_2 \\ -P_1 \le -5\, y_1 \qquad -P_2 \le -20\, y_2 \\ y_1 + y_2 = 1 \end{gathered} \tag{15.46}$$

It is important to note that often the convex hull formulation will simplify if there are only two terms in the disjunction and one requires the variable to take a value at zero. For instance, consider a flow $F \ge 0$ for which

$$[F \le 20] \vee [F = 0] \tag{15.47}$$

It can easily be shown that applying Eq. (15.45) to Eq. (15.47), since $F_2 = 0.y_2$, $F = F_1$, and hence the convex hull at Eq. (15.47) is given by

$$F \le 20\, y_1 \tag{15.48}$$

In practice, the big-$M$ constraints as in Eq. (15.43) are easiest to use and will not cause major difficulties if the problem is small. For larger problems the convex hull formulation is often the superior one.

## 15.9 NOTES AND FURTHER READING

A recent review on optimization approaches to process synthesis can be found in Grossmann and Daichendt (1996). Modeling is largely an art that has a large impact in mixed-integer programming. Good practices can be learned from examples. The book by Williams (1985) is perhaps the most useful. Similarly, the book by Schrage (1984) has a good number of examples for LP and MILP problems. Nemhauser and Wolsey (1988) also present some interesting examples. Finally, the papers by Raman and Grossmann (1991, 1994) provide logic-based formalisms for the modeling of the 0–1 and disjunctive constraints.

## REFERENCES

Andrecovich, M. J., & Westerberg, A. W. (1985). MILP formulation for heat-integrated distillation sequence synthesis. *AIChE J.*, 31, 1461.