More, J. J.; Global Methods for Nonlinear Complementarity Problems. Technical report, Argonne National Laboratory, Mathematics and Computer Science Division, April 1994.

Pantelides, C. C.; SPEEDUP-Recent Advances in Process Simulation. *Comput. Chem. Eng.*, 12(7):745–755, 1988.

Pantelides, C. C. and Barton, P. I.; Equation-Oriented Dynamic Simulation: Current Status and Future Perspectives. *Comput. Chem. Eng.*, 17S:263–285, 1993.

Piela, P., Epperly, T., Westerberg, K., Westerberg, A. W.; An Object-Oriented Computer Environment for Modeling and Analysis: The modeling language. *Comput. Chem. Eng.*, 15(1):53–72, 1991.

Piela, P.; ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis. PhD thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, April 1989.

Quesada, I. and Grossmann I. E.; A Global Optimization Algorithm for Linear Fractional and Bilinear Programs. *Journal of Global Optimization*, 6:39–76, 1995.

Raman, R. and Grossmann, I. E.; Symbolic Integration of Logic in Mixed-Integer Linear Programming Techniques for Process Synthesis. *Comput. Chem. Eng.*, 17(9):909–927, 1993.

Raman, R. and Grossmann, I. E.; Modeling and Computational Techniques for Logic Based Integer Programming. *Comput. Chem. Eng.*, 18(7):563–578, 1994.

Turkay, M. and Grossmann, I. E.; Logic-Based MINLP Algorithms for the Optimal Synthesis of Process Networks. Comput. Chem. Eng., 20(8):959–978, 1996.

Viswanathan, J. and Grossmann, I. E.; A Combined Penalty Function and Outer Approximation Method for MINLP Optimization. *Comput. Chem. Eng.*, 14(7):769–782, 1990.

Westerberg, A.W., Abbott, K. A., and Allan, B. A.; Plans for ASCEND IV: Our Next Generation Equational-Based Modeling Environment. Boston, Massachusetts, November 1994. AspenWorld'94.

Westerberg, A. W. and Director, S. W.; A Modified Least Square Algorithm for Solving Sparse n x n Sets of Nonlinear Equations. Technical report, Carnegie Mellon University, Engineering Design Research Center, DRC-06-5-79, January 1979.

Zaher, J. J.; Conditional Modeling. PhD thesis, Department of Chemical Engineering,Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1995.

Zaher, J. J.; Conditional Modeling in an Equation-Based Modeling Environment. The Annual AIChE National Meeting, 1991. paper 138c.

Zaher, J. J.; Conditional Programming. The Annual AIChE National Meeting, March 1993.

of this project. Some of the weaknesses and advantages of each of these approaches have been mentioned, but overcoming some of the difficulties and testing the approaches in large scale problems are issues still ahead as part of the future work. The following remain as some of the questions yet to be answered: *i)* How can the information provided by the structure of the problem help to simplifying the degrees of freedom analysis of a large conditional model? *ii)* How is the performance of a complementarity formulation against that of the boundary crossing algorithm in the solution of conditional models of practical size? *iii)* How is their performance in an optimization scheme? How does one overcome the difficulties incorporated by the bilinear terms in the complementarity formulation? Under what conditions can these approaches be considered a reliable alternative to MINLP algorithms in the simulation and optimization of large scale models?.

7 REFERENCES

Allen, G. L. and Westerberg, A. W.; Solution Procedures for Indexed Equation Sets: Structural Considerations. *AIChE Journal*, 22(3):549–558, May 1976.

Barton, P. I.; The Modeling and Simulation of Combined Discrete/Continuous Processes. PhD thesis, Department of Chemical Engineering, Imperial College of Science, Technology and Medicine, 1992.

Barton, P. I. and Pantelides, C. C.; Modeling of Combined Discrete/Continuous Processes. *AIChE Journal*, 40(6):966–979, June 1994.

Billups, S. C.; Algorithms for Complementarity Problems and Generalized Equations. PhD thesis, University of Wisconsin-Madison, August 1995.

Bullard, L. G. and Biegler, L. T.; Iterative Linear Programming Strategies for Constrained Simulation. *Comput. Chem. Eng.*, 15(4):239–254, 1991.

Bullard, L. G. and Biegler, L. T.; Iterated Linear Programming Strategies for Nonsmooth Simulation: Continuous and Mixed-Integer Approach. *Comput. Chem. Eng.*, 16(10), 1992

Dirske, S. P. and Ferris, M. C.; MCPLIB: A Collection of Nonlinear Mixed Complementarity Problems. Optimization Methods and Software, 5, 1995.

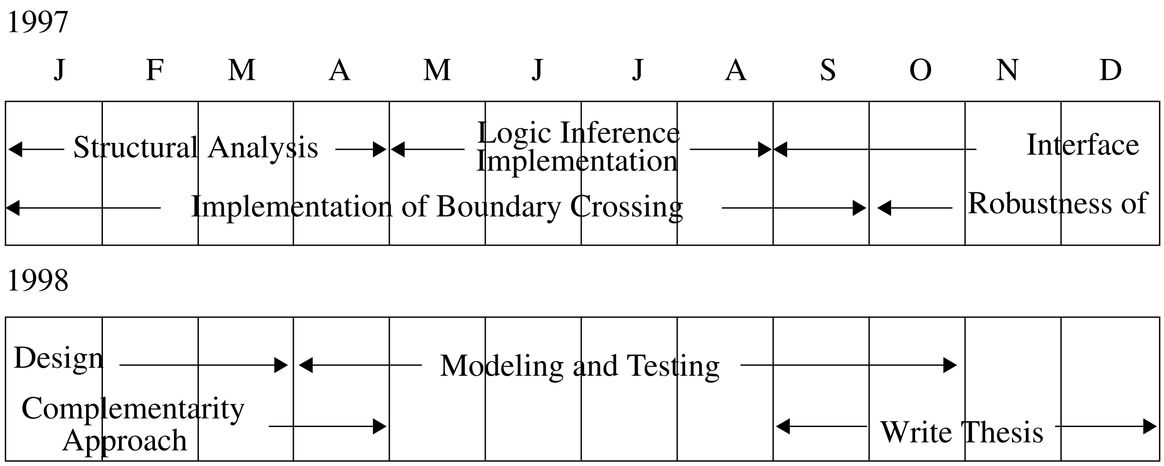
Ferris, M. C. and Pang, J. S. ; Engineering and Economic Applications of Complementarity Problems. Technical report, University of Colorado, 1995.

Grossmann, I. E. and Turkay, M.; Solution of Algebraic Systems of Disjunctive Equations. *Comput. Chem. Eng.*, 20:S339–44, 1996. Suppl. Part A.

An aspect that I have not mentioned, but that I anticipate it will be required, is the redesign of the existing interface to the ASCEND solver. I expect that the new interface tools not only will fit in with the current look of the ASCEND environment but also will provide the user with the capability of analyzing conditional models. An example of the new features that have to be added to the system is the capability of analyzing a conditional model at a boundary, where the neighboring sets of equations contain different structure.

The following is a tentative schedule for the remaining time of my research in this project.

Schedule of Proposed Work



6 CONCLUSIONS

There is a contribution to be made in the modeling world. Namely, the efficient representation of conditional models in an equation-based environment. We have described the modeling tools required to accomplish this, showing how the expressiveness of the modeling language increases with their incorporation. The computer implementation of these tools has been completed. There are surely other applications of the tools we propose that are being overlooked here. We hope to discover them.

We have also analyzed two alternative approaches to the solution of conditional models. The representation of conditional models as complementarity problems has been proposed and tested with small problems. The theoretical issues of the boundary crossing algorithm have been previously established, but an efficient computer implementation of it is considered to be a part

In the example of the linear mass balance, Turkay and Grossmann (1996) solve the problem by using a mixed-integer approach. The resulting MILP contains 18 binary variables, 66 continuous variables and 89 linear equations. In the complementarity formulation, the NLP contains 81 equations, 47 are linear, but the remaining 34 contain multiplications among slacks. The size of the problem is very similar, the difference will be in either using a branch and bound search in the MILP or dealing with the complementarity equations in the solution of one NLP.

We have thought that an interesting approach for dealing with conditional models would be to use the boundary crossing algorithm but representing some parts of the problem as a complementarity set of equations. In that way, we could obtain the advantages of both approaches. Once again, we consider this analysis to be part of the future work.

5 FUTURE WORK

In the previous two sections, we tried to emphasize the issues that still have to be addressed in both the representation and solution of conditional models; we also mentioned a description of a starting point for each of them. Making a brief summary list, we have:

1. Implementation of the boundary crossing algorithm. Issues inherently involved in this implementation (but with a much more general scope of application) include:
   i) Structural analysis in conditional models. This would include the idea of using the information provided by the structure of the problem in order to simplify the analysis, as outlined in section 3.2.1
   ii) Implementation of a logic inference algorithm.
2. Research in the field of the complementarity formulation for disjunctive systems of algebraic equations. Mainly, we need to investigate how to apply some of the ideas found in the literature to deal with the complementarity equations in an optimization mode and to improve the robustness of a NLP solver for the solution of this kind of equations.
3. Modeling and testing of large conditional models. From this analysis, we expect to be able to define the scope of application of the different approaches, to draw conclusions about the advantages and disadvantages of each of them and to combine them in order to get the best of both approaches.

be expected with problems of larger size.

Table 1: Examples of algebraic systems of disjunctive equations found in the literature

Example	Reference	Number of Equations	Number of Disjunctions	Number of Complementarity Equations	Iterations
Fluid transition	Zaher (1995)	7	1	1	11
Phase equilibria	Zaher (1995)	18	3	3	8
Heat exchanger	Zaher (1995)	56	2	4	12
Linear mass balance	Grossmann and Turkay (1996)	81	6	34	17
Pipeline network	Bullard and Biegler (1992)	250	38	76	27

The results of some of these examples are shown in Table 1. When possible, the initial values for the variables used were the same as those given in the reference (in some cases these values are not reported). To solve the examples, we used our ASCEND NLP solver which applies the modified Newton algorithm given in Westerberg and Director (1979). Since the hardware and the technique that we are using to get the solution is different, we are not comparing neither time nor number of iterations with other works. Still we consider it important to make remarks about some differences of the alternative approaches.

In the example of the pipeline network, the result of the complementarity representation is one NLP containing 250 equations, 76 of them containing 2 bilinear terms. In the same example, the number of boundaries in the boundary crossing algorithm would be 38 (that means 2^{38}=2.748779 x 10^{11} possible subregions) and the NLP to be solved in every subregion would contain 98 equations. The combinatorial complications present in examples like this impose a severe restriction to the boundary crossing algorithm. On the other hand, the reverse situation is also possible, and the boundary crossing may be clearly a better option than the complementarity formulation. For example, in a problem with only one disjunction, but 20 equations in each of the terms of the disjunction, we may not be able to solve a NLP in which 20 of the equations contain 20 bilinear terms each. However, the number of subregions in the boundary crossing algorithm would be only 2, and the possibility of applying this algorithm efficiently would be much greater.

```
MODEL flowsheet;
(* units *)
       f1                             IS_A cheap_feed;
       f2                             IS_A expensive_feed;
       c1,c2                          IS_A single_compressor;
       s1,s2                          IS_A staged_compressor;
       r1                             IS_A cheap_reactor;
       r2                             IS_A expensive_reactor;
       co1,co2                        IS_A cooler;
       h1,h2,h3                       IS_A heater;
       fl1                            IS_A flash;
       sp1                            IS_A splitter;
       m1                             IS_A mixer;
       v1                             IS_A expansion_valve;
(* boolean variables  *)
       select_feed1,   select_single1               IS_A boolean_var;
       select_cheapr1, select_single2               IS_A boolean_var;
(* define sets  *)
       m1.n_inputs :==2;
       sp1.n_outputs :== 2;
(* wire up flowsheet *)
       f1.stream, f2.stream, c1.input, s1.input     ARE_THE_SAME;
       c1.output, s1.output, m1.feed[2]             ARE_THE_SAME;
       m1.out,co1.input                             ARE_THE_SAME;
       co1.output, h1.input                         ARE_THE_SAME;
       h1.output, r1.input, r2.input                ARE_THE_SAME;
       r1.output, r2.output,v1.input                ARE_THE_SAME;
       v1.output,co2.input                          ARE_THE_SAME;
       co2.output, fl1.feed                         ARE_THE_SAME;
       fl1.liq, h2.input                            ARE_THE_SAME;
       fl1.vap, sp1.feed                            ARE_THE_SAME;
       sp1.out[1], h3.input                         ARE_THE_SAME;
       sp1.out[2],c2.input, s2.input                ARE_THE_SAME;
      c2.output, s2.output,m1.feed[1]               ARE_THE_SAME;
(* Conditional statements *)
       WHEN select_feed1
         CASE TRUE:
              USE f1;
         CASE FALSE:
              USE f2;
       END;
       WHEN select_single1
         CASE TRUE:
              USE c1;
         CASE FALSE:
              USE s1;
       END;
       WHEN select_cheapr1
         CASE TRUE:
              USE r1;
         CASE FALSE:
              USE r2;
       END;
       WHEN select_single2
         CASE TRUE:
              USE c2;
         CASE FALSE:
              USE s2;
       END;
END flowsheet;
```

FIGURE 6 ASCEND model for the superstructure of Figure 5

implementation of the WHEN statement and the logical relations. Its performance has been tested in several small problems. As an example, it was applied to the system presented in Figure 5 and Figure 6. The value of the four boolean variables will determine the structure of the problem to be solved. As mentioned above, the values of the boolean variables could be defined interactively by the user, but they also could be defined by some logic inference algorithm which would allow the automatic change of the structure of the problem. The NLP contains 137 invariant equations and 68 variant equations for a total of 205. The configuration defined by one of the feeds, two single-stage compressors and one of the reactors contains 169 equations, the 137 invariant and 32 active equations out of the 68 variant equations. Switching form one structure to another is done without the need of recompilation and, since to reconfigure the system requires only rebuilding several list of pointers, it is computationally very efficient.

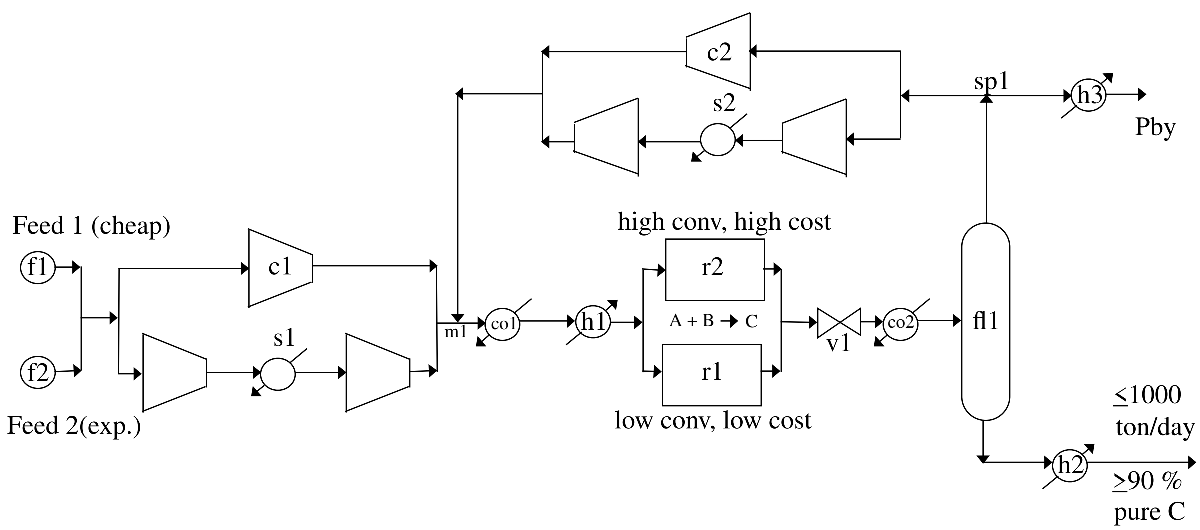


FIGURE 5 Superstructure taken from Turkay and Grossmann (1996).

4.2.2 Complementarity Formulation

Many examples of algebraic systems of disjunctive equations found in the literature have been represented and solved by using a complementarity formulation. In all the cases the results are encouraging. The solution of only one NLP is required and only a few iterations are used for convergence. However, it must be emphasized that, in all the cases, the number of equations in each disjunctive term (not the number of disjunctions) is very small; as a consequence, the complexity of the equations involving products among slack variables is not as bad as it can

Basically, a literature search has been conducted to find a possible solution for difficulty 3. To deal with nonconvexities in an optimization mode, Quesada and Grossman (1995) used linear relaxations based on the reformulation and linearization techniques for bilinear programming. Also, Viswanathan and Grossmann (1990) proposed augmenting the objective function with a penalty function that allows the violation of the linearizations of nonconvex constraints. A similar approach had been suggested by Bullard and Biegler (1991, 1992) in their iterated LP techniques for nonsmooth simulation. We are still not sure about how the application of these ideas could help to overcome the problems of having nonconvexities in the complementarity formulation. Research in this area is still ahead.

As mentioned before, the motivation for looking at a complementarity formulation is that only one NLP has to be solved. In general, we are presenting this approach as a very favorable alternative when the number of equations in each disjunctive term is small.

4 WORK TO DATE

4.1 MODELING

Both the WHEN statement and logical relations have already been incorporated to the ASCEND system as they were outlined above. We are currently testing this initial implementation, looking for improvements that could enhance its efficiency as modeling tools. To give an example of their application, we developed a simplified model for the superstructure given in Figure 5, taken from the work of Turkay and Grossmann (1996). In this example there are two alternative feedstocks, two possible choices of the reactor and two choices of the compression systems. Therefore, there are $2^4 = 16$ feasible configurations of the problem. All these 16 configurations are encapsulated in one ASCEND model containing 4 WHEN statements which depend on the value of 4 boolean variables. Figure 6 shows this model. The procedural section of the model and the model for each unit operation has been omitted for simplicity.

4.2 SOLUTION OF CONDITIONAL MODELS

4.2.1 Feeding a Conditional Model to an NLP solver

The mechanism suggested in section 3.2.2 has been implemented simultaneously with the

$$\begin{array}{l} h_1 \cdot h_2 + h_1 \cdot h_4 = 0 \\ h_3 \cdot h_2 + h_3 \cdot h_4 = 0 \end{array} \Rightarrow \begin{array}{l} h_1 \cdot (h_2 + h_4) = 0 \\ h_3 \cdot (h_2 + h_4) = 0 \end{array}$$

	h_1	h_2	h_3	h_4
1	$h_2 + h_4$	h_1	0	h_1
2	0	h_3	$h_2 + h_4$	h_3

FIGURE 4 Numerical singularities in factorable complementarity equations

the rows of the Jacobian corresponding to these equations. Note that the equations in Figure 4 contain the same four complementarity terms as the equations in (10), but they are grouped differently. In the case presented, if the solution of the problem is $h_2=h_4=0$, rows 1 and 2 of the Jacobian become numerically dependent as the Newton method approaches the solution (if $h_2+h_4 \sim 0$, we could get row 2 multiplying row 1 by the factor h_3/h_1). This is not the case in nonfactorable complementarity equations. There may be cases in which property 3 conflicts with property 4. Property 3 will have priority since it is what gives the simultaneous satisfaction of the equations in the same disjunctive term, a restriction that cannot be lost.

To get a formulation with all the properties outlined above is not a trivial exercise when the number of terms of the disjunction increases. However, for a disjunctive statement with two terms, the generalization to any number of equations is straightforward.

Many problems are incorporated with this kind of representation:

1. First of all, the problem grows quickly. If N is the number of equations in each term of the disjunction and D is the number of terms in the disjunction, the number of equations representing the disjunctive statement in the complementarity formulation is N(D+1).
2. The number of bilinear terms incorporated in each equation also grows with the number of equations in each term of the disjunction. The combinatorial nature of the problem is encapsulated here.
3. The performance of optimization techniques is badly affected by the introduction of nonconvexities (bilinear terms in the example) to the system of equations.
4. Numerical singularities can still arise for cases in which the solution resides on a boundary.

4. Every slack appears only once in each equation. There is no repeated slack index.

Posed in a formal way, given the disjunctive set of algebraic equations (two-term disjunction for simplicity)

$$\begin{bmatrix} f^1(x) = 0 \\ r(x) \le 0 \end{bmatrix} \vee \begin{bmatrix} f^2(x) = 0 \\ r(x) \ge 0 \end{bmatrix} \tag{11}$$

the problem is reformulated as

$$\begin{gathered} g(x,y) = \begin{pmatrix} f^1(x) - h_1 \\ f^2(x) - h_2 \\ r(x) - h_3 + h_4 \end{pmatrix} \\ g(x, y) = 0 \\ C(h_1, h_2, h_3, h_4) = 0 \\ h_i \ge 0 \qquad i \in [1 \ldots 4] \end{gathered} \tag{12}$$

where now *hi* represents a vector of slacks variables and $C(h_1,h_2,h_3,h_4)$ is used to represent the set of complementarity equations involving the elements of these vectors. Note that in the case of inequalities, the complementarity condition is expressed in terms of the slacks defined in the same equation. Also, we will require that the equations defined by $C(h_1,h_2,h_3,h_4)$ present the features outlined above for the equations given in (10). Property 1 is intended to avoid a change in the number of degrees of freedom, while property 3 is required to ensure the simultaneous satisfaction of all the equations in the same disjunctive term. Property 1 and 3 must always be satisfied. In our example, property 2 is only a consequence of 3 and 4 rather than a requirement, and may not be present in a general case.

Newton and quasi-Newton methods are the most popular techniques for the solution of NLP problems. Property 4 is intended to decrease the possibility of numerical singularities in the Jacobian which can occur during an iterative scheme based on these methods. If a repeated index occurs and, therefore, factorization is possible for some of the slacks, numerical singularities could arise as the iterative scheme approaches some solutions of the complementarity problem. This is illustrated in Figure 4, where we show a case of factorable complementarity equations and

$$Re - 64/f = h_1 \qquad Re - (0.206307/f)^4 = h_2$$
$$Re = 2100 + h_3 - h_4 \tag{7}$$

in terms of the slacks, the disjunctive statement becomes:

$$\begin{bmatrix} h_1 = 0 \\ h_3 = 0 \\ h_2, h_4 \geq 0 \end{bmatrix} \vee \begin{bmatrix} h_2 = 0 \\ h_4 = 0 \\ h_1, h_3 \geq 0 \end{bmatrix} \tag{8}$$

and the standard complementarity formulation is given by (7) and

$$h_1 \cdot h_2 = 0$$
$$h_3 \cdot h_4 = 0 \tag{9}$$
$$h_i \geq 0 \qquad i \in [1 \ldots 4]$$

It should be noted that the disjunctive statement requires either both h_1 and h_3 or both h_2 and h_4 simultaneously to be zero, and this is not a restriction in the complementarity formulation given by (9). The following formulation for disjunctive statements is proposed instead:

$$h_1 \cdot h_2 + h_3 \cdot h_4 = 0$$
$$h_1 \cdot h_4 + h_3 \cdot h_2 = 0 \tag{10}$$
$$h_i \geq 0 \qquad i \in [1 \ldots 4]$$

Since the slacks variables are all positive, the equations in (10) not only contains the complementarity condition given in (9) but also enforces simultaneous satisfaction of all the equations defined in the same term of the disjunction.

The complementarity equations given in (10) have the following properties:

1. The number of equations is equal to the number of equations in each term of the disjunction.
2. The number of terms in each equation is also equal to the number of equations in each term of the disjunction.
3. Every slack variable is multiplied by every slack in the other term of the disjunction.

implicitly or explicitly in such a CASE are set as active.

4. All the variables incident in the active set of equations are active.
5. The current problem to be solved consists of the active set of equations and variables.

The mechanism outline above is not precisely specific for the boundary crossing algorithm. For example, a branch and bound technique would also require the switching from one problem to another. Specifically for the boundary crossing algorithm, there is another major complication that must be addressed. While analyzing a boundary, the gradient step requires information about the equations in all of the subregions neighboring the boundary. That means the more than one set of equations has to be used simultaneously. We still do not address this and some other difficulties that we may not have discovered yet. However, based on previous programming exercises, we feel optimistic of being able to overcome any difficulty involved in this implementation in the time remaining for this research project.

3.2.3 Complementarity Representation of a Conditional Model

In this section, a complementarity formulation is proposed for the representation of conditional models. This formulation not only establishes the complementarity condition among alternative sets of equations but also enforces simultaneous satisfaction of all the equations in the solution set. The formulation proposed here is an extension of that presented by More(1994) who reformulates the nonlinear complementarity problem (3) as:

$$
\begin{gathered}
g(x,y) = \begin{pmatrix} f(x) - h \\ H \cdot x \end{pmatrix} \\
g(x, y) = 0 \qquad (5) \\
x \geq 0 \qquad\qquad y \geq 0
\end{gathered}
$$

where H is the diagonal matrix diag(h_i). The next simple example will show how a complementarity formulation should change for the case of disjunctive statements. Consider the flow transition given by:

$$
\begin{bmatrix} Re = 64/f \\ Re \leq 2100 \end{bmatrix} \vee \begin{bmatrix} Re = (0.206307/f)^4 \\ Re \geq 2100 \end{bmatrix} \qquad (6)
$$

defining slack variables h for both the equalities and inequalities:

some parameters, such as cost factors, mass balance coefficients, power of a correlation, etc. In such a case, the need of a combinatorial structural analysis is eliminated.

2. The disjunctions of the problem are disjointed, and its structural analysis may be done independently. There are two cases in which this may happen *i)* the incidence pattern of the disjunction are totally decoupled, and *ii)* the existing coupling still allows the calculation of the variables output assigned to the disjunctive equations separately in some precedence order.

We have applied all these ideas for analyzing by hand small conditional problems, and we believe that their practical implementation is possible. A computer implementation which can take advantages of them is desirable.

3.2.2 Implementation of the Boundary Crossing Algorithm

A part of this project will consist in the implementation and testing of the boundary crossing algorithm. In general, it can be considered that both the Newton and gradients steps required in this algorithm are already well defined. We are looking, however, for ideas which could give the computational efficiency needed for its application in a general purpose environment like ASCEND.

We mentioned in section 3.1.1 that the implementation of the WHEN statement in ASCEND would allow us to have available the data structure for all the variables and equations of the conditional model. Therefore, the first step in the implementation of a conditional solver is to decide how the solver is going to be fed with the correct structure and how to change this structure efficiently in an iterative solution scheme. To perform this task, we will use the notion of active equation and active variable. By active we mean "it is part of the problem currently being solved". The following steps would provide a mechanism to select the structure of the problem:

1. Initially, all the equations resulting from the compilation are considered as active.
2. Then, all the equations which are implicitly or explicitly (inside models) stated in the CASE of a WHEN statement are set as inactive. The equations set as inactive in this step constitute the variant set of equations. All the equations which remain active constitute the invariant set of equations.
3. Analyze the WHEN statements. According to the current values of the variables on which each WHEN statement depends, determine which of its CASEs applies. The equations stated

crossing algorithm.

3.2 SOLVING CONDITIONAL MODELS

3.2.1 Structural Analysis

The analysis required for partitioning the variables and the equations in a conditional model was outlined in section 2.2.1. In our point of view, the most serious disadvantage of this analysis is the combinatorial nature of the search consistency algorithm, which requires the analysis of all the feasible subregions resulting from the dissection of the feasible region. We think that, in general, this analysis cannot be avoided. If we want to be sure about the consistency of the conditional model, we would have to pay the price. However, based on the work of Allen and Westerberg (1976), we also think that sometimes it is possible to take considerable advantage of the structure of the problem, reducing the effort required by such an analysis.

In chemical engineering design and simulation, systems modeled with indexed equations and variables occur very often. Typical examples are a distillation column (indexed by the number of stages) and systems of equations arising in the discretization of an initial value problem (indexed by the number of time steps). If we think of a problem in which a disjunction exists in each of the repeated structures, the combinatorial complexity in the structural analysis would be unmanageable. However, intuition suggests that, in this kind of problem, the degrees of freedom analysis should not be affected by the number of repeated structures and, therefore, the analysis could be simplified. In his work, Allen used a representative incidence matrix to perform a systematic analysis of indexed equation sets. Again, this could not be applied to an arbitrary system of disjunctive equations, but if we know a priori that our problem contains an indexed repeated structure, then we may take advantage of it.

We have also observed special features of some problems which can contribute to simplifying the analysis. Even though they represent very particular cases, if found alone or in any combination, a simple checking in the computational implementation of the algorithm could give us saves in time and in the computational effort needed to perform the analysis. These are:

1. The incidence pattern of the different terms of the disjunction is the same. It happens sometimes that the alternative equations are only different because the difference in value of

(a)

```
laminar     IS_A boolean;
Re,f,k      IS_A factor;

CONDITIONAL
      condition: Re <= 2100;
END;

laminar == SATISFIED(condition);

eq1: Re = 64/f;
eq2: Re = (0.206307/f)^4;

WHEN laminar:
      CASE TRUE:
            USE eq1;
      CASE FALSE:
            USE eq2;
END;
```

(b)

```
valve_open       IS_A boolean;
pump_on          IS_A boolean;
full_tank        IS_A boolean;

valve_open == pump_on AND NOT(full_tank);
```

FIGURE 3 Examples of the syntax for the implementation of logical relations

AND, OR, and NOT are allowed. Equality in a logical equation can also be interpreted as an *if and only if* implication between two logical terms expressed in clausal form. Also, it should be noted that we can express any logical clause using the proposed syntax by simply writing the clause in one of the terms, and the constant boolean value TRUE in the other term of the logical equality. Example (a) is a more complete version of the model presented in Figure 2. In this case it can be noted that, when the value of a boolean variable depends on the truth value of a real expression, the real expression is defined and labeled inside the CONDITIONAL statement; then the logical operator SATISFIED will give the truth value of such an expression. If the truth value of the real expression is TRUE, the result of SATISFIED(label of expression) will be TRUE. The objective of the operator SATISFIED is twofold: 1) it makes the life of the compiler easier, since the logical expressions are decoupled from the real expressions on which they depend, and 2) the use of this syntax forces the user to define a boolean variable (laminar in the example) which will have the same truth value as the real expression. That is useful for browsing purposes.

It is important to emphasize that, by checking the value of the appropriate boolean variables after each iteration, the incorporation of logical relations will allow, if required, automatic change of the structure of the problem in an iterative scheme such as the boundary

the value of the symbol method. Once a configuration has been select, it will be kept unless the user decides to change it. Note that the user does not have to recompile the model to make the switching, since the data structure for both of the problems is available and, therefore, the reconfiguration of the system can be done automatically by simply changing the value of the conditional variable. An example of this application would be the synthesis of process networks.

On the other hand, in case (a) we cannot expect the boolean value of the variable laminar to be a user decision. Its value will depend on the value of the Reynolds number which is an unknown in the problem. Actually, the value of the variable laminar will be the truth value of the expression Re$\leq$2100. Here we have a conditional model. In an iterative solution scheme, we will expect that the value of the boolean variable laminar will change, and so will the structure of the system of equations that we have to solve.

3.1.2 Incorporation of Logical Relations

Motivated by some recent applications of logic in the solution of conditional models (Raman and Grossmann, 1994; Turkay and Grossmann, 1996) and trying to expand the ASCEND capabilities for modeling, an implementation of logical relations as a modeling tool is proposed. To our knowledge, no other general purpose equation-based modeling environment provides this capability.

In the implementation of logical relations, the equation based approach is going to be maintained. That is, the user will state the logical relations that must be true at the solution to the problem but not how to solve them. Each logical equation will have a residual attached to it. This residual will indicate if the expression is satisfied or not. Therefore, this incorporation also means that we need to provide a solver which knows how to deal with logical relations. The work of Raman and Grossman (1993) can be used as a starting point for such a solver implementation. They used formal procedures for performing logic inference depending on the way in which the logical relations are present (CNF or DNF).

Figure 3 illustrates two examples of the syntax proposed for the implementation of logical relations. The symbol double equality will indicate that we have a logical relation and will separate the two logical terms of the relation. In each of the two terms, logical operators such as

whose name is given in each of the cases should be declared inside the model which contains the WHEN statement.

2 The variables in the list of variables can be of any type among boolean, integer or symbol or any combination of them. That is, we are not limited to the use of boolean variables. Obviously, The list of values in each case must be in agreement with the list of variables in the number of elements and type of each element.

3 Names of arrays of models or equations are also allowed inside the scope of each case.

The proposed extension (allowing the user to define the domain of validity of both models and equations inside the cases of a WHEN statement) enormously increases the scope of modeling in an equation based modeling environment. Of course, this implementation may be possible in other modeling environments which support object oriented concepts, but we have proposed it as an extension to the ASCEND modeling language because ASCEND is the best one locally understood. Figure 2 shows two incomplete ASCEND models in which the WHEN statement is used, each in a different manner.

```
      (a)
laminar      IS_A boolean;
Re,f,k       IS_A factor;

eq1: Re = 64/f;
eq2: Re = (0.206307/f)^4;

WHEN laminar:
      CASE TRUE:
            USE eq1;
      CASE FALSE:
            USE eq2;
END;
```

```
      (b)
method             IS_A symbol;
simplified_flash   IS_A VLE_flash;
rigorous_flash     IS_S td_VLE_flash;

WHEN method
      CASE 'rigorous':
            USE rigorous_flash;
      CASE 'simplified':
            USE simplified_flash;
END;
```

FIGURE 2 Two different applications of the WHEN statement.

In case (b) the value of the symbol method is used only to select between two alternative configurations of the problem, a flash calculation assuming constant relative volatility (VLE_flash) or a flash calculation using a more rigorous thermodynamic calculation (td_VLE_flash). The method that is going to be used and, therefore, the value of the symbol method, is a user decision. We can expect that the user will select the simplified model looking for good initial values of the variables and then he will switch to the rigorous model by changing

with the solution of these problems. Some issues involved in the implementation of the boundary crossing algorithm will be briefly explained, and an extension of a complementarity formulation to represent conditional models will be proposed.

3.1 MODELING

In this section, an extension to the ASCEND modeling language is proposed. Essentially, it consists of implementing the WHEN statement and incorporating logical relations into the system.

3.1.1 The WHEN Statement

Originally, the syntax for incorporating conditional dependence of some equations of the model in an equation-based environment was suggested by Piela (1989). That syntax is very similar to the CASE equation of *gPROMS* and suffers the same limitations. Instead, we want to take advantage of the fact that ASCEND is based on object oriented concepts where model definitions can contain parts that contain parts to any level. Furthermore, in ASCEND, a simple relation is treated as an object by itself and can have a name. Based on these ideas, the proposed syntax for the WHEN statement is:

```
WHEN list_of_variables
      CASE list_of_values_1:
                              USE name_of_equation_1;
                              USE name_of_model_1;
      CASE list_of_values_2:
                              USE name_of_equation_2;
                              USE name_of_model_2;
      CASE list_of_values_nminus1:
                              USE name_of_equation_nminu1;
                              USE name_of_model_nminus1;
     OTHERWISE:
                              USE name_of_equation_n;
                              USE name_of_model_n;
END;
```

The following are important observations about the proposed implementation:

1 We are not dealing with conditional compilation. We create and have available the data structures for all of the variables and equations in each of the models. This is actually a requirement for the solution algorithms of conditional models. All the models and equations

quite arbitrary while the other is the vector of variables. Many commonly occurring problems have a more general form:

$$
\begin{aligned}
f^1(x)^T \cdot f^2(x) &= 0 \\
f^1(x) \geq 0 \quad f^2(x) &\geq 0
\end{aligned}
\tag{4}
$$

This problem is called the vertical nonlinear complementarity problem (Ferris and Pang, 1995). It is possible, of course, to have more than two vectors of functions in the above equations. Later, we will propose the representation of disjunctive sets of algebraic equations as complementarity problems. It is important to mention that we are aware of many disadvantages associated with this representation and, therefore, potential solutions to these problems will be mentioned as well. Our motivation is the fact that a complementarity formulation only requires the solution of a nonlinear system of equations, avoiding all of the complications encountered in the boundary crossing algorithm and the discrete decision making of a MINLP solution.

2.2.3 Solution of Conditional Models in an Equation-Based Modeling Environment

Finally in this section, we consider again the analysis of the previous implementations of conditional modeling to mention their status in the solution of these problems. Both the IF construct of *SpeedUp* and the CASE equation of *gPROMS* are used only to dynamically change the active set of equations during a dynamic simulation by evaluating the logical expression at any point in time. They assume, however, that the initial state of the system (and hence, the initial active set of equations) is known a priori. In other words, they do not address the problem of finding a consistent set of initial values that satisfy both the logical conditions and the active set of equations.

3 PROPOSED CONTRIBUTIONS

One of the goals of our research group has been to improve the ability to develop and solve process models. One of the results of this effort has been the development of ASCEND (Piela *et al*., 1991). Westerberg *et al.* (1994) discuss the essential features of the current ASCEND system and present several ways in which we can improve it to solve larger models and to increase its scope as a modeling environment. In section 3.1 we will propose the implementation of tools which will allow ASCEND to represent conditional models efficiently. Section 3.2 is concerned

In the gradient analysis at boundaries, a descent direction for the objective functions (norm of the residuals of the equations) local to each neighboring subregion is required for convergence. This direction d will be given by:

$$\begin{aligned} \min_{\alpha} \quad & d^T \cdot d \\ \text{s.t.} \quad & d = N \cdot \alpha \\ & e^T \cdot \alpha = 1 \\ & \alpha \geq 0 \end{aligned} \tag{2}$$

where the matrix N is known and is used to store the vectors of the gradients of the objective functions local to each neighboring subregion which borders a boundary, and e is a vector with all elements equal to one. The vector α, whose elements sum to one, is used to generate the direction d as the linear combination of the gradients which yields a vector of smallest length. Termination occurs if a minimum value of zero is obtained for d, which means that the gradient vectors of N cannot be confined to one side of a half-plane.

2.2.2.2 Complementarity Formulation

The class of problems known as complementarity problems has become increasingly popular as a tool for addressing practical problems arising in mathematical programming, engineering, and the sciences (Billups, 1995; Ferris and Pang, 1995). Dirske and Ferris (1995) describe the standard forms for the different classes of complementarity problems and give examples of how to formulate many popular problems as mixed complementarity problems (MCP). Billups (1995) proposes strategies which enhance the robustness of Newton-based methods for their application in these problems.

In general, the nonlinear complementarity problem is expressed as the following set of equations and inequality constraints:

$$\begin{aligned} x^T \cdot f(x) = 0 \\ x \geq 0 \qquad f(x) \geq 0 \end{aligned} \tag{3}$$

There is certain lack of symmetry in the previous formulation. One of the functions is

problem that is often solvable as a relaxed LP. They consider the nonlinear case although no guarantee of convergence is given for it. Also, based in the works of Raman and Grossmann (1993, 1994), Turkay and Grossmann (1996) address the solution of disjunctive set of nonlinear programming problems. It is shown that the use of logic can improve the efficiency and robustness of MINLP algorithms on the solution of structural optimization problems.

It is worth summarizing some of the observations and conclusions about MINLP algorithms arisen from the work of Zaher (1995): 1) it can be expected that during the solution of a MINLP, the algorithm will jump from one subregion to another which is remote from it, leading to the poor initialization of some of the nonlinear subproblems, 2) the nonlinear subproblems insist in converging the equations of what might be an incorrect subregion and 3) this method employs a highly mathematical approach which remains as a rigorous general purpose algorithm designed for problems where discrete decision making cannot be avoided.

The following two sections summarize two alternative approaches to the solution of conditional models.

2.2.2.1 Boundary Crossing Algorithm

Zaher (1991) introduced the boundary crossing algorithm as an alternative to the MINLP formulation for the solution of conditional models. In this approach, the inequalities defining the domain of each set of variant equations are considered as logical conditions which, when satisfied, will identify a particular subregion and, if violated, will identify an alternative one. In that way, every equation $r_{ik}(x)=0$ constitutes a boundary which will separate the feasible region in two neighboring subregions. In general, identification of a subregion requires a unique combination of boolean values for the collection of all logical conditions. If l is the number of boundary equations, as many as 2^l subregions can be created.

The boundary crossing algorithm has been given in Zaher (1991). Basically, the algorithm applies a Newton step for all the points not residing at boundaries. If during the iterative process a boundary is reached, then a Newton step cannot be taken because the system is nondifferentiable at that point (different subregions have different set of equations). Therefore, a gradient analysis is performed instead.

problem can be decomposed in smaller partitions which can be individually solved in some precedence order. Structural analysis is an indispensable tool in an equation-based modeling environment, and as it will be shown, conditional models make the need for this tool an even stronger requirement.

In conditional models, the sparsity pattern is expected to change from one subregion to another. This implies that a consistent set of independent variables for one subregion may not be valid for another subregion. For the same reason, block partitioning of the equations for one region may not be possible using the equations of another.

Zaher (1995) addressed the structural analysis of conditional models. A necessary condition for structural consistency in conditional models is that each of the alternative sets of equations must be structurally consistent. Hence, consistency of a conditional model is assessed by finding at least one consistent partitioning of the variables (independent-dependent) such that output assignment of all of the equations in each subregion can be performed. This requirement makes the problem combinatorial, since we have to perform the analysis for all of the possible alternative set of equations.

Regarding the equation partitioning, it is recommended one to obtain a representative incidence matrix rather than attempt to incorrectly decouple a wrong set (not the solution set) of equations. This representative incidence of equations in a disjunctive statement can be achieved by 'OR'ing the incidences of the inequalities (or boundary expressions) with each locally defined equation. Then, additional incidences are obtained for each equation in a subregion one at a time by merging the incidences of the equation with those of exactly one equation from all of the other subregions. It should be pointed out that it is also possible and valid to obtain an individual block partitioning while analyzing any particular subregion, and that the previous algorithm was proposed by Zaher when seeking efficiency.

2.2.2 Solution Algorithms for Conditional Models

As mentioned earlier, conditional models can be formulated as mixed integer programming problems. In a very recent work, Grossmann and Turkay (1996) show that, in the case of linear equations, the solution of a conditional model can be found by solving a MILP

2 BACKGROUND

2.1 BUILDING A CONDITIONAL MODEL

Previous implementations of conditional statements in an equation-based modeling environment have been reported. One such mechanism is the IF-THEN-ELSE construct of *SpeedUp* described by Pantelides (1988):

```
IF logical_condition THEN
      equation1
ELSE
      equation2
ENDIF
```

where both the logical conditions and the equations are expressed in terms of the model variables. Such a construct defines two system states corresponding to the IF and the ELSE clauses respectively. Multiple states may be described by nesting several IF statements.

Similarly, Barton (1992) and Barton and Pantelides (1994) incorporated the CASE equation to *gPROMS*. The CASE equation is used to define both the appropriate modeling equations in each state and the logical conditions for transitions among states. It covers multiple states within only one statement and has the advantage of successfully representing irreversible discontinuities.

There is a major difficulty in both of the previous approaches as tools for conditional modeling. They only allow the substitution of a list of equations (or arrays of equations) for another. In an equation-based modeling environment in which concepts like hierarchy (building complex models from small models) and inheritance (adding new features to a model) are constantly in use, that represents a severe disadvantage in efficiency. Thinking of how many equations have to be written down when substituting some of the units of a superstructure makes obvious the lack of flexibility in the scope of these approaches. Later in this paper, we will show our extension to these modeling capabilities with the incorporation of the WHEN statement to the ASCEND modeling environment.

2.2 SOLVING A CONDITIONAL MODEL

2.2.1 Structural Analysis for Conditional Models

Before attempting to solve a model, a structural analysis has to be performed to determine if the formulation is well posed. Similarly, a partitioning algorithm is applied to find if the original

as determined by the simultaneously calculated value of the flow type indicator (Reynolds or Mach number) relative to some critical value.

While many currently available equation-based modeling systems have been reported in the literature, only a few of them have given attention to conditional models (Pantelides, 1988; Piela,1989; Barton, 1992). Several problems arise when building and testing models of complex processes in an equation-based environment (providing initial guesses, analyzing degrees of freedom, efficiently computing solutions, etc.). The combinatorial nature of a conditional model makes the solution of these problems even a more difficult task. In this work, we propose the incorporation of a series of tools which enables the user of an equation-based modeling environment with the capability of dealing with complex conditional models. It is important to distinguish between two well defined and separated (but complementary) goals of this research project:

1. Efficient representation of a conditional model in an equation-based modeling environment. This is intended to be independent of any particular application, or of any solver or algorithm used for finding a solution to the system of equations. Potential applications of this capability varies from the simple substitution of one equation for another (as in the case of the laminar-turbulent flow transition), to the substitution of a section of a chemical plant for another (as can be required while analyzing and initializing a superstructure).
2. Implementation and testing of alternative approaches to the solution of a conditional model. Modeling is a user dependent task. Frequently, the same problem is formulated differently from one modeler to another and, as a consequence, different approaches and techniques could be used to find a solution to it. Conditional models can be formulated as mixed-integer programming problems. See for example Grossmann and Turkay (1996). Besides the mixed-integer formulation, some alternative approaches exist which, in principle, could avoid the need of discrete decisions. Research in this work will focus in two of them: the boundary crossing algorithm (Zaher, 1991; Zaher, 1993; Zaher, 1995) and the complementarity formulation (Billups, 1995). In particular, we will propose an approach for the implementation of the boundary crossing algorithm in an equation-based environment, and an extension to the standard complementarity formulation for the representation of conditional models.

1 INTRODUCTION

The design and simulation of engineering processes require one to find the solution to a large system of linear and/or nonlinear equations. This work is particularly concerned with the representation and solution of conditional models. Conditional models exist when the defining equations of the system depend on where the model solution lies. A conditional model consists of a system of equations expressed by two sets: a globally defined invariant set of equations and a variant (or locally defined) set of conditional equations which are expressed as disjunctions. Basically, as shown by Zaher(1993), and Grossmann and Turkay (1996), this problem reduces to representing and finding the solution to the system of disjunctive equations:

$$
\begin{gathered}
g(x) = 0 \\
\bigvee_{i \in D_k} \begin{bmatrix} h_{ik}(x) = 0 \\ r_{ik}(x) \le 0 \end{bmatrix} \quad k \in K \\
x \in R^n
\end{gathered}
\tag{1}
$$

where $g(x)$ and $h_{ik}(x)$ represent the invariant and the variant sets of equations respectively, K represents the set of disjunctions and the index i is used to indicate the i-th term in each disjunction D_k. The equations $g(x)$ can be said to be defined over the entire feasible region, while the inequalities define the domain of validity of each variant set of equations $h_{ik}(x)$. In that way, each variant set of equations is confined to some subregion resulting from the dissection of the feasible region (Zaher, 1991). The solution of the system will be given by the vector x satisfying the invariant set $g(x)$ and exactly one set of equations for each of the disjunctions, providing that the corresponding set of inequalities is satisfied.

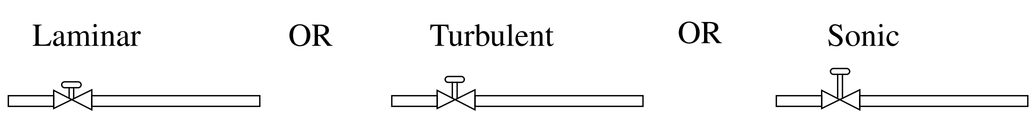


FIGURE 1 Fluid flow transition

Fluid transition represents an example commonly encountered in chemical engineering. This is illustrated in Figure 1 taken from the work of Zaher (1995). In this case, alternative equations for the transport properties and velocity bounds must be incorporated within the model

Representing, Analyzing and Solving Conditional Models in an Equation-Based Environment

by Vicente Rico-Ramírez

Advisor: Arthur W. Westerberg

Ph.D. Thesis Proposal

Department of Chemical Engineering

Carnegie Mellon University

Pittsburgh, PA

January 15, 1997

Abstract

Conditional models arise in chemical engineering when modeling systems involving physicochemical discontinuities, such as phase transitions. There are also cases where conditional cost functions are used in the optimization of process systems. Modeling tools for the efficient representation of conditional models in an equation-based environment are proposed in this paper. Also, some of the issues involved in the solution of these problems are discussed. The boundary crossing algorithm and the complementarity formulation are identified as alternatives to the MINLP algorithms for the solution of conditional models. In particular, we will propose an approach to the structural analysis of a conditional model, an extension to the standard complementarity formulation for the representation of conditional models, and an approach for the implementation of the boundary crossing algorithm in an equation-based environment.

More, J. J.; Global Methods for Nonlinear Complementarity Problems. Technical report, Argonne National Laboratory, Mathematics and Computer Science Division, April 1994.

Pantelides, C. C.; SPEEDUP-Recent Advances in Process Simulation. *Comput. Chem. Eng.*, 12(7):745–755, 1988.

Pantelides, C. C. and Barton, P. I.; Equation-Oriented Dynamic Simulation: Current Status and Future Perspectives. *Comput. Chem. Eng.*, 17S:263–285, 1993.

Piela, P., Epperly, T., Westerberg, K., Westerberg, A. W.; An Object-Oriented Computer Environment for Modeling and Analysis: The modeling language. *Comput. Chem. Eng.*, 15(1):53–72, 1991.

Piela, P.; ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis. PhD thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, April 1989.

Quesada, I. and Grossmann I. E.; A Global Optimization Algorithm for Linear Fractional and Bilinear Programs. *Journal of Global Optimization*, 6:39–76, 1995.

Raman, R. and Grossmann, I. E.; Symbolic Integration of Logic in Mixed-Integer Linear Programming Techniques for Process Synthesis. *Comput. Chem. Eng.*, 17(9):909–927, 1993.

Raman, R. and Grossmann, I. E.; Modeling and Computational Techniques for Logic Based Integer Programming. *Comput. Chem. Eng.*, 18(7):563–578, 1994.

Turkay, M. and Grossmann, I. E.; Logic-Based MINLP Algorithms for the Optimal Synthesis of Process Networks. Comput. Chem. Eng., 20(8):959–978, 1996.

Viswanathan, J. and Grossmann, I. E.; A Combined Penalty Function and Outer Approximation Method for MINLP Optimization. *Comput. Chem. Eng.*, 14(7):769–782, 1990.

Westerberg, A.W., Abbott, K. A., and Allan, B. A.; Plans for ASCEND IV: Our Next Generation Equational-Based Modeling Environment. Boston, Massachusetts, November 1994. AspenWorld'94.

Westerberg, A. W. and Director, S. W.; A Modified Least Square Algorithm for Solving Sparse n x n Sets of Nonlinear Equations. Technical report, Carnegie Mellon University, Engineering Design Research Center, DRC-06-5-79, January 1979.

Zaher, J. J.; Conditional Modeling. PhD thesis, Department of Chemical Engineering,Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1995.

Zaher, J. J.; Conditional Modeling in an Equation-Based Modeling Environment. The Annual AIChE National Meeting, 1991. paper 138c.

Zaher, J. J.; Conditional Programming. The Annual AIChE National Meeting, March 1993.

of this project. Some of the weaknesses and advantages of each of these approaches have been mentioned, but overcoming some of the difficulties and testing the approaches in large scale problems are issues still ahead as part of the future work. The following remain as some of the questions yet to be answered: *i)* How can the information provided by the structure of the problem help to simplifying the degrees of freedom analysis of a large conditional model? *ii)* How is the performance of a complementarity formulation against that of the boundary crossing algorithm in the solution of conditional models of practical size? *iii)* How is their performance in an optimization scheme? How does one overcome the difficulties incorporated by the bilinear terms in the complementarity formulation? Under what conditions can these approaches be considered a reliable alternative to MINLP algorithms in the simulation and optimization of large scale models?.

7 REFERENCES

Allen, G. L. and Westerberg, A. W.; Solution Procedures for Indexed Equation Sets: Structural Considerations. *AIChE Journal*, 22(3):549–558, May 1976.

Barton, P. I.; The Modeling and Simulation of Combined Discrete/Continuous Processes. PhD thesis, Department of Chemical Engineering, Imperial College of Science, Technology and Medicine, 1992.

Barton, P. I. and Pantelides, C. C.; Modeling of Combined Discrete/Continuous Processes. *AIChE Journal*, 40(6):966–979, June 1994.

Billups, S. C.; Algorithms for Complementarity Problems and Generalized Equations. PhD thesis, University of Wisconsin-Madison, August 1995.

Bullard, L. G. and Biegler, L. T.; Iterative Linear Programming Strategies for Constrained Simulation. *Comput. Chem. Eng.*, 15(4):239–254, 1991.

Bullard, L. G. and Biegler, L. T.; Iterated Linear Programming Strategies for Nonsmooth Simulation: Continuous and Mixed-Integer Approach. *Comput. Chem. Eng.*, 16(10), 1992

Dirske, S. P. and Ferris, M. C.; MCPLIB: A Collection of Nonlinear Mixed Complementarity Problems. Optimization Methods and Software, 5, 1995.

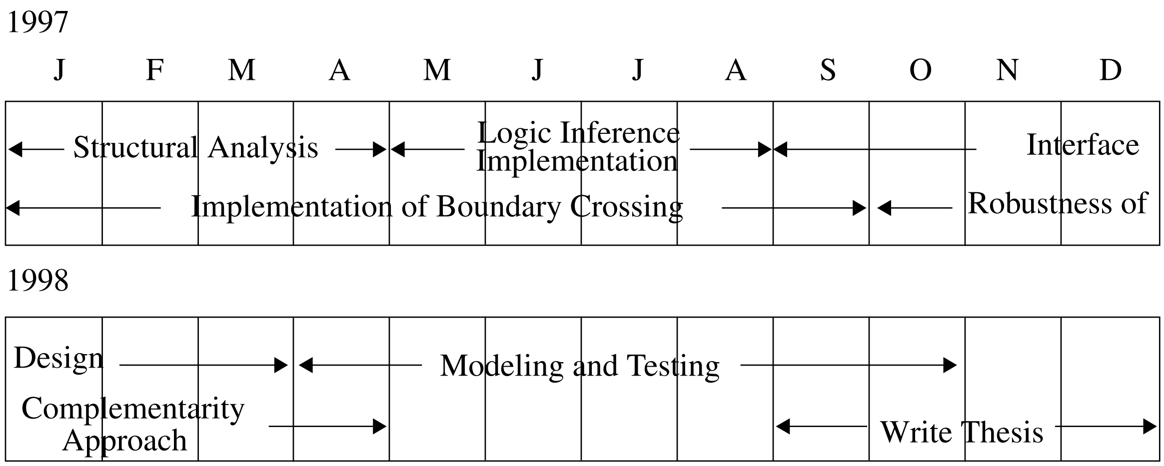
Ferris, M. C. and Pang, J. S. ; Engineering and Economic Applications of Complementarity Problems. Technical report, University of Colorado, 1995.

Grossmann, I. E. and Turkay, M.; Solution of Algebraic Systems of Disjunctive Equations. *Comput. Chem. Eng.*, 20:S339–44, 1996. Suppl. Part A.

An aspect that I have not mentioned, but that I anticipate it will be required, is the redesign of the existing interface to the ASCEND solver. I expect that the new interface tools not only will fit in with the current look of the ASCEND environment but also will provide the user with the capability of analyzing conditional models. An example of the new features that have to be added to the system is the capability of analyzing a conditional model at a boundary, where the neighboring sets of equations contain different structure.

The following is a tentative schedule for the remaining time of my research in this project.

Schedule of Proposed Work



6 CONCLUSIONS

There is a contribution to be made in the modeling world. Namely, the efficient representation of conditional models in an equation-based environment. We have described the modeling tools required to accomplish this, showing how the expressiveness of the modeling language increases with their incorporation. The computer implementation of these tools has been completed. There are surely other applications of the tools we propose that are being overlooked here. We hope to discover them.

We have also analyzed two alternative approaches to the solution of conditional models. The representation of conditional models as complementarity problems has been proposed and tested with small problems. The theoretical issues of the boundary crossing algorithm have been previously established, but an efficient computer implementation of it is considered to be a part

In the example of the linear mass balance, Turkay and Grossmann (1996) solve the problem by using a mixed-integer approach. The resulting MILP contains 18 binary variables, 66 continuous variables and 89 linear equations. In the complementarity formulation, the NLP contains 81 equations, 47 are linear, but the remaining 34 contain multiplications among slacks. The size of the problem is very similar, the difference will be in either using a branch and bound search in the MILP or dealing with the complementarity equations in the solution of one NLP.

We have thought that an interesting approach for dealing with conditional models would be to use the boundary crossing algorithm but representing some parts of the problem as a complementarity set of equations. In that way, we could obtain the advantages of both approaches. Once again, we consider this analysis to be part of the future work.

5 FUTURE WORK

In the previous two sections, we tried to emphasize the issues that still have to be addressed in both the representation and solution of conditional models; we also mentioned a description of a starting point for each of them. Making a brief summary list, we have:

1. Implementation of the boundary crossing algorithm. Issues inherently involved in this implementation (but with a much more general scope of application) include:
   i) Structural analysis in conditional models. This would include the idea of using the information provided by the structure of the problem in order to simplify the analysis, as outlined in section 3.2.1
   ii) Implementation of a logic inference algorithm.
2. Research in the field of the complementarity formulation for disjunctive systems of algebraic equations. Mainly, we need to investigate how to apply some of the ideas found in the literature to deal with the complementarity equations in an optimization mode and to improve the robustness of a NLP solver for the solution of this kind of equations.
3. Modeling and testing of large conditional models. From this analysis, we expect to be able to define the scope of application of the different approaches, to draw conclusions about the advantages and disadvantages of each of them and to combine them in order to get the best of both approaches.

be expected with problems of larger size.

Table 1: Examples of algebraic systems of disjunctive equations found in the literature

Example	Reference	Number of Equations	Number of Disjunctions	Number of Complementarity Equations	Iterations
Fluid transition	Zaher (1995)	7	1	1	11
Phase equilibria	Zaher (1995)	18	3	3	8
Heat exchanger	Zaher (1995)	56	2	4	12
Linear mass balance	Grossmann and Turkay (1996)	81	6	34	17
Pipeline network	Bullard and Biegler (1992)	250	38	76	27

The results of some of these examples are shown in Table 1. When possible, the initial values for the variables used were the same as those given in the reference (in some cases these values are not reported). To solve the examples, we used our ASCEND NLP solver which applies the modified Newton algorithm given in Westerberg and Director (1979). Since the hardware and the technique that we are using to get the solution is different, we are not comparing neither time nor number of iterations with other works. Still we consider it important to make remarks about some differences of the alternative approaches.

In the example of the pipeline network, the result of the complementarity representation is one NLP containing 250 equations, 76 of them containing 2 bilinear terms. In the same example, the number of boundaries in the boundary crossing algorithm would be 38 (that means 2^{38}=2.748779 x 10^{11} possible subregions) and the NLP to be solved in every subregion would contain 98 equations. The combinatorial complications present in examples like this impose a severe restriction to the boundary crossing algorithm. On the other hand, the reverse situation is also possible, and the boundary crossing may be clearly a better option than the complementarity formulation. For example, in a problem with only one disjunction, but 20 equations in each of the terms of the disjunction, we may not be able to solve a NLP in which 20 of the equations contain 20 bilinear terms each. However, the number of subregions in the boundary crossing algorithm would be only 2, and the possibility of applying this algorithm efficiently would be much greater.

```
MODEL flowsheet;
(* units *)
       f1                            IS_A cheap_feed;
       f2                            IS_A expensive_feed;
       c1,c2                         IS_A single_compressor;
       s1,s2                         IS_A staged_compressor;
       r1                            IS_A cheap_reactor;
       r2                            IS_A expensive_reactor;
       co1,co2                       IS_A cooler;
       h1,h2,h3                      IS_A heater;
       fl1                           IS_A flash;
       sp1                           IS_A splitter;
       m1                            IS_A mixer;
       v1                            IS_A expansion_valve;
(* boolean variables  *)
       select_feed1,   select_single1               IS_A boolean_var;
       select_cheapr1, select_single2               IS_A boolean_var;
(* define sets  *)
       m1.n_inputs :==2;
       sp1.n_outputs :== 2;
(* wire up flowsheet *)
       f1.stream, f2.stream, c1.input, s1.input     ARE_THE_SAME;
       c1.output, s1.output, m1.feed[2]             ARE_THE_SAME;
       m1.out,co1.input                             ARE_THE_SAME;
       co1.output, h1.input                         ARE_THE_SAME;
       h1.output, r1.input, r2.input                ARE_THE_SAME;
       r1.output, r2.output,v1.input                ARE_THE_SAME;
       v1.output,co2.input                          ARE_THE_SAME;
       co2.output, fl1.feed                         ARE_THE_SAME;
       fl1.liq, h2.input                            ARE_THE_SAME;
       fl1.vap, sp1.feed                            ARE_THE_SAME;
       sp1.out[1], h3.input                         ARE_THE_SAME;
       sp1.out[2],c2.input, s2.input                ARE_THE_SAME;
      c2.output, s2.output,m1.feed[1]               ARE_THE_SAME;
(* Conditional statements *)
       WHEN select_feed1
         CASE TRUE:
              USE f1;
         CASE FALSE:
              USE f2;
       END;
       WHEN select_single1
         CASE TRUE:
              USE c1;
         CASE FALSE:
              USE s1;
       END;
       WHEN select_cheapr1
         CASE TRUE:
              USE r1;
         CASE FALSE:
              USE r2;
       END;
       WHEN select_single2
         CASE TRUE:
              USE c2;
         CASE FALSE:
              USE s2;
       END;
END flowsheet;
```

FIGURE 6 ASCEND model for the superstructure of Figure 5

implementation of the WHEN statement and the logical relations. Its performance has been tested in several small problems. As an example, it was applied to the system presented in Figure 5 and Figure 6. The value of the four boolean variables will determine the structure of the problem to be solved. As mentioned above, the values of the boolean variables could be defined interactively by the user, but they also could be defined by some logic inference algorithm which would allow the automatic change of the structure of the problem. The NLP contains 137 invariant equations and 68 variant equations for a total of 205. The configuration defined by one of the feeds, two single-stage compressors and one of the reactors contains 169 equations, the 137 invariant and 32 active equations out of the 68 variant equations. Switching form one structure to another is done without the need of recompilation and, since to reconfigure the system requires only rebuilding several list of pointers, it is computationally very efficient.

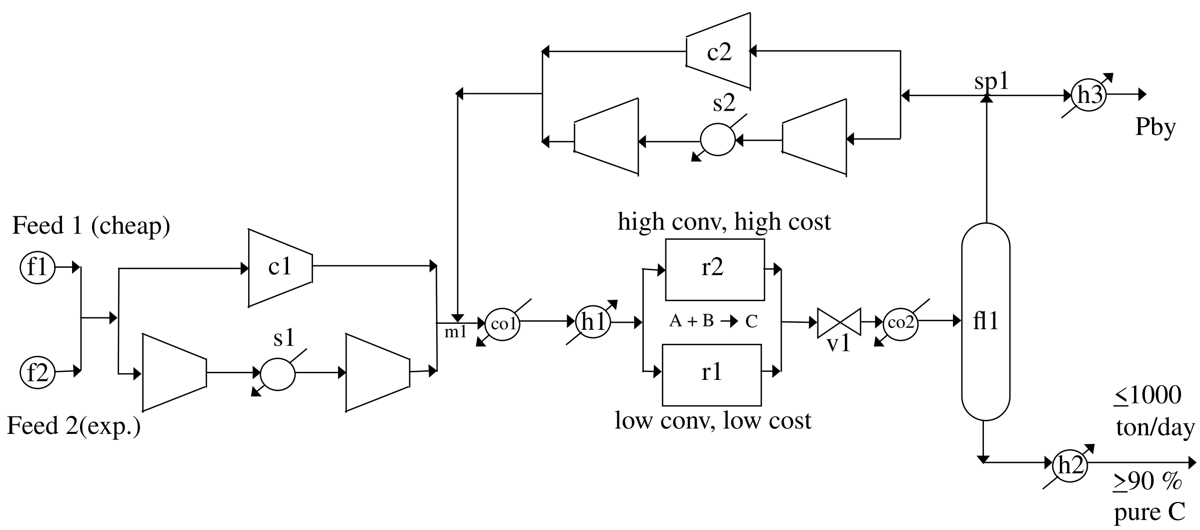


FIGURE 5 Superstructure taken from Turkay and Grossmann (1996).

4.2.2 Complementarity Formulation

Many examples of algebraic systems of disjunctive equations found in the literature have been represented and solved by using a complementarity formulation. In all the cases the results are encouraging. The solution of only one NLP is required and only a few iterations are used for convergence. However, it must be emphasized that, in all the cases, the number of equations in each disjunctive term (not the number of disjunctions) is very small; as a consequence, the complexity of the equations involving products among slack variables is not as bad as it can

Basically, a literature search has been conducted to find a possible solution for difficulty 3. To deal with nonconvexities in an optimization mode, Quesada and Grossman (1995) used linear relaxations based on the reformulation and linearization techniques for bilinear programming. Also, Viswanathan and Grossmann (1990) proposed augmenting the objective function with a penalty function that allows the violation of the linearizations of nonconvex constraints. A similar approach had been suggested by Bullard and Biegler (1991, 1992) in their iterated LP techniques for nonsmooth simulation. We are still not sure about how the application of these ideas could help to overcome the problems of having nonconvexities in the complementarity formulation. Research in this area is still ahead.

As mentioned before, the motivation for looking at a complementarity formulation is that only one NLP has to be solved. In general, we are presenting this approach as a very favorable alternative when the number of equations in each disjunctive term is small.

4 WORK TO DATE

4.1 MODELING

Both the WHEN statement and logical relations have already been incorporated to the ASCEND system as they were outlined above. We are currently testing this initial implementation, looking for improvements that could enhance its efficiency as modeling tools. To give an example of their application, we developed a simplified model for the superstructure given in Figure 5, taken from the work of Turkay and Grossmann (1996). In this example there are two alternative feedstocks, two possible choices of the reactor and two choices of the compression systems. Therefore, there are $2^4 = 16$ feasible configurations of the problem. All these 16 configurations are encapsulated in one ASCEND model containing 4 WHEN statements which depend on the value of 4 boolean variables. Figure 6 shows this model. The procedural section of the model and the model for each unit operation has been omitted for simplicity.

4.2 SOLUTION OF CONDITIONAL MODELS

4.2.1 Feeding a Conditional Model to an NLP solver

The mechanism suggested in section 3.2.2 has been implemented simultaneously with the

$$\begin{matrix} h_1 \cdot h_2 + h_1 \cdot h_4 = 0 \\ h_3 \cdot h_2 + h_3 \cdot h_4 = 0 \end{matrix} \Rightarrow \begin{matrix} h_1 \cdot (h_2 + h_4) = 0 \\ h_3 \cdot (h_2 + h_4) = 0 \end{matrix}$$

	h_1	h_2	h_3	h_4
1	$h_2 + h_4$	h_1	0	h_1
2	0	h_3	$h_2 + h_4$	h_3

FIGURE 4 Numerical singularities in factorable complementarity equations

the rows of the Jacobian corresponding to these equations. Note that the equations in Figure 4 contain the same four complementarity terms as the equations in (10), but they are grouped differently. In the case presented, if the solution of the problem is $h_2=h_4=0$, rows 1 and 2 of the Jacobian become numerically dependent as the Newton method approaches the solution (if $h_2+h_4 \sim 0$, we could get row 2 multiplying row 1 by the factor h_3/h_1). This is not the case in nonfactorable complementarity equations. There may be cases in which property 3 conflicts with property 4. Property 3 will have priority since it is what gives the simultaneous satisfaction of the equations in the same disjunctive term, a restriction that cannot be lost.

To get a formulation with all the properties outlined above is not a trivial exercise when the number of terms of the disjunction increases. However, for a disjunctive statement with two terms, the generalization to any number of equations is straightforward.

Many problems are incorporated with this kind of representation:

1. First of all, the problem grows quickly. If N is the number of equations in each term of the disjunction and D is the number of terms in the disjunction, the number of equations representing the disjunctive statement in the complementarity formulation is N(D+1).
2. The number of bilinear terms incorporated in each equation also grows with the number of equations in each term of the disjunction. The combinatorial nature of the problem is encapsulated here.
3. The performance of optimization techniques is badly affected by the introduction of nonconvexities (bilinear terms in the example) to the system of equations.
4. Numerical singularities can still arise for cases in which the solution resides on a boundary.

4. Every slack appears only once in each equation. There is no repeated slack index.

Posed in a formal way, given the disjunctive set of algebraic equations (two-term disjunction for simplicity)

$$\begin{bmatrix} f^1(x) = 0 \\ r(x) \le 0 \end{bmatrix} \vee \begin{bmatrix} f^2(x) = 0 \\ r(x) \ge 0 \end{bmatrix} \tag{11}$$

the problem is reformulated as

$$\begin{gathered} g(x,y) = \begin{pmatrix} f^1(x) - h_1 \\ f^2(x) - h_2 \\ r(x) - h_3 + h_4 \end{pmatrix} \\ g(x,y) = 0 \\ C(h_1, h_2, h_3, h_4) = 0 \\ h_i \ge 0 \qquad i \in [1 \ldots 4] \end{gathered} \tag{12}$$

where now *hi* represents a vector of slacks variables and $C(h_1,h_2,h_3,h_4)$ is used to represent the set of complementarity equations involving the elements of these vectors. Note that in the case of inequalities, the complementarity condition is expressed in terms of the slacks defined in the same equation. Also, we will require that the equations defined by $C(h_1,h_2,h_3,h_4)$ present the features outlined above for the equations given in (10). Property 1 is intended to avoid a change in the number of degrees of freedom, while property 3 is required to ensure the simultaneous satisfaction of all the equations in the same disjunctive term. Property 1 and 3 must always be satisfied. In our example, property 2 is only a consequence of 3 and 4 rather than a requirement, and may not be present in a general case.

Newton and quasi-Newton methods are the most popular techniques for the solution of NLP problems. Property 4 is intended to decrease the possibility of numerical singularities in the Jacobian which can occur during an iterative scheme based on these methods. If a repeated index occurs and, therefore, factorization is possible for some of the slacks, numerical singularities could arise as the iterative scheme approaches some solutions of the complementarity problem. This is illustrated in Figure 4, where we show a case of factorable complementarity equations and

$$Re - 64/f = h_1 \qquad Re - (0.206307/f)^4 = h_2$$
$$Re = 2100 + h_3 - h_4 \tag{7}$$

in terms of the slacks, the disjunctive statement becomes:

$$\begin{bmatrix} h_1 = 0 \\ h_3 = 0 \\ h_2, h_4 \geq 0 \end{bmatrix} \vee \begin{bmatrix} h_2 = 0 \\ h_4 = 0 \\ h_1, h_3 \geq 0 \end{bmatrix} \tag{8}$$

and the standard complementarity formulation is given by (7) and

$$h_1 \cdot h_2 = 0$$
$$h_3 \cdot h_4 = 0 \tag{9}$$
$$h_i \geq 0 \qquad i \in [1 \ldots 4]$$

It should be noted that the disjunctive statement requires either both h_1 and h_3 or both h_2 and h_4 simultaneously to be zero, and this is not a restriction in the complementarity formulation given by (9). The following formulation for disjunctive statements is proposed instead:

$$h_1 \cdot h_2 + h_3 \cdot h_4 = 0$$
$$h_1 \cdot h_4 + h_3 \cdot h_2 = 0 \tag{10}$$
$$h_i \geq 0 \qquad i \in [1 \ldots 4]$$

Since the slacks variables are all positive, the equations in (10) not only contains the complementarity condition given in (9) but also enforces simultaneous satisfaction of all the equations defined in the same term of the disjunction.

The complementarity equations given in (10) have the following properties:

1. The number of equations is equal to the number of equations in each term of the disjunction.
2. The number of terms in each equation is also equal to the number of equations in each term of the disjunction.
3. Every slack variable is multiplied by every slack in the other term of the disjunction.

implicitly or explicitly in such a CASE are set as active.

4. All the variables incident in the active set of equations are active.
5. The current problem to be solved consists of the active set of equations and variables.

The mechanism outline above is not precisely specific for the boundary crossing algorithm. For example, a branch and bound technique would also require the switching from one problem to another. Specifically for the boundary crossing algorithm, there is another major complication that must be addressed. While analyzing a boundary, the gradient step requires information about the equations in all of the subregions neighboring the boundary. That means the more than one set of equations has to be used simultaneously. We still do not address this and some other difficulties that we may not have discovered yet. However, based on previous programming exercises, we feel optimistic of being able to overcome any difficulty involved in this implementation in the time remaining for this research project.

3.2.3 Complementarity Representation of a Conditional Model

In this section, a complementarity formulation is proposed for the representation of conditional models. This formulation not only establishes the complementarity condition among alternative sets of equations but also enforces simultaneous satisfaction of all the equations in the solution set. The formulation proposed here is an extension of that presented by More(1994) who reformulates the nonlinear complementarity problem (3) as:

$$
\begin{gathered}
g(x,y) = \begin{pmatrix} f(x) - h \\ H \cdot x \end{pmatrix} \\
g(x, y) = 0 \\
x \geq 0 \qquad\qquad y \geq 0
\end{gathered}
\tag{5}
$$

where H is the diagonal matrix diag(h_i). The next simple example will show how a complementarity formulation should change for the case of disjunctive statements. Consider the flow transition given by:

$$
\begin{bmatrix} Re = 64/f \\ Re \leq 2100 \end{bmatrix} \vee \begin{bmatrix} Re = (0.206307/f)^4 \\ Re \geq 2100 \end{bmatrix} \tag{6}
$$

defining slack variables h for both the equalities and inequalities:

some parameters, such as cost factors, mass balance coefficients, power of a correlation, etc. In such a case, the need of a combinatorial structural analysis is eliminated.

2. The disjunctions of the problem are disjointed, and its structural analysis may be done independently. There are two cases in which this may happen *i)* the incidence pattern of the disjunction are totally decoupled, and *ii)* the existing coupling still allows the calculation of the variables output assigned to the disjunctive equations separately in some precedence order.

We have applied all these ideas for analyzing by hand small conditional problems, and we believe that their practical implementation is possible. A computer implementation which can take advantages of them is desirable.

3.2.2 Implementation of the Boundary Crossing Algorithm

A part of this project will consist in the implementation and testing of the boundary crossing algorithm. In general, it can be considered that both the Newton and gradients steps required in this algorithm are already well defined. We are looking, however, for ideas which could give the computational efficiency needed for its application in a general purpose environment like ASCEND.

We mentioned in section 3.1.1 that the implementation of the WHEN statement in ASCEND would allow us to have available the data structure for all the variables and equations of the conditional model. Therefore, the first step in the implementation of a conditional solver is to decide how the solver is going to be fed with the correct structure and how to change this structure efficiently in an iterative solution scheme. To perform this task, we will use the notion of active equation and active variable. By active we mean "it is part of the problem currently being solved". The following steps would provide a mechanism to select the structure of the problem:

1. Initially, all the equations resulting from the compilation are considered as active.
2. Then, all the equations which are implicitly or explicitly (inside models) stated in the CASE of a WHEN statement are set as inactive. The equations set as inactive in this step constitute the variant set of equations. All the equations which remain active constitute the invariant set of equations.
3. Analyze the WHEN statements. According to the current values of the variables on which each WHEN statement depends, determine which of its CASEs applies. The equations stated

crossing algorithm.

3.2 SOLVING CONDITIONAL MODELS

3.2.1 Structural Analysis

The analysis required for partitioning the variables and the equations in a conditional model was outlined in section 2.2.1. In our point of view, the most serious disadvantage of this analysis is the combinatorial nature of the search consistency algorithm, which requires the analysis of all the feasible subregions resulting from the dissection of the feasible region. We think that, in general, this analysis cannot be avoided. If we want to be sure about the consistency of the conditional model, we would have to pay the price. However, based on the work of Allen and Westerberg (1976), we also think that sometimes it is possible to take considerable advantage of the structure of the problem, reducing the effort required by such an analysis.

In chemical engineering design and simulation, systems modeled with indexed equations and variables occur very often. Typical examples are a distillation column (indexed by the number of stages) and systems of equations arising in the discretization of an initial value problem (indexed by the number of time steps). If we think of a problem in which a disjunction exists in each of the repeated structures, the combinatorial complexity in the structural analysis would be unmanageable. However, intuition suggests that, in this kind of problem, the degrees of freedom analysis should not be affected by the number of repeated structures and, therefore, the analysis could be simplified. In his work, Allen used a representative incidence matrix to perform a systematic analysis of indexed equation sets. Again, this could not be applied to an arbitrary system of disjunctive equations, but if we know a priori that our problem contains an indexed repeated structure, then we may take advantage of it.

We have also observed special features of some problems which can contribute to simplifying the analysis. Even though they represent very particular cases, if found alone or in any combination, a simple checking in the computational implementation of the algorithm could give us saves in time and in the computational effort needed to perform the analysis. These are:

1. The incidence pattern of the different terms of the disjunction is the same. It happens sometimes that the alternative equations are only different because the difference in value of

(a)

```
laminar     IS_A boolean;
Re,f,k      IS_A factor;

CONDITIONAL
      condition: Re <= 2100;
END;

laminar == SATISFIED(condition);

eq1: Re = 64/f;
eq2: Re = (0.206307/f)^4;

WHEN laminar:
      CASE TRUE:
            USE eq1;
      CASE FALSE:
            USE eq2;
END;
```

(b)

```
valve_open        IS_A boolean;
pump_on           IS_A boolean;
full_tank         IS_A boolean;

valve_open == pump_on AND NOT(full_tank);
```

FIGURE 3 Examples of the syntax for the implementation of logical relations

AND, OR, and NOT are allowed. Equality in a logical equation can also be interpreted as an *if and only if* implication between two logical terms expressed in clausal form. Also, it should be noted that we can express any logical clause using the proposed syntax by simply writing the clause in one of the terms, and the constant boolean value TRUE in the other term of the logical equality. Example (a) is a more complete version of the model presented in Figure 2. In this case it can be noted that, when the value of a boolean variable depends on the truth value of a real expression, the real expression is defined and labeled inside the CONDITIONAL statement; then the logical operator SATISFIED will give the truth value of such an expression. If the truth value of the real expression is TRUE, the result of SATISFIED(label of expression) will be TRUE. The objective of the operator SATISFIED is twofold: 1) it makes the life of the compiler easier, since the logical expressions are decoupled from the real expressions on which they depend, and 2) the use of this syntax forces the user to define a boolean variable (laminar in the example) which will have the same truth value as the real expression. That is useful for browsing purposes.

It is important to emphasize that, by checking the value of the appropriate boolean variables after each iteration, the incorporation of logical relations will allow, if required, automatic change of the structure of the problem in an iterative scheme such as the boundary

the value of the symbol method. Once a configuration has been select, it will be kept unless the user decides to change it. Note that the user does not have to recompile the model to make the switching, since the data structure for both of the problems is available and, therefore, the reconfiguration of the system can be done automatically by simply changing the value of the conditional variable. An example of this application would be the synthesis of process networks.

On the other hand, in case (a) we cannot expect the boolean value of the variable laminar to be a user decision. Its value will depend on the value of the Reynolds number which is an unknown in the problem. Actually, the value of the variable laminar will be the truth value of the expression $Re \leq 2100$. Here we have a conditional model. In an iterative solution scheme, we will expect that the value of the boolean variable laminar will change, and so will the structure of the system of equations that we have to solve.

3.1.2 Incorporation of Logical Relations

Motivated by some recent applications of logic in the solution of conditional models (Raman and Grossmann, 1994; Turkay and Grossmann, 1996) and trying to expand the ASCEND capabilities for modeling, an implementation of logical relations as a modeling tool is proposed. To our knowledge, no other general purpose equation-based modeling environment provides this capability.

In the implementation of logical relations, the equation based approach is going to be maintained. That is, the user will state the logical relations that must be true at the solution to the problem but not how to solve them. Each logical equation will have a residual attached to it. This residual will indicate if the expression is satisfied or not. Therefore, this incorporation also means that we need to provide a solver which knows how to deal with logical relations. The work of Raman and Grossman (1993) can be used as a starting point for such a solver implementation. They used formal procedures for performing logic inference depending on the way in which the logical relations are present (CNF or DNF).

Figure 3 illustrates two examples of the syntax proposed for the implementation of logical relations. The symbol double equality will indicate that we have a logical relation and will separate the two logical terms of the relation. In each of the two terms, logical operators such as

whose name is given in each of the cases should be declared inside the model which contains the WHEN statement.

2 The variables in the list of variables can be of any type among boolean, integer or symbol or any combination of them. That is, we are not limited to the use of boolean variables. Obviously, The list of values in each case must be in agreement with the list of variables in the number of elements and type of each element.

3 Names of arrays of models or equations are also allowed inside the scope of each case.

The proposed extension (allowing the user to define the domain of validity of both models and equations inside the cases of a WHEN statement) enormously increases the scope of modeling in an equation based modeling environment. Of course, this implementation may be possible in other modeling environments which support object oriented concepts, but we have proposed it as an extension to the ASCEND modeling language because ASCEND is the best one locally understood. Figure 2 shows two incomplete ASCEND models in which the WHEN statement is used, each in a different manner.

(a)

```
laminar      IS_A boolean;
Re,f,k       IS_A factor;

eq1: Re = 64/f;
eq2: Re = (0.206307/f)^4;

WHEN laminar:
      CASE TRUE:
            USE eq1;
      CASE FALSE:
            USE eq2;
END;
```

(b)

```
method             IS_A symbol;
simplified_flash   IS_A VLE_flash;
rigorous_flash     IS_S td_VLE_flash;

WHEN method
      CASE 'rigorous':
            USE rigorous_flash;
      CASE 'simplified':
            USE simplified_flash;
END;
```

FIGURE 2 Two different applications of the WHEN statement.

In case (b) the value of the symbol method is used only to select between two alternative configurations of the problem, a flash calculation assuming constant relative volatility (VLE_flash) or a flash calculation using a more rigorous thermodynamic calculation (td_VLE_flash). The method that is going to be used and, therefore, the value of the symbol method, is a user decision. We can expect that the user will select the simplified model looking for good initial values of the variables and then he will switch to the rigorous model by changing

with the solution of these problems. Some issues involved in the implementation of the boundary crossing algorithm will be briefly explained, and an extension of a complementarity formulation to represent conditional models will be proposed.

3.1 MODELING

In this section, an extension to the ASCEND modeling language is proposed. Essentially, it consists of implementing the WHEN statement and incorporating logical relations into the system.

3.1.1 The WHEN Statement

Originally, the syntax for incorporating conditional dependence of some equations of the model in an equation-based environment was suggested by Piela (1989). That syntax is very similar to the CASE equation of *gPROMS* and suffers the same limitations. Instead, we want to take advantage of the fact that ASCEND is based on object oriented concepts where model definitions can contain parts that contain parts to any level. Furthermore, in ASCEND, a simple relation is treated as an object by itself and can have a name. Based on these ideas, the proposed syntax for the WHEN statement is:

```
WHEN list_of_variables
      CASE list_of_values_1:
                              USE name_of_equation_1;
                              USE name_of_model_1;
      CASE list_of_values_2:
                              USE name_of_equation_2;
                              USE name_of_model_2;
      CASE list_of_values_nminus1:
                              USE name_of_equation_nminu1;
                              USE name_of_model_nminus1;
     OTHERWISE:
                              USE name_of_equation_n;
                              USE name_of_model_n;
END;
```

The following are important observations about the proposed implementation:

1 We are not dealing with conditional compilation. We create and have available the data structures for all of the variables and equations in each of the models. This is actually a requirement for the solution algorithms of conditional models. All the models and equations

quite arbitrary while the other is the vector of variables. Many commonly occurring problems have a more general form:

$$\begin{aligned} f^1(x)^T \cdot f^2(x) &= 0 \\ f^1(x) \geq 0 \quad f^2(x) &\geq 0 \end{aligned} \tag{4}$$

This problem is called the vertical nonlinear complementarity problem (Ferris and Pang, 1995). It is possible, of course, to have more than two vectors of functions in the above equations. Later, we will propose the representation of disjunctive sets of algebraic equations as complementarity problems. It is important to mention that we are aware of many disadvantages associated with this representation and, therefore, potential solutions to these problems will be mentioned as well. Our motivation is the fact that a complementarity formulation only requires the solution of a nonlinear system of equations, avoiding all of the complications encountered in the boundary crossing algorithm and the discrete decision making of a MINLP solution.

2.2.3 Solution of Conditional Models in an Equation-Based Modeling Environment

Finally in this section, we consider again the analysis of the previous implementations of conditional modeling to mention their status in the solution of these problems. Both the IF construct of *SpeedUp* and the CASE equation of *gPROMS* are used only to dynamically change the active set of equations during a dynamic simulation by evaluating the logical expression at any point in time. They assume, however, that the initial state of the system (and hence, the initial active set of equations) is known a priori. In other words, they do not address the problem of finding a consistent set of initial values that satisfy both the logical conditions and the active set of equations.

3 PROPOSED CONTRIBUTIONS

One of the goals of our research group has been to improve the ability to develop and solve process models. One of the results of this effort has been the development of ASCEND (Piela *et al.*, 1991). Westerberg *et al.* (1994) discuss the essential features of the current ASCEND system and present several ways in which we can improve it to solve larger models and to increase its scope as a modeling environment. In section 3.1 we will propose the implementation of tools which will allow ASCEND to represent conditional models efficiently. Section 3.2 is concerned

In the gradient analysis at boundaries, a descent direction for the objective functions (norm of the residuals of the equations) local to each neighboring subregion is required for convergence. This direction d will be given by:

$$\begin{aligned} \min_{\alpha} \quad & d^T \cdot d \\ \text{s.t.} \quad & d = N \cdot \alpha \\ & e^T \cdot \alpha = 1 \\ & \alpha \geq 0 \end{aligned} \tag{2}$$

where the matrix N is known and is used to store the vectors of the gradients of the objective functions local to each neighboring subregion which borders a boundary, and e is a vector with all elements equal to one. The vector α, whose elements sum to one, is used to generate the direction d as the linear combination of the gradients which yields a vector of smallest length. Termination occurs if a minimum value of zero is obtained for d, which means that the gradient vectors of N cannot be confined to one side of a half-plane.

2.2.2.2 Complementarity Formulation

The class of problems known as complementarity problems has become increasingly popular as a tool for addressing practical problems arising in mathematical programming, engineering, and the sciences (Billups, 1995; Ferris and Pang, 1995). Dirske and Ferris (1995) describe the standard forms for the different classes of complementarity problems and give examples of how to formulate many popular problems as mixed complementarity problems (MCP). Billups (1995) proposes strategies which enhance the robustness of Newton-based methods for their application in these problems.

In general, the nonlinear complementarity problem is expressed as the following set of equations and inequality constraints:

$$\begin{gathered} x^T \cdot f(x) = 0 \\ x \geq 0 \qquad f(x) \geq 0 \end{gathered} \tag{3}$$

There is certain lack of symmetry in the previous formulation. One of the functions is

problem that is often solvable as a relaxed LP. They consider the nonlinear case although no guarantee of convergence is given for it. Also, based in the works of Raman and Grossmann (1993, 1994), Turkay and Grossmann (1996) address the solution of disjunctive set of nonlinear programming problems. It is shown that the use of logic can improve the efficiency and robustness of MINLP algorithms on the solution of structural optimization problems.

It is worth summarizing some of the observations and conclusions about MINLP algorithms arisen from the work of Zaher (1995): 1) it can be expected that during the solution of a MINLP, the algorithm will jump from one subregion to another which is remote from it, leading to the poor initialization of some of the nonlinear subproblems, 2) the nonlinear subproblems insist in converging the equations of what might be an incorrect subregion and 3) this method employs a highly mathematical approach which remains as a rigorous general purpose algorithm designed for problems where discrete decision making cannot be avoided.

The following two sections summarize two alternative approaches to the solution of conditional models.

2.2.2.1 Boundary Crossing Algorithm

Zaher (1991) introduced the boundary crossing algorithm as an alternative to the MINLP formulation for the solution of conditional models. In this approach, the inequalities defining the domain of each set of variant equations are considered as logical conditions which, when satisfied, will identify a particular subregion and, if violated, will identify an alternative one. In that way, every equation $r_{ik}(x)=0$ constitutes a boundary which will separate the feasible region in two neighboring subregions. In general, identification of a subregion requires a unique combination of boolean values for the collection of all logical conditions. If l is the number of boundary equations, as many as 2^l subregions can be created.

The boundary crossing algorithm has been given in Zaher (1991). Basically, the algorithm applies a Newton step for all the points not residing at boundaries. If during the iterative process a boundary is reached, then a Newton step cannot be taken because the system is nondifferentiable at that point (different subregions have different set of equations). Therefore, a gradient analysis is performed instead.

problem can be decomposed in smaller partitions which can be individually solved in some precedence order. Structural analysis is an indispensable tool in an equation-based modeling environment, and as it will be shown, conditional models make the need for this tool an even stronger requirement.

In conditional models, the sparsity pattern is expected to change from one subregion to another. This implies that a consistent set of independent variables for one subregion may not be valid for another subregion. For the same reason, block partitioning of the equations for one region may not be possible using the equations of another.

Zaher (1995) addressed the structural analysis of conditional models. A necessary condition for structural consistency in conditional models is that each of the alternative sets of equations must be structurally consistent. Hence, consistency of a conditional model is assessed by finding at least one consistent partitioning of the variables (independent-dependent) such that output assignment of all of the equations in each subregion can be performed. This requirement makes the problem combinatorial, since we have to perform the analysis for all of the possible alternative set of equations.

Regarding the equation partitioning, it is recommended one to obtain a representative incidence matrix rather than attempt to incorrectly decouple a wrong set (not the solution set) of equations. This representative incidence of equations in a disjunctive statement can be achieved by 'OR'ing the incidences of the inequalities (or boundary expressions) with each locally defined equation. Then, additional incidences are obtained for each equation in a subregion one at a time by merging the incidences of the equation with those of exactly one equation from all of the other subregions. It should be pointed out that it is also possible and valid to obtain an individual block partitioning while analyzing any particular subregion, and that the previous algorithm was proposed by Zaher when seeking efficiency.

2.2.2 Solution Algorithms for Conditional Models

As mentioned earlier, conditional models can be formulated as mixed integer programming problems. In a very recent work, Grossmann and Turkay (1996) show that, in the case of linear equations, the solution of a conditional model can be found by solving a MILP

2 BACKGROUND

2.1 BUILDING A CONDITIONAL MODEL

Previous implementations of conditional statements in an equation-based modeling environment have been reported. One such mechanism is the IF-THEN-ELSE construct of *SpeedUp* described by Pantelides (1988):

```
IF logical_condition THEN
      equation1
ELSE
      equation2
ENDIF
```

where both the logical conditions and the equations are expressed in terms of the model variables. Such a construct defines two system states corresponding to the IF and the ELSE clauses respectively. Multiple states may be described by nesting several IF statements.

Similarly, Barton (1992) and Barton and Pantelides (1994) incorporated the CASE equation to *gPROMS*. The CASE equation is used to define both the appropriate modeling equations in each state and the logical conditions for transitions among states. It covers multiple states within only one statement and has the advantage of successfully representing irreversible discontinuities.

There is a major difficulty in both of the previous approaches as tools for conditional modeling. They only allow the substitution of a list of equations (or arrays of equations) for another. In an equation-based modeling environment in which concepts like hierarchy (building complex models from small models) and inheritance (adding new features to a model) are constantly in use, that represents a severe disadvantage in efficiency. Thinking of how many equations have to be written down when substituting some of the units of a superstructure makes obvious the lack of flexibility in the scope of these approaches. Later in this paper, we will show our extension to these modeling capabilities with the incorporation of the WHEN statement to the ASCEND modeling environment.

2.2 SOLVING A CONDITIONAL MODEL

2.2.1 Structural Analysis for Conditional Models

Before attempting to solve a model, a structural analysis has to be performed to determine if the formulation is well posed. Similarly, a partitioning algorithm is applied to find if the original

as determined by the simultaneously calculated value of the flow type indicator (Reynolds or Mach number) relative to some critical value.

While many currently available equation-based modeling systems have been reported in the literature, only a few of them have given attention to conditional models (Pantelides, 1988; Piela,1989; Barton, 1992). Several problems arise when building and testing models of complex processes in an equation-based environment (providing initial guesses, analyzing degrees of freedom, efficiently computing solutions, etc.). The combinatorial nature of a conditional model makes the solution of these problems even a more difficult task. In this work, we propose the incorporation of a series of tools which enables the user of an equation-based modeling environment with the capability of dealing with complex conditional models. It is important to distinguish between two well defined and separated (but complementary) goals of this research project:

1. Efficient representation of a conditional model in an equation-based modeling environment. This is intended to be independent of any particular application, or of any solver or algorithm used for finding a solution to the system of equations. Potential applications of this capability varies from the simple substitution of one equation for another (as in the case of the laminar-turbulent flow transition), to the substitution of a section of a chemical plant for another (as can be required while analyzing and initializing a superstructure).
2. Implementation and testing of alternative approaches to the solution of a conditional model. Modeling is a user dependent task. Frequently, the same problem is formulated differently from one modeler to another and, as a consequence, different approaches and techniques could be used to find a solution to it. Conditional models can be formulated as mixed-integer programming problems. See for example Grossmann and Turkay (1996). Besides the mixed-integer formulation, some alternative approaches exist which, in principle, could avoid the need of discrete decisions. Research in this work will focus in two of them: the boundary crossing algorithm (Zaher, 1991; Zaher, 1993; Zaher, 1995) and the complementarity formulation (Billups, 1995). In particular, we will propose an approach for the implementation of the boundary crossing algorithm in an equation-based environment, and an extension to the standard complementarity formulation for the representation of conditional models.

1 INTRODUCTION

The design and simulation of engineering processes require one to find the solution to a large system of linear and/or nonlinear equations. This work is particularly concerned with the representation and solution of conditional models. Conditional models exist when the defining equations of the system depend on where the model solution lies. A conditional model consists of a system of equations expressed by two sets: a globally defined invariant set of equations and a variant (or locally defined) set of conditional equations which are expressed as disjunctions. Basically, as shown by Zaher(1993), and Grossmann and Turkay (1996), this problem reduces to representing and finding the solution to the system of disjunctive equations:

$$
\begin{gathered}
g(x) = 0 \\
\bigvee_{i \in D_k} \begin{bmatrix} h_{ik}(x) = 0 \\ r_{ik}(x) \le 0 \end{bmatrix} \quad k \in K \\
x \in R^n
\end{gathered}
\tag{1}
$$

where $g(x)$ and $h_{ik}(x)$ represent the invariant and the variant sets of equations respectively, K represents the set of disjunctions and the index i is used to indicate the i-th term in each disjunction D_k. The equations $g(x)$ can be said to be defined over the entire feasible region, while the inequalities define the domain of validity of each variant set of equations $h_{ik}(x)$. In that way, each variant set of equations is confined to some subregion resulting from the dissection of the feasible region (Zaher, 1991). The solution of the system will be given by the vector x satisfying the invariant set $g(x)$ and exactly one set of equations for each of the disjunctions, providing that the corresponding set of inequalities is satisfied.

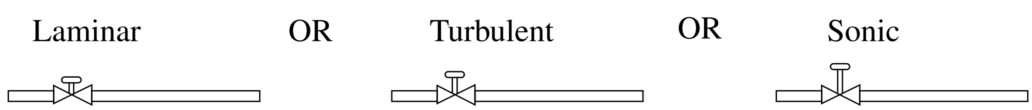


FIGURE 1 Fluid flow transition

Fluid transition represents an example commonly encountered in chemical engineering. This is illustrated in Figure 1 taken from the work of Zaher (1995). In this case, alternative equations for the transport properties and velocity bounds must be incorporated within the model

Representing, Analyzing and Solving Conditional Models in an Equation-Based Environment

by Vicente Rico-Ramírez

Advisor: Arthur W. Westerberg

Ph.D. Thesis Proposal

Department of Chemical Engineering

Carnegie Mellon University

Pittsburgh, PA

January 15, 1997

Abstract

Conditional models arise in chemical engineering when modeling systems involving physicochemical discontinuities, such as phase transitions. There are also cases where conditional cost functions are used in the optimization of process systems. Modeling tools for the efficient representation of conditional models in an equation-based environment are proposed in this paper. Also, some of the issues involved in the solution of these problems are discussed. The boundary crossing algorithm and the complementarity formulation are identified as alternatives to the MINLP algorithms for the solution of conditional models. In particular, we will propose an approach to the structural analysis of a conditional model, an extension to the standard complementarity formulation for the representation of conditional models, and an approach for the implementation of the boundary crossing algorithm in an equation-based environment.

More, J. J.; Global Methods for Nonlinear Complementarity Problems. Technical report, Argonne National Laboratory, Mathematics and Computer Science Division, April 1994.

Pantelides, C. C.; SPEEDUP-Recent Advances in Process Simulation. *Comput. Chem. Eng.*, 12(7):745–755, 1988.

Pantelides, C. C. and Barton, P. I.; Equation-Oriented Dynamic Simulation: Current Status and Future Perspectives. *Comput. Chem. Eng.*, 17S:263–285, 1993.

Piela, P., Epperly, T., Westerberg, K., Westerberg, A. W.; An Object-Oriented Computer Environment for Modeling and Analysis: The modeling language. *Comput. Chem. Eng.*, 15(1):53–72, 1991.

Piela, P.; ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis. PhD thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, April 1989.

Quesada, I. and Grossmann I. E.; A Global Optimization Algorithm for Linear Fractional and Bilinear Programs. *Journal of Global Optimization*, 6:39–76, 1995.

Raman, R. and Grossmann, I. E.; Symbolic Integration of Logic in Mixed-Integer Linear Programming Techniques for Process Synthesis. *Comput. Chem. Eng.*, 17(9):909–927, 1993.

Raman, R. and Grossmann, I. E.; Modeling and Computational Techniques for Logic Based Integer Programming. *Comput. Chem. Eng.*, 18(7):563–578, 1994.

Turkay, M. and Grossmann, I. E.; Logic-Based MINLP Algorithms for the Optimal Synthesis of Process Networks. Comput. Chem. Eng., 20(8):959–978, 1996.

Viswanathan, J. and Grossmann, I. E.; A Combined Penalty Function and Outer Approximation Method for MINLP Optimization. *Comput. Chem. Eng.*, 14(7):769–782, 1990.

Westerberg, A.W., Abbott, K. A., and Allan, B. A.; Plans for ASCEND IV: Our Next Generation Equational-Based Modeling Environment. Boston, Massachusetts, November 1994. AspenWorld'94.

Westerberg, A. W. and Director, S. W.; A Modified Least Square Algorithm for Solving Sparse n x n Sets of Nonlinear Equations. Technical report, Carnegie Mellon University, Engineering Design Research Center, DRC-06-5-79, January 1979.

Zaher, J. J.; Conditional Modeling. PhD thesis, Department of Chemical Engineering,Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1995.

Zaher, J. J.; Conditional Modeling in an Equation-Based Modeling Environment. The Annual AIChE National Meeting, 1991. paper 138c.

Zaher, J. J.; Conditional Programming. The Annual AIChE National Meeting, March 1993.

of this project. Some of the weaknesses and advantages of each of these approaches have been mentioned, but overcoming some of the difficulties and testing the approaches in large scale problems are issues still ahead as part of the future work. The following remain as some of the questions yet to be answered: *i)* How can the information provided by the structure of the problem help to simplifying the degrees of freedom analysis of a large conditional model? *ii)* How is the performance of a complementarity formulation against that of the boundary crossing algorithm in the solution of conditional models of practical size? *iii)* How is their performance in an optimization scheme? How does one overcome the difficulties incorporated by the bilinear terms in the complementarity formulation? Under what conditions can these approaches be considered a reliable alternative to MINLP algorithms in the simulation and optimization of large scale models?.

7 REFERENCES

Allen, G. L. and Westerberg, A. W.; Solution Procedures for Indexed Equation Sets: Structural Considerations. *AIChE Journal*, 22(3):549–558, May 1976.

Barton, P. I.; The Modeling and Simulation of Combined Discrete/Continuous Processes. PhD thesis, Department of Chemical Engineering, Imperial College of Science, Technology and Medicine, 1992.

Barton, P. I. and Pantelides, C. C.; Modeling of Combined Discrete/Continuous Processes. *AIChE Journal*, 40(6):966–979, June 1994.

Billups, S. C.; Algorithms for Complementarity Problems and Generalized Equations. PhD thesis, University of Wisconsin-Madison, August 1995.

Bullard, L. G. and Biegler, L. T.; Iterative Linear Programming Strategies for Constrained Simulation. *Comput. Chem. Eng.*, 15(4):239–254, 1991.

Bullard, L. G. and Biegler, L. T.; Iterated Linear Programming Strategies for Nonsmooth Simulation: Continuous and Mixed-Integer Approach. *Comput. Chem. Eng.*, 16(10), 1992

Dirske, S. P. and Ferris, M. C.; MCPLIB: A Collection of Nonlinear Mixed Complementarity Problems. Optimization Methods and Software, 5, 1995.

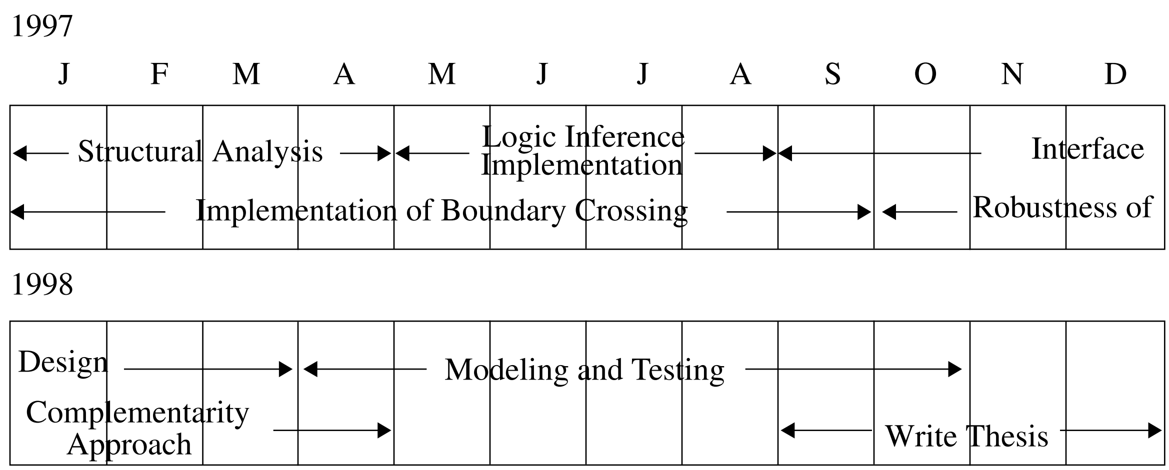
Ferris, M. C. and Pang, J. S. ; Engineering and Economic Applications of Complementarity Problems. Technical report, University of Colorado, 1995.

Grossmann, I. E. and Turkay, M.; Solution of Algebraic Systems of Disjunctive Equations. *Comput. Chem. Eng.*, 20:S339–44, 1996. Suppl. Part A.

An aspect that I have not mentioned, but that I anticipate it will be required, is the redesign of the existing interface to the ASCEND solver. I expect that the new interface tools not only will fit in with the current look of the ASCEND environment but also will provide the user with the capability of analyzing conditional models. An example of the new features that have to be added to the system is the capability of analyzing a conditional model at a boundary, where the neighboring sets of equations contain different structure.

The following is a tentative schedule for the remaining time of my research in this project.

Schedule of Proposed Work



6 CONCLUSIONS

There is a contribution to be made in the modeling world. Namely, the efficient representation of conditional models in an equation-based environment. We have described the modeling tools required to accomplish this, showing how the expressiveness of the modeling language increases with their incorporation. The computer implementation of these tools has been completed. There are surely other applications of the tools we propose that are being overlooked here. We hope to discover them.

We have also analyzed two alternative approaches to the solution of conditional models. The representation of conditional models as complementarity problems has been proposed and tested with small problems. The theoretical issues of the boundary crossing algorithm have been previously established, but an efficient computer implementation of it is considered to be a part

In the example of the linear mass balance, Turkay and Grossmann (1996) solve the problem by using a mixed-integer approach. The resulting MILP contains 18 binary variables, 66 continuous variables and 89 linear equations. In the complementarity formulation, the NLP contains 81 equations, 47 are linear, but the remaining 34 contain multiplications among slacks. The size of the problem is very similar, the difference will be in either using a branch and bound search in the MILP or dealing with the complementarity equations in the solution of one NLP.

We have thought that an interesting approach for dealing with conditional models would be to use the boundary crossing algorithm but representing some parts of the problem as a complementarity set of equations. In that way, we could obtain the advantages of both approaches. Once again, we consider this analysis to be part of the future work.

5 FUTURE WORK

In the previous two sections, we tried to emphasize the issues that still have to be addressed in both the representation and solution of conditional models; we also mentioned a description of a starting point for each of them. Making a brief summary list, we have:

1. Implementation of the boundary crossing algorithm. Issues inherently involved in this implementation (but with a much more general scope of application) include:
   i) Structural analysis in conditional models. This would include the idea of using the information provided by the structure of the problem in order to simplify the analysis, as outlined in section 3.2.1
   ii) Implementation of a logic inference algorithm.
2. Research in the field of the complementarity formulation for disjunctive systems of algebraic equations. Mainly, we need to investigate how to apply some of the ideas found in the literature to deal with the complementarity equations in an optimization mode and to improve the robustness of a NLP solver for the solution of this kind of equations.
3. Modeling and testing of large conditional models. From this analysis, we expect to be able to define the scope of application of the different approaches, to draw conclusions about the advantages and disadvantages of each of them and to combine them in order to get the best of both approaches.

be expected with problems of larger size.

Table 1: Examples of algebraic systems of disjunctive equations found in the literature

Example	Reference	Number of Equations	Number of Disjunctions	Number of Complementarity Equations	Iterations
Fluid transition	Zaher (1995)	7	1	1	11
Phase equilibria	Zaher (1995)	18	3	3	8
Heat exchanger	Zaher (1995)	56	2	4	12
Linear mass balance	Grossmann and Turkay (1996)	81	6	34	17
Pipeline network	Bullard and Biegler (1992)	250	38	76	27

The results of some of these examples are shown in Table 1. When possible, the initial values for the variables used were the same as those given in the reference (in some cases these values are not reported). To solve the examples, we used our ASCEND NLP solver which applies the modified Newton algorithm given in Westerberg and Director (1979). Since the hardware and the technique that we are using to get the solution is different, we are not comparing neither time nor number of iterations with other works. Still we consider it important to make remarks about some differences of the alternative approaches.

In the example of the pipeline network, the result of the complementarity representation is one NLP containing 250 equations, 76 of them containing 2 bilinear terms. In the same example, the number of boundaries in the boundary crossing algorithm would be 38 (that means 2^{38}=2.748779 x 10^{11} possible subregions) and the NLP to be solved in every subregion would contain 98 equations. The combinatorial complications present in examples like this impose a severe restriction to the boundary crossing algorithm. On the other hand, the reverse situation is also possible, and the boundary crossing may be clearly a better option than the complementarity formulation. For example, in a problem with only one disjunction, but 20 equations in each of the terms of the disjunction, we may not be able to solve a NLP in which 20 of the equations contain 20 bilinear terms each. However, the number of subregions in the boundary crossing algorithm would be only 2, and the possibility of applying this algorithm efficiently would be much greater.

```
MODEL flowsheet;
(* units *)
       f1                             IS_A cheap_feed;
       f2                             IS_A expensive_feed;
       c1,c2                          IS_A single_compressor;
       s1,s2                          IS_A staged_compressor;
       r1                             IS_A cheap_reactor;
       r2                             IS_A expensive_reactor;
       co1,co2                        IS_A cooler;
       h1,h2,h3                       IS_A heater;
       fl1                            IS_A flash;
       sp1                            IS_A splitter;
       m1                             IS_A mixer;
       v1                             IS_A expansion_valve;
(* boolean variables  *)
       select_feed1,   select_single1               IS_A boolean_var;
       select_cheapr1, select_single2               IS_A boolean_var;
(* define sets  *)
       m1.n_inputs :==2;
       sp1.n_outputs :== 2;
(* wire up flowsheet *)
       f1.stream, f2.stream, c1.input, s1.input     ARE_THE_SAME;
       c1.output, s1.output, m1.feed[2]             ARE_THE_SAME;
       m1.out,co1.input                             ARE_THE_SAME;
       co1.output, h1.input                         ARE_THE_SAME;
       h1.output, r1.input, r2.input                ARE_THE_SAME;
       r1.output, r2.output,v1.input                ARE_THE_SAME;
       v1.output,co2.input                          ARE_THE_SAME;
       co2.output, fl1.feed                         ARE_THE_SAME;
       fl1.liq, h2.input                            ARE_THE_SAME;
       fl1.vap, sp1.feed                            ARE_THE_SAME;
       sp1.out[1], h3.input                         ARE_THE_SAME;
       sp1.out[2],c2.input, s2.input                ARE_THE_SAME;
      c2.output, s2.output,m1.feed[1]               ARE_THE_SAME;
(* Conditional statements *)
       WHEN select_feed1
         CASE TRUE:
              USE f1;
         CASE FALSE:
              USE f2;
       END;
       WHEN select_single1
         CASE TRUE:
              USE c1;
         CASE FALSE:
              USE s1;
       END;
       WHEN select_cheapr1
         CASE TRUE:
              USE r1;
         CASE FALSE:
              USE r2;
       END;
       WHEN select_single2
         CASE TRUE:
              USE c2;
         CASE FALSE:
              USE s2;
       END;
END flowsheet;
```

FIGURE 6 ASCEND model for the superstructure of Figure 5

implementation of the WHEN statement and the logical relations. Its performance has been tested in several small problems. As an example, it was applied to the system presented in Figure 5 and Figure 6. The value of the four boolean variables will determine the structure of the problem to be solved. As mentioned above, the values of the boolean variables could be defined interactively by the user, but they also could be defined by some logic inference algorithm which would allow the automatic change of the structure of the problem. The NLP contains 137 invariant equations and 68 variant equations for a total of 205. The configuration defined by one of the feeds, two single-stage compressors and one of the reactors contains 169 equations, the 137 invariant and 32 active equations out of the 68 variant equations. Switching form one structure to another is done without the need of recompilation and, since to reconfigure the system requires only rebuilding several list of pointers, it is computationally very efficient.

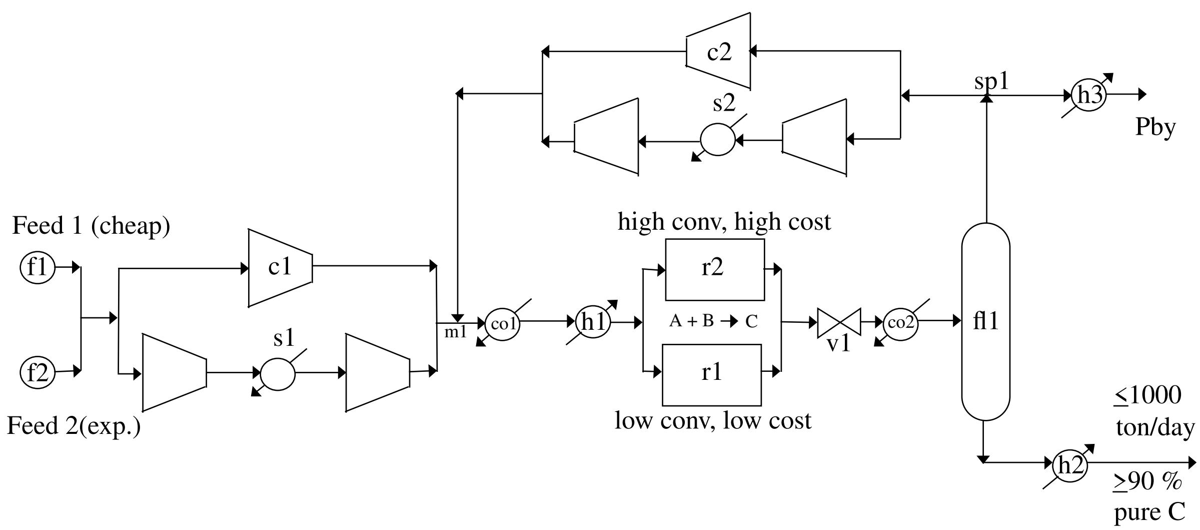


FIGURE 5 Superstructure taken from Turkay and Grossmann (1996).

4.2.2 Complementarity Formulation

Many examples of algebraic systems of disjunctive equations found in the literature have been represented and solved by using a complementarity formulation. In all the cases the results are encouraging. The solution of only one NLP is required and only a few iterations are used for convergence. However, it must be emphasized that, in all the cases, the number of equations in each disjunctive term (not the number of disjunctions) is very small; as a consequence, the complexity of the equations involving products among slack variables is not as bad as it can

Basically, a literature search has been conducted to find a possible solution for difficulty 3. To deal with nonconvexities in an optimization mode, Quesada and Grossman (1995) used linear relaxations based on the reformulation and linearization techniques for bilinear programming. Also, Viswanathan and Grossmann (1990) proposed augmenting the objective function with a penalty function that allows the violation of the linearizations of nonconvex constraints. A similar approach had been suggested by Bullard and Biegler (1991, 1992) in their iterated LP techniques for nonsmooth simulation. We are still not sure about how the application of these ideas could help to overcome the problems of having nonconvexities in the complementarity formulation. Research in this area is still ahead.

As mentioned before, the motivation for looking at a complementarity formulation is that only one NLP has to be solved. In general, we are presenting this approach as a very favorable alternative when the number of equations in each disjunctive term is small.

4 WORK TO DATE

4.1 MODELING

Both the WHEN statement and logical relations have already been incorporated to the ASCEND system as they were outlined above. We are currently testing this initial implementation, looking for improvements that could enhance its efficiency as modeling tools. To give an example of their application, we developed a simplified model for the superstructure given in Figure 5, taken from the work of Turkay and Grossmann (1996). In this example there are two alternative feedstocks, two possible choices of the reactor and two choices of the compression systems. Therefore, there are $2^4 = 16$ feasible configurations of the problem. All these 16 configurations are encapsulated in one ASCEND model containing 4 WHEN statements which depend on the value of 4 boolean variables. Figure 6 shows this model. The procedural section of the model and the model for each unit operation has been omitted for simplicity.

4.2 SOLUTION OF CONDITIONAL MODELS

4.2.1 Feeding a Conditional Model to an NLP solver

The mechanism suggested in section 3.2.2 has been implemented simultaneously with the

$$\begin{array}{l} h_1 \cdot h_2 + h_1 \cdot h_4 = 0 \\ h_3 \cdot h_2 + h_3 \cdot h_4 = 0 \end{array} \Rightarrow \begin{array}{l} h_1 \cdot (h_2 + h_4) = 0 \\ h_3 \cdot (h_2 + h_4) = 0 \end{array}$$

	h_1	h_2	h_3	h_4
1	$h_2 + h_4$	h_1	0	h_1
2	0	h_3	$h_2 + h_4$	h_3

FIGURE 4 Numerical singularities in factorable complementarity equations

the rows of the Jacobian corresponding to these equations. Note that the equations in Figure 4 contain the same four complementarity terms as the equations in (10), but they are grouped differently. In the case presented, if the solution of the problem is $h_2=h_4=0$, rows 1 and 2 of the Jacobian become numerically dependent as the Newton method approaches the solution (if $h_2+h_4 \sim 0$, we could get row 2 multiplying row 1 by the factor h_3/h_1). This is not the case in nonfactorable complementarity equations. There may be cases in which property 3 conflicts with property 4. Property 3 will have priority since it is what gives the simultaneous satisfaction of the equations in the same disjunctive term, a restriction that cannot be lost.

To get a formulation with all the properties outlined above is not a trivial exercise when the number of terms of the disjunction increases. However, for a disjunctive statement with two terms, the generalization to any number of equations is straightforward.

Many problems are incorporated with this kind of representation:

1. First of all, the problem grows quickly. If N is the number of equations in each term of the disjunction and D is the number of terms in the disjunction, the number of equations representing the disjunctive statement in the complementarity formulation is N(D+1).
2. The number of bilinear terms incorporated in each equation also grows with the number of equations in each term of the disjunction. The combinatorial nature of the problem is encapsulated here.
3. The performance of optimization techniques is badly affected by the introduction of nonconvexities (bilinear terms in the example) to the system of equations.
4. Numerical singularities can still arise for cases in which the solution resides on a boundary.

4. Every slack appears only once in each equation. There is no repeated slack index.

Posed in a formal way, given the disjunctive set of algebraic equations (two-term disjunction for simplicity)

$$\begin{bmatrix} f^1(x) = 0 \\ r(x) \le 0 \end{bmatrix} \vee \begin{bmatrix} f^2(x) = 0 \\ r(x) \ge 0 \end{bmatrix} \tag{11}$$

the problem is reformulated as

$$\begin{gathered} g(x,y) = \begin{pmatrix} f^1(x) - h_1 \\ f^2(x) - h_2 \\ r(x) - h_3 + h_4 \end{pmatrix} \\ g(x, y) = 0 \\ C(h_1, h_2, h_3, h_4) = 0 \\ h_i \ge 0 \qquad i \in [1 \ldots 4] \end{gathered} \tag{12}$$

where now *hi* represents a vector of slacks variables and $C(h_1,h_2,h_3,h_4)$ is used to represent the set of complementarity equations involving the elements of these vectors. Note that in the case of inequalities, the complementarity condition is expressed in terms of the slacks defined in the same equation. Also, we will require that the equations defined by $C(h_1,h_2,h_3,h_4)$ present the features outlined above for the equations given in (10). Property 1 is intended to avoid a change in the number of degrees of freedom, while property 3 is required to ensure the simultaneous satisfaction of all the equations in the same disjunctive term. Property 1 and 3 must always be satisfied. In our example, property 2 is only a consequence of 3 and 4 rather than a requirement, and may not be present in a general case.

Newton and quasi-Newton methods are the most popular techniques for the solution of NLP problems. Property 4 is intended to decrease the possibility of numerical singularities in the Jacobian which can occur during an iterative scheme based on these methods. If a repeated index occurs and, therefore, factorization is possible for some of the slacks, numerical singularities could arise as the iterative scheme approaches some solutions of the complementarity problem. This is illustrated in Figure 4, where we show a case of factorable complementarity equations and

$$Re - 64/f = h_1 \qquad Re - (0.206307/f)^4 = h_2$$
$$Re = 2100 + h_3 - h_4 \tag{7}$$

in terms of the slacks, the disjunctive statement becomes:

$$\begin{bmatrix} h_1 = 0 \\ h_3 = 0 \\ h_2, h_4 \geq 0 \end{bmatrix} \vee \begin{bmatrix} h_2 = 0 \\ h_4 = 0 \\ h_1, h_3 \geq 0 \end{bmatrix} \tag{8}$$

and the standard complementarity formulation is given by (7) and

$$\begin{aligned} h_1 \cdot h_2 &= 0 \\ h_3 \cdot h_4 &= 0 \\ h_i \geq 0 \qquad & i \in [1 \dots 4] \end{aligned} \tag{9}$$

It should be noted that the disjunctive statement requires either both h_1 and h_3 or both h_2 and h_4 simultaneously to be zero, and this is not a restriction in the complementarity formulation given by (9). The following formulation for disjunctive statements is proposed instead:

$$\begin{aligned} h_1 \cdot h_2 + h_3 \cdot h_4 &= 0 \\ h_1 \cdot h_4 + h_3 \cdot h_2 &= 0 \\ h_i \geq 0 \qquad & i \in [1 \dots 4] \end{aligned} \tag{10}$$

Since the slacks variables are all positive, the equations in (10) not only contains the complementarity condition given in (9) but also enforces simultaneous satisfaction of all the equations defined in the same term of the disjunction.

The complementarity equations given in (10) have the following properties:

1. The number of equations is equal to the number of equations in each term of the disjunction.
2. The number of terms in each equation is also equal to the number of equations in each term of the disjunction.
3. Every slack variable is multiplied by every slack in the other term of the disjunction.

implicitly or explicitly in such a CASE are set as active.

4. All the variables incident in the active set of equations are active.
5. The current problem to be solved consists of the active set of equations and variables.

The mechanism outline above is not precisely specific for the boundary crossing algorithm. For example, a branch and bound technique would also require the switching from one problem to another. Specifically for the boundary crossing algorithm, there is another major complication that must be addressed. While analyzing a boundary, the gradient step requires information about the equations in all of the subregions neighboring the boundary. That means the more than one set of equations has to be used simultaneously. We still do not address this and some other difficulties that we may not have discovered yet. However, based on previous programming exercises, we feel optimistic of being able to overcome any difficulty involved in this implementation in the time remaining for this research project.

3.2.3 Complementarity Representation of a Conditional Model

In this section, a complementarity formulation is proposed for the representation of conditional models. This formulation not only establishes the complementarity condition among alternative sets of equations but also enforces simultaneous satisfaction of all the equations in the solution set. The formulation proposed here is an extension of that presented by More(1994) who reformulates the nonlinear complementarity problem (3) as:

$$g(x,y) = \begin{pmatrix} f(x) - h \\ H \cdot x \end{pmatrix}$$

$$g(x, y) = 0 \tag{5}$$

$$x \geq 0 \qquad y \geq 0$$

where H is the diagonal matrix diag(h_i). The next simple example will show how a complementarity formulation should change for the case of disjunctive statements. Consider the flow transition given by:

$$\begin{bmatrix} Re = 64/f \\ Re \leq 2100 \end{bmatrix} \vee \begin{bmatrix} Re = (0.206307/f)^4 \\ Re \geq 2100 \end{bmatrix} \tag{6}$$

defining slack variables h for both the equalities and inequalities:

some parameters, such as cost factors, mass balance coefficients, power of a correlation, etc. In such a case, the need of a combinatorial structural analysis is eliminated.

2. The disjunctions of the problem are disjointed, and its structural analysis may be done independently. There are two cases in which this may happen *i)* the incidence pattern of the disjunction are totally decoupled, and *ii)* the existing coupling still allows the calculation of the variables output assigned to the disjunctive equations separately in some precedence order.

We have applied all these ideas for analyzing by hand small conditional problems, and we believe that their practical implementation is possible. A computer implementation which can take advantages of them is desirable.

3.2.2 Implementation of the Boundary Crossing Algorithm

A part of this project will consist in the implementation and testing of the boundary crossing algorithm. In general, it can be considered that both the Newton and gradients steps required in this algorithm are already well defined. We are looking, however, for ideas which could give the computational efficiency needed for its application in a general purpose environment like ASCEND.

We mentioned in section 3.1.1 that the implementation of the WHEN statement in ASCEND would allow us to have available the data structure for all the variables and equations of the conditional model. Therefore, the first step in the implementation of a conditional solver is to decide how the solver is going to be fed with the correct structure and how to change this structure efficiently in an iterative solution scheme. To perform this task, we will use the notion of active equation and active variable. By active we mean "it is part of the problem currently being solved". The following steps would provide a mechanism to select the structure of the problem:

1. Initially, all the equations resulting from the compilation are considered as active.
2. Then, all the equations which are implicitly or explicitly (inside models) stated in the CASE of a WHEN statement are set as inactive. The equations set as inactive in this step constitute the variant set of equations. All the equations which remain active constitute the invariant set of equations.
3. Analyze the WHEN statements. According to the current values of the variables on which each WHEN statement depends, determine which of its CASEs applies. The equations stated

crossing algorithm.

3.2 SOLVING CONDITIONAL MODELS

3.2.1 Structural Analysis

The analysis required for partitioning the variables and the equations in a conditional model was outlined in section 2.2.1. In our point of view, the most serious disadvantage of this analysis is the combinatorial nature of the search consistency algorithm, which requires the analysis of all the feasible subregions resulting from the dissection of the feasible region. We think that, in general, this analysis cannot be avoided. If we want to be sure about the consistency of the conditional model, we would have to pay the price. However, based on the work of Allen and Westerberg (1976), we also think that sometimes it is possible to take considerable advantage of the structure of the problem, reducing the effort required by such an analysis.

In chemical engineering design and simulation, systems modeled with indexed equations and variables occur very often. Typical examples are a distillation column (indexed by the number of stages) and systems of equations arising in the discretization of an initial value problem (indexed by the number of time steps). If we think of a problem in which a disjunction exists in each of the repeated structures, the combinatorial complexity in the structural analysis would be unmanageable. However, intuition suggests that, in this kind of problem, the degrees of freedom analysis should not be affected by the number of repeated structures and, therefore, the analysis could be simplified. In his work, Allen used a representative incidence matrix to perform a systematic analysis of indexed equation sets. Again, this could not be applied to an arbitrary system of disjunctive equations, but if we know a priori that our problem contains an indexed repeated structure, then we may take advantage of it.

We have also observed special features of some problems which can contribute to simplifying the analysis. Even though they represent very particular cases, if found alone or in any combination, a simple checking in the computational implementation of the algorithm could give us saves in time and in the computational effort needed to perform the analysis. These are:

1. The incidence pattern of the different terms of the disjunction is the same. It happens sometimes that the alternative equations are only different because the difference in value of

(a)

```
laminar     IS_A boolean;
Re,f,k      IS_A factor;

CONDITIONAL
      condition: Re <= 2100;
END;

laminar == SATISFIED(condition);

eq1: Re = 64/f;
eq2: Re = (0.206307/f)^4;

WHEN laminar:
      CASE TRUE:
            USE eq1;
      CASE FALSE:
            USE eq2;
END;
```

(b)

```
valve_open      IS_A boolean;
pump_on         IS_A boolean;
full_tank       IS_A boolean;

valve_open == pump_on AND NOT(full_tank);
```

FIGURE 3 Examples of the syntax for the implementation of logical relations

AND, OR, and NOT are allowed. Equality in a logical equation can also be interpreted as an *if and only if* implication between two logical terms expressed in clausal form. Also, it should be noted that we can express any logical clause using the proposed syntax by simply writing the clause in one of the terms, and the constant boolean value TRUE in the other term of the logical equality. Example (a) is a more complete version of the model presented in Figure 2. In this case it can be noted that, when the value of a boolean variable depends on the truth value of a real expression, the real expression is defined and labeled inside the CONDITIONAL statement; then the logical operator SATISFIED will give the truth value of such an expression. If the truth value of the real expression is TRUE, the result of SATISFIED(label of expression) will be TRUE. The objective of the operator SATISFIED is twofold: 1) it makes the life of the compiler easier, since the logical expressions are decoupled from the real expressions on which they depend, and 2) the use of this syntax forces the user to define a boolean variable (laminar in the example) which will have the same truth value as the real expression. That is useful for browsing purposes.

It is important to emphasize that, by checking the value of the appropriate boolean variables after each iteration, the incorporation of logical relations will allow, if required, automatic change of the structure of the problem in an iterative scheme such as the boundary

the value of the symbol method. Once a configuration has been select, it will be kept unless the user decides to change it. Note that the user does not have to recompile the model to make the switching, since the data structure for both of the problems is available and, therefore, the reconfiguration of the system can be done automatically by simply changing the value of the conditional variable. An example of this application would be the synthesis of process networks.

On the other hand, in case (a) we cannot expect the boolean value of the variable laminar to be a user decision. Its value will depend on the value of the Reynolds number which is an unknown in the problem. Actually, the value of the variable laminar will be the truth value of the expression $Re \leq 2100$. Here we have a conditional model. In an iterative solution scheme, we will expect that the value of the boolean variable laminar will change, and so will the structure of the system of equations that we have to solve.

3.1.2 Incorporation of Logical Relations

Motivated by some recent applications of logic in the solution of conditional models (Raman and Grossmann, 1994; Turkay and Grossmann, 1996) and trying to expand the ASCEND capabilities for modeling, an implementation of logical relations as a modeling tool is proposed. To our knowledge, no other general purpose equation-based modeling environment provides this capability.

In the implementation of logical relations, the equation based approach is going to be maintained. That is, the user will state the logical relations that must be true at the solution to the problem but not how to solve them. Each logical equation will have a residual attached to it. This residual will indicate if the expression is satisfied or not. Therefore, this incorporation also means that we need to provide a solver which knows how to deal with logical relations. The work of Raman and Grossman (1993) can be used as a starting point for such a solver implementation. They used formal procedures for performing logic inference depending on the way in which the logical relations are present (CNF or DNF).

Figure 3 illustrates two examples of the syntax proposed for the implementation of logical relations. The symbol double equality will indicate that we have a logical relation and will separate the two logical terms of the relation. In each of the two terms, logical operators such as

whose name is given in each of the cases should be declared inside the model which contains the WHEN statement.

2 The variables in the list of variables can be of any type among boolean, integer or symbol or any combination of them. That is, we are not limited to the use of boolean variables. Obviously, The list of values in each case must be in agreement with the list of variables in the number of elements and type of each element.

3 Names of arrays of models or equations are also allowed inside the scope of each case.

The proposed extension (allowing the user to define the domain of validity of both models and equations inside the cases of a WHEN statement) enormously increases the scope of modeling in an equation based modeling environment. Of course, this implementation may be possible in other modeling environments which support object oriented concepts, but we have proposed it as an extension to the ASCEND modeling language because ASCEND is the best one locally understood. Figure 2 shows two incomplete ASCEND models in which the WHEN statement is used, each in a different manner.

```
      (a)
laminar      IS_A boolean;
Re,f,k       IS_A factor;

eq1: Re = 64/f;
eq2: Re = (0.206307/f)^4;

WHEN laminar:
      CASE TRUE:
            USE eq1;
      CASE FALSE:
            USE eq2;
END;
```

```
                  (b)
method              IS_A symbol;
simplified_flash    IS_A VLE_flash;
rigorous_flash      IS_S td_VLE_flash;

WHEN method
      CASE 'rigorous':
            USE rigorous_flash;
      CASE 'simplified':
            USE simplified_flash;
END;
```

FIGURE 2 Two different applications of the WHEN statement.

In case (b) the value of the symbol method is used only to select between two alternative configurations of the problem, a flash calculation assuming constant relative volatility (VLE_flash) or a flash calculation using a more rigorous thermodynamic calculation (td_VLE_flash). The method that is going to be used and, therefore, the value of the symbol method, is a user decision. We can expect that the user will select the simplified model looking for good initial values of the variables and then he will switch to the rigorous model by changing

with the solution of these problems. Some issues involved in the implementation of the boundary crossing algorithm will be briefly explained, and an extension of a complementarity formulation to represent conditional models will be proposed.

3.1 MODELING

In this section, an extension to the ASCEND modeling language is proposed. Essentially, it consists of implementing the WHEN statement and incorporating logical relations into the system.

3.1.1 The WHEN Statement

Originally, the syntax for incorporating conditional dependence of some equations of the model in an equation-based environment was suggested by Piela (1989). That syntax is very similar to the CASE equation of *gPROMS* and suffers the same limitations. Instead, we want to take advantage of the fact that ASCEND is based on object oriented concepts where model definitions can contain parts that contain parts to any level. Furthermore, in ASCEND, a simple relation is treated as an object by itself and can have a name. Based on these ideas, the proposed syntax for the WHEN statement is:

```
WHEN list_of_variables
      CASE list_of_values_1:
                              USE name_of_equation_1;
                              USE name_of_model_1;
      CASE list_of_values_2:
                              USE name_of_equation_2;
                              USE name_of_model_2;
      CASE list_of_values_nminus1:
                              USE name_of_equation_nminu1;
                              USE name_of_model_nminus1;
     OTHERWISE:
                              USE name_of_equation_n;
                              USE name_of_model_n;
END;
```

The following are important observations about the proposed implementation:

1 We are not dealing with conditional compilation. We create and have available the data structures for all of the variables and equations in each of the models. This is actually a requirement for the solution algorithms of conditional models. All the models and equations

quite arbitrary while the other is the vector of variables. Many commonly occurring problems have a more general form:

$$
\begin{aligned}
f^1(x)^T \cdot f^2(x) &= 0 \\
f^1(x) \geq 0 \quad f^2(x) &\geq 0
\end{aligned}
\tag{4}
$$

This problem is called the vertical nonlinear complementarity problem (Ferris and Pang, 1995). It is possible, of course, to have more than two vectors of functions in the above equations. Later, we will propose the representation of disjunctive sets of algebraic equations as complementarity problems. It is important to mention that we are aware of many disadvantages associated with this representation and, therefore, potential solutions to these problems will be mentioned as well. Our motivation is the fact that a complementarity formulation only requires the solution of a nonlinear system of equations, avoiding all of the complications encountered in the boundary crossing algorithm and the discrete decision making of a MINLP solution.

2.2.3 Solution of Conditional Models in an Equation-Based Modeling Environment

Finally in this section, we consider again the analysis of the previous implementations of conditional modeling to mention their status in the solution of these problems. Both the IF construct of *SpeedUp* and the CASE equation of *gPROMS* are used only to dynamically change the active set of equations during a dynamic simulation by evaluating the logical expression at any point in time. They assume, however, that the initial state of the system (and hence, the initial active set of equations) is known a priori. In other words, they do not address the problem of finding a consistent set of initial values that satisfy both the logical conditions and the active set of equations.

3 PROPOSED CONTRIBUTIONS

One of the goals of our research group has been to improve the ability to develop and solve process models. One of the results of this effort has been the development of ASCEND (Piela *et al*., 1991). Westerberg *et al.* (1994) discuss the essential features of the current ASCEND system and present several ways in which we can improve it to solve larger models and to increase its scope as a modeling environment. In section 3.1 we will propose the implementation of tools which will allow ASCEND to represent conditional models efficiently. Section 3.2 is concerned

In the gradient analysis at boundaries, a descent direction for the objective functions (norm of the residuals of the equations) local to each neighboring subregion is required for convergence. This direction d will be given by:

$$\begin{aligned} \min_{\alpha} \quad & d^T \cdot d \\ \text{s.t.} \quad & d = N \cdot \alpha \\ & e^T \cdot \alpha = 1 \\ & \alpha \geq 0 \end{aligned} \tag{2}$$

where the matrix N is known and is used to store the vectors of the gradients of the objective functions local to each neighboring subregion which borders a boundary, and e is a vector with all elements equal to one. The vector α, whose elements sum to one, is used to generate the direction d as the linear combination of the gradients which yields a vector of smallest length. Termination occurs if a minimum value of zero is obtained for d, which means that the gradient vectors of N cannot be confined to one side of a half-plane.

2.2.2.2 Complementarity Formulation

The class of problems known as complementarity problems has become increasingly popular as a tool for addressing practical problems arising in mathematical programming, engineering, and the sciences (Billups, 1995; Ferris and Pang, 1995). Dirske and Ferris (1995) describe the standard forms for the different classes of complementarity problems and give examples of how to formulate many popular problems as mixed complementarity problems (MCP). Billups (1995) proposes strategies which enhance the robustness of Newton-based methods for their application in these problems.

In general, the nonlinear complementarity problem is expressed as the following set of equations and inequality constraints:

$$\begin{gathered} x^T \cdot f(x) = 0 \\ x \geq 0 \qquad f(x) \geq 0 \end{gathered} \tag{3}$$

There is certain lack of symmetry in the previous formulation. One of the functions is

problem that is often solvable as a relaxed LP. They consider the nonlinear case although no guarantee of convergence is given for it. Also, based in the works of Raman and Grossmann (1993, 1994), Turkay and Grossmann (1996) address the solution of disjunctive set of nonlinear programming problems. It is shown that the use of logic can improve the efficiency and robustness of MINLP algorithms on the solution of structural optimization problems.

It is worth summarizing some of the observations and conclusions about MINLP algorithms arisen from the work of Zaher (1995): 1) it can be expected that during the solution of a MINLP, the algorithm will jump from one subregion to another which is remote from it, leading to the poor initialization of some of the nonlinear subproblems, 2) the nonlinear subproblems insist in converging the equations of what might be an incorrect subregion and 3) this method employs a highly mathematical approach which remains as a rigorous general purpose algorithm designed for problems where discrete decision making cannot be avoided.

The following two sections summarize two alternative approaches to the solution of conditional models.

2.2.2.1 Boundary Crossing Algorithm

Zaher (1991) introduced the boundary crossing algorithm as an alternative to the MINLP formulation for the solution of conditional models. In this approach, the inequalities defining the domain of each set of variant equations are considered as logical conditions which, when satisfied, will identify a particular subregion and, if violated, will identify an alternative one. In that way, every equation $r_{ik}(x)=0$ constitutes a boundary which will separate the feasible region in two neighboring subregions. In general, identification of a subregion requires a unique combination of boolean values for the collection of all logical conditions. If l is the number of boundary equations, as many as 2^l subregions can be created.

The boundary crossing algorithm has been given in Zaher (1991). Basically, the algorithm applies a Newton step for all the points not residing at boundaries. If during the iterative process a boundary is reached, then a Newton step cannot be taken because the system is nondifferentiable at that point (different subregions have different set of equations). Therefore, a gradient analysis is performed instead.

problem can be decomposed in smaller partitions which can be individually solved in some precedence order. Structural analysis is an indispensable tool in an equation-based modeling environment, and as it will be shown, conditional models make the need for this tool an even stronger requirement.

In conditional models, the sparsity pattern is expected to change from one subregion to another. This implies that a consistent set of independent variables for one subregion may not be valid for another subregion. For the same reason, block partitioning of the equations for one region may not be possible using the equations of another.

Zaher (1995) addressed the structural analysis of conditional models. A necessary condition for structural consistency in conditional models is that each of the alternative sets of equations must be structurally consistent. Hence, consistency of a conditional model is assessed by finding at least one consistent partitioning of the variables (independent-dependent) such that output assignment of all of the equations in each subregion can be performed. This requirement makes the problem combinatorial, since we have to perform the analysis for all of the possible alternative set of equations.

Regarding the equation partitioning, it is recommended one to obtain a representative incidence matrix rather than attempt to incorrectly decouple a wrong set (not the solution set) of equations. This representative incidence of equations in a disjunctive statement can be achieved by ’OR’ing the incidences of the inequalities (or boundary expressions) with each locally defined equation. Then, additional incidences are obtained for each equation in a subregion one at a time by merging the incidences of the equation with those of exactly one equation from all of the other subregions. It should be pointed out that it is also possible and valid to obtain an individual block partitioning while analyzing any particular subregion, and that the previous algorithm was proposed by Zaher when seeking efficiency.

2.2.2 Solution Algorithms for Conditional Models

As mentioned earlier, conditional models can be formulated as mixed integer programming problems. In a very recent work, Grossmann and Turkay (1996) show that, in the case of linear equations, the solution of a conditional model can be found by solving a MILP

2 BACKGROUND

2.1 BUILDING A CONDITIONAL MODEL

Previous implementations of conditional statements in an equation-based modeling environment have been reported. One such mechanism is the IF-THEN-ELSE construct of *SpeedUp* described by Pantelides (1988):

```
IF logical_condition THEN
      equation1
ELSE
      equation2
ENDIF
```

where both the logical conditions and the equations are expressed in terms of the model variables. Such a construct defines two system states corresponding to the IF and the ELSE clauses respectively. Multiple states may be described by nesting several IF statements.

Similarly, Barton (1992) and Barton and Pantelides (1994) incorporated the CASE equation to *gPROMS*. The CASE equation is used to define both the appropriate modeling equations in each state and the logical conditions for transitions among states. It covers multiple states within only one statement and has the advantage of successfully representing irreversible discontinuities.

There is a major difficulty in both of the previous approaches as tools for conditional modeling. They only allow the substitution of a list of equations (or arrays of equations) for another. In an equation-based modeling environment in which concepts like hierarchy (building complex models from small models) and inheritance (adding new features to a model) are constantly in use, that represents a severe disadvantage in efficiency. Thinking of how many equations have to be written down when substituting some of the units of a superstructure makes obvious the lack of flexibility in the scope of these approaches. Later in this paper, we will show our extension to these modeling capabilities with the incorporation of the WHEN statement to the ASCEND modeling environment.

2.2 SOLVING A CONDITIONAL MODEL

2.2.1 Structural Analysis for Conditional Models

Before attempting to solve a model, a structural analysis has to be performed to determine if the formulation is well posed. Similarly, a partitioning algorithm is applied to find if the original

as determined by the simultaneously calculated value of the flow type indicator (Reynolds or Mach number) relative to some critical value.

While many currently available equation-based modeling systems have been reported in the literature, only a few of them have given attention to conditional models (Pantelides, 1988; Piela,1989; Barton, 1992). Several problems arise when building and testing models of complex processes in an equation-based environment (providing initial guesses, analyzing degrees of freedom, efficiently computing solutions, etc.). The combinatorial nature of a conditional model makes the solution of these problems even a more difficult task. In this work, we propose the incorporation of a series of tools which enables the user of an equation-based modeling environment with the capability of dealing with complex conditional models. It is important to distinguish between two well defined and separated (but complementary) goals of this research project:

1. Efficient representation of a conditional model in an equation-based modeling environment. This is intended to be independent of any particular application, or of any solver or algorithm used for finding a solution to the system of equations. Potential applications of this capability varies from the simple substitution of one equation for another (as in the case of the laminar-turbulent flow transition), to the substitution of a section of a chemical plant for another (as can be required while analyzing and initializing a superstructure).
2. Implementation and testing of alternative approaches to the solution of a conditional model. Modeling is a user dependent task. Frequently, the same problem is formulated differently from one modeler to another and, as a consequence, different approaches and techniques could be used to find a solution to it. Conditional models can be formulated as mixed-integer programming problems. See for example Grossmann and Turkay (1996). Besides the mixed-integer formulation, some alternative approaches exist which, in principle, could avoid the need of discrete decisions. Research in this work will focus in two of them: the boundary crossing algorithm (Zaher, 1991; Zaher, 1993; Zaher, 1995) and the complementarity formulation (Billups, 1995). In particular, we will propose an approach for the implementation of the boundary crossing algorithm in an equation-based environment, and an extension to the standard complementarity formulation for the representation of conditional models.

1 INTRODUCTION

The design and simulation of engineering processes require one to find the solution to a large system of linear and/or nonlinear equations. This work is particularly concerned with the representation and solution of conditional models. Conditional models exist when the defining equations of the system depend on where the model solution lies. A conditional model consists of a system of equations expressed by two sets: a globally defined invariant set of equations and a variant (or locally defined) set of conditional equations which are expressed as disjunctions. Basically, as shown by Zaher(1993), and Grossmann and Turkay (1996), this problem reduces to representing and finding the solution to the system of disjunctive equations:

$$g(x) = 0$$

$$\bigvee_{i \in D_k} \begin{bmatrix} h_{ik}(x) = 0 \\ r_{ik}(x) \leq 0 \end{bmatrix} \quad k \in K \tag{1}$$

$$x \in R^n$$

where $g(x)$ and $h_{ik}(x)$ represent the invariant and the variant sets of equations respectively, K represents the set of disjunctions and the index i is used to indicate the i-th term in each disjunction D_k. The equations $g(x)$ can be said to be defined over the entire feasible region, while the inequalities define the domain of validity of each variant set of equations $h_{ik}(x)$. In that way, each variant set of equations is confined to some subregion resulting from the dissection of the feasible region (Zaher, 1991). The solution of the system will be given by the vector x satisfying the invariant set $g(x)$ and exactly one set of equations for each of the disjunctions, providing that the corresponding set of inequalities is satisfied.

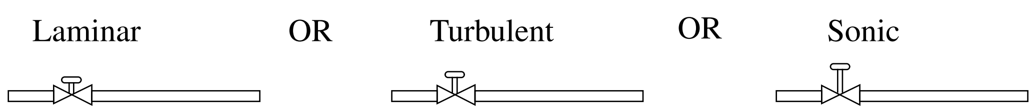


FIGURE 1 Fluid flow transition

Fluid transition represents an example commonly encountered in chemical engineering. This is illustrated in Figure 1 taken from the work of Zaher (1995). In this case, alternative equations for the transport properties and velocity bounds must be incorporated within the model

Representing, Analyzing and Solving Conditional Models in an Equation-Based Environment

by Vicente Rico-Ramírez

Advisor: Arthur W. Westerberg

Ph.D. Thesis Proposal

Department of Chemical Engineering

Carnegie Mellon University

Pittsburgh, PA

January 15, 1997

Abstract

Conditional models arise in chemical engineering when modeling systems involving physicochemical discontinuities, such as phase transitions. There are also cases where conditional cost functions are used in the optimization of process systems. Modeling tools for the efficient representation of conditional models in an equation-based environment are proposed in this paper. Also, some of the issues involved in the solution of these problems are discussed. The boundary crossing algorithm and the complementarity formulation are identified as alternatives to the MINLP algorithms for the solution of conditional models. In particular, we will propose an approach to the structural analysis of a conditional model, an extension to the standard complementarity formulation for the representation of conditional models, and an approach for the implementation of the boundary crossing algorithm in an equation-based environment.

More, J. J.; Global Methods for Nonlinear Complementarity Problems. Technical report, Argonne National Laboratory, Mathematics and Computer Science Division, April 1994.

Pantelides, C. C.; SPEEDUP-Recent Advances in Process Simulation. *Comput. Chem. Eng.*, 12(7):745–755, 1988.

Pantelides, C. C. and Barton, P. I.; Equation-Oriented Dynamic Simulation: Current Status and Future Perspectives. *Comput. Chem. Eng.*, 17S:263–285, 1993.

Piela, P., Epperly, T., Westerberg, K., Westerberg, A. W.; An Object-Oriented Computer Environment for Modeling and Analysis: The modeling language. *Comput. Chem. Eng.*, 15(1):53–72, 1991.

Piela, P.; ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis. PhD thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, April 1989.

Quesada, I. and Grossmann I. E.; A Global Optimization Algorithm for Linear Fractional and Bilinear Programs. *Journal of Global Optimization*, 6:39–76, 1995.

Raman, R. and Grossmann, I. E.; Symbolic Integration of Logic in Mixed-Integer Linear Programming Techniques for Process Synthesis. *Comput. Chem. Eng.*, 17(9):909–927, 1993.

Raman, R. and Grossmann, I. E.; Modeling and Computational Techniques for Logic Based Integer Programming. *Comput. Chem. Eng.*, 18(7):563–578, 1994.

Turkay, M. and Grossmann, I. E.; Logic-Based MINLP Algorithms for the Optimal Synthesis of Process Networks. Comput. Chem. Eng., 20(8):959–978, 1996.

Viswanathan, J. and Grossmann, I. E.; A Combined Penalty Function and Outer Approximation Method for MINLP Optimization. *Comput. Chem. Eng.*, 14(7):769–782, 1990.

Westerberg, A.W., Abbott, K. A., and Allan, B. A.; Plans for ASCEND IV: Our Next Generation Equational-Based Modeling Environment. Boston, Massachusetts, November 1994. AspenWorld'94.

Westerberg, A. W. and Director, S. W.; A Modified Least Square Algorithm for Solving Sparse n x n Sets of Nonlinear Equations. Technical report, Carnegie Mellon University, Engineering Design Research Center, DRC-06-5-79, January 1979.

Zaher, J. J.; Conditional Modeling. PhD thesis, Department of Chemical Engineering,Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1995.

Zaher, J. J.; Conditional Modeling in an Equation-Based Modeling Environment. The Annual AIChE National Meeting, 1991. paper 138c.

Zaher, J. J.; Conditional Programming. The Annual AIChE National Meeting, March 1993.

of this project. Some of the weaknesses and advantages of each of these approaches have been mentioned, but overcoming some of the difficulties and testing the approaches in large scale problems are issues still ahead as part of the future work. The following remain as some of the questions yet to be answered: *i)* How can the information provided by the structure of the problem help to simplifying the degrees of freedom analysis of a large conditional model? *ii)* How is the performance of a complementarity formulation against that of the boundary crossing algorithm in the solution of conditional models of practical size? *iii)* How is their performance in an optimization scheme? How does one overcome the difficulties incorporated by the bilinear terms in the complementarity formulation? Under what conditions can these approaches be considered a reliable alternative to MINLP algorithms in the simulation and optimization of large scale models?.

7 REFERENCES

Allen, G. L. and Westerberg, A. W.; Solution Procedures for Indexed Equation Sets: Structural Considerations. *AIChE Journal*, 22(3):549–558, May 1976.

Barton, P. I.; The Modeling and Simulation of Combined Discrete/Continuous Processes. PhD thesis, Department of Chemical Engineering, Imperial College of Science, Technology and Medicine, 1992.

Barton, P. I. and Pantelides, C. C.; Modeling of Combined Discrete/Continuous Processes. *AIChE Journal*, 40(6):966–979, June 1994.

Billups, S. C.; Algorithms for Complementarity Problems and Generalized Equations. PhD thesis, University of Wisconsin-Madison, August 1995.

Bullard, L. G. and Biegler, L. T.; Iterative Linear Programming Strategies for Constrained Simulation. *Comput. Chem. Eng.*, 15(4):239–254, 1991.

Bullard, L. G. and Biegler, L. T.; Iterated Linear Programming Strategies for Nonsmooth Simulation: Continuous and Mixed-Integer Approach. *Comput. Chem. Eng.*, 16(10), 1992

Dirske, S. P. and Ferris, M. C.; MCPLIB: A Collection of Nonlinear Mixed Complementarity Problems. Optimization Methods and Software, 5, 1995.

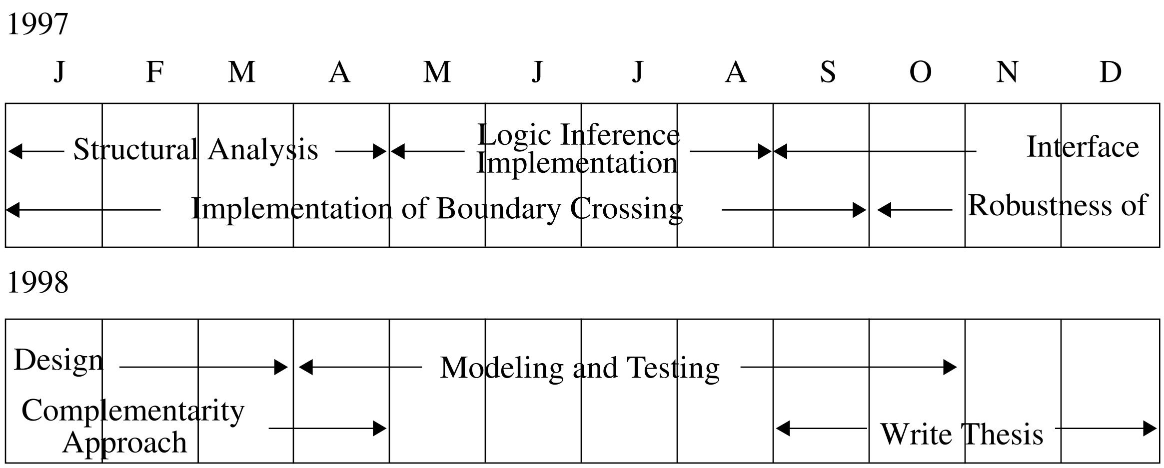
Ferris, M. C. and Pang, J. S. ; Engineering and Economic Applications of Complementarity Problems. Technical report, University of Colorado, 1995.

Grossmann, I. E. and Turkay, M.; Solution of Algebraic Systems of Disjunctive Equations. *Comput. Chem. Eng.*, 20:S339–44, 1996. Suppl. Part A.

An aspect that I have not mentioned, but that I anticipate it will be required, is the redesign of the existing interface to the ASCEND solver. I expect that the new interface tools not only will fit in with the current look of the ASCEND environment but also will provide the user with the capability of analyzing conditional models. An example of the new features that have to be added to the system is the capability of analyzing a conditional model at a boundary, where the neighboring sets of equations contain different structure.

The following is a tentative schedule for the remaining time of my research in this project.

Schedule of Proposed Work



6 CONCLUSIONS

There is a contribution to be made in the modeling world. Namely, the efficient representation of conditional models in an equation-based environment. We have described the modeling tools required to accomplish this, showing how the expressiveness of the modeling language increases with their incorporation. The computer implementation of these tools has been completed. There are surely other applications of the tools we propose that are being overlooked here. We hope to discover them.

We have also analyzed two alternative approaches to the solution of conditional models. The representation of conditional models as complementarity problems has been proposed and tested with small problems. The theoretical issues of the boundary crossing algorithm have been previously established, but an efficient computer implementation of it is considered to be a part

In the example of the linear mass balance, Turkay and Grossmann (1996) solve the problem by using a mixed-integer approach. The resulting MILP contains 18 binary variables, 66 continuous variables and 89 linear equations. In the complementarity formulation, the NLP contains 81 equations, 47 are linear, but the remaining 34 contain multiplications among slacks. The size of the problem is very similar, the difference will be in either using a branch and bound search in the MILP or dealing with the complementarity equations in the solution of one NLP.

We have thought that an interesting approach for dealing with conditional models would be to use the boundary crossing algorithm but representing some parts of the problem as a complementarity set of equations. In that way, we could obtain the advantages of both approaches. Once again, we consider this analysis to be part of the future work.

5 FUTURE WORK

In the previous two sections, we tried to emphasize the issues that still have to be addressed in both the representation and solution of conditional models; we also mentioned a description of a starting point for each of them. Making a brief summary list, we have:

1. Implementation of the boundary crossing algorithm. Issues inherently involved in this implementation (but with a much more general scope of application) include:
   i) Structural analysis in conditional models. This would include the idea of using the information provided by the structure of the problem in order to simplify the analysis, as outlined in section 3.2.1
   ii) Implementation of a logic inference algorithm.
2. Research in the field of the complementarity formulation for disjunctive systems of algebraic equations. Mainly, we need to investigate how to apply some of the ideas found in the literature to deal with the complementarity equations in an optimization mode and to improve the robustness of a NLP solver for the solution of this kind of equations.
3. Modeling and testing of large conditional models. From this analysis, we expect to be able to define the scope of application of the different approaches, to draw conclusions about the advantages and disadvantages of each of them and to combine them in order to get the best of both approaches.

be expected with problems of larger size.

Table 1: Examples of algebraic systems of disjunctive equations found in the literature

Example	Reference	Number of Equations	Number of Disjunctions	Number of Complementarity Equations	Iterations
Fluid transition	Zaher (1995)	7	1	1	11
Phase equilibria	Zaher (1995)	18	3	3	8
Heat exchanger	Zaher (1995)	56	2	4	12
Linear mass balance	Grossmann and Turkay (1996)	81	6	34	17
Pipeline network	Bullard and Biegler (1992)	250	38	76	27

The results of some of these examples are shown in Table 1. When possible, the initial values for the variables used were the same as those given in the reference (in some cases these values are not reported). To solve the examples, we used our ASCEND NLP solver which applies the modified Newton algorithm given in Westerberg and Director (1979). Since the hardware and the technique that we are using to get the solution is different, we are not comparing neither time nor number of iterations with other works. Still we consider it important to make remarks about some differences of the alternative approaches.

In the example of the pipeline network, the result of the complementarity representation is one NLP containing 250 equations, 76 of them containing 2 bilinear terms. In the same example, the number of boundaries in the boundary crossing algorithm would be 38 (that means 2^{38}=2.748779 x 10^{11} possible subregions) and the NLP to be solved in every subregion would contain 98 equations. The combinatorial complications present in examples like this impose a severe restriction to the boundary crossing algorithm. On the other hand, the reverse situation is also possible, and the boundary crossing may be clearly a better option than the complementarity formulation. For example, in a problem with only one disjunction, but 20 equations in each of the terms of the disjunction, we may not be able to solve a NLP in which 20 of the equations contain 20 bilinear terms each. However, the number of subregions in the boundary crossing algorithm would be only 2, and the possibility of applying this algorithm efficiently would be much greater.

```
MODEL flowsheet;
(* units *)
       f1                             IS_A cheap_feed;
       f2                             IS_A expensive_feed;
       c1,c2                          IS_A single_compressor;
       s1,s2                          IS_A staged_compressor;
       r1                             IS_A cheap_reactor;
       r2                             IS_A expensive_reactor;
       co1,co2                        IS_A cooler;
       h1,h2,h3                       IS_A heater;
       fl1                            IS_A flash;
       sp1                            IS_A splitter;
       m1                             IS_A mixer;
       v1                             IS_A expansion_valve;
(* boolean variables  *)
       select_feed1,   select_single1               IS_A boolean_var;
       select_cheapr1, select_single2               IS_A boolean_var;
(* define sets  *)
       m1.n_inputs :==2;
       sp1.n_outputs :== 2;
(* wire up flowsheet *)
       f1.stream, f2.stream, c1.input, s1.input     ARE_THE_SAME;
       c1.output, s1.output, m1.feed[2]             ARE_THE_SAME;
       m1.out,co1.input                             ARE_THE_SAME;
       co1.output, h1.input                         ARE_THE_SAME;
       h1.output, r1.input, r2.input                ARE_THE_SAME;
       r1.output, r2.output,v1.input                ARE_THE_SAME;
       v1.output,co2.input                          ARE_THE_SAME;
       co2.output, fl1.feed                         ARE_THE_SAME;
       fl1.liq, h2.input                            ARE_THE_SAME;
       fl1.vap, sp1.feed                            ARE_THE_SAME;
       sp1.out[1], h3.input                         ARE_THE_SAME;
       sp1.out[2],c2.input, s2.input                ARE_THE_SAME;
      c2.output, s2.output,m1.feed[1]               ARE_THE_SAME;
(* Conditional statements *)
       WHEN select_feed1
         CASE TRUE:
              USE f1;
         CASE FALSE:
              USE f2;
       END;
       WHEN select_single1
         CASE TRUE:
              USE c1;
         CASE FALSE:
              USE s1;
       END;
       WHEN select_cheapr1
         CASE TRUE:
              USE r1;
         CASE FALSE:
              USE r2;
       END;
       WHEN select_single2
         CASE TRUE:
              USE c2;
         CASE FALSE:
              USE s2;
       END;
END flowsheet;
```

FIGURE 6 ASCEND model for the superstructure of Figure 5

implementation of the WHEN statement and the logical relations. Its performance has been tested in several small problems. As an example, it was applied to the system presented in Figure 5 and Figure 6. The value of the four boolean variables will determine the structure of the problem to be solved. As mentioned above, the values of the boolean variables could be defined interactively by the user, but they also could be defined by some logic inference algorithm which would allow the automatic change of the structure of the problem. The NLP contains 137 invariant equations and 68 variant equations for a total of 205. The configuration defined by one of the feeds, two single-stage compressors and one of the reactors contains 169 equations, the 137 invariant and 32 active equations out of the 68 variant equations. Switching form one structure to another is done without the need of recompilation and, since to reconfigure the system requires only rebuilding several list of pointers, it is computationally very efficient.

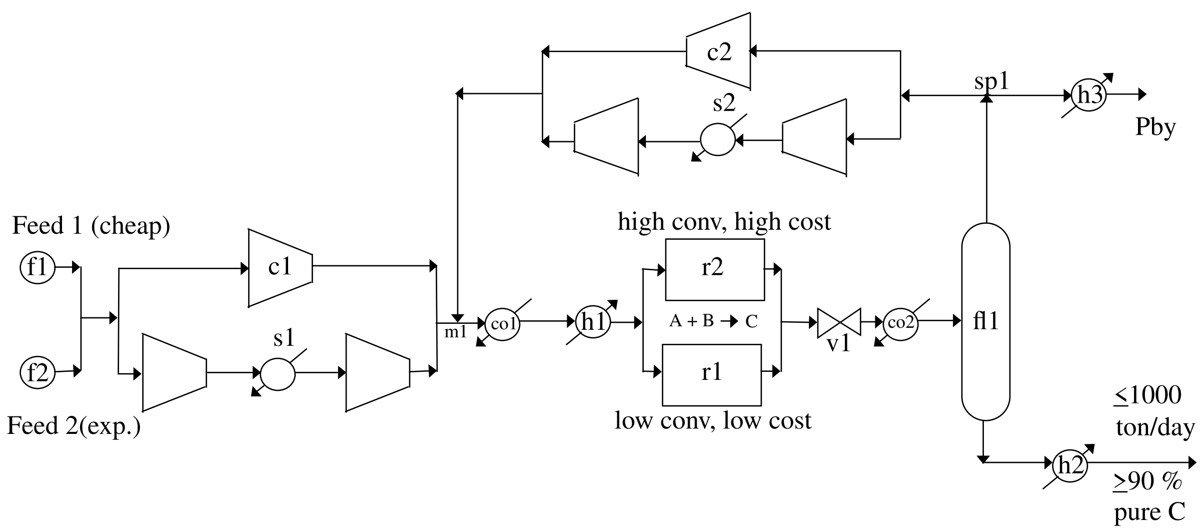


FIGURE 5 Superstructure taken from Turkay and Grossmann (1996).

4.2.2 Complementarity Formulation

Many examples of algebraic systems of disjunctive equations found in the literature have been represented and solved by using a complementarity formulation. In all the cases the results are encouraging. The solution of only one NLP is required and only a few iterations are used for convergence. However, it must be emphasized that, in all the cases, the number of equations in each disjunctive term (not the number of disjunctions) is very small; as a consequence, the complexity of the equations involving products among slack variables is not as bad as it can

Basically, a literature search has been conducted to find a possible solution for difficulty 3. To deal with nonconvexities in an optimization mode, Quesada and Grossman (1995) used linear relaxations based on the reformulation and linearization techniques for bilinear programming. Also, Viswanathan and Grossmann (1990) proposed augmenting the objective function with a penalty function that allows the violation of the linearizations of nonconvex constraints. A similar approach had been suggested by Bullard and Biegler (1991, 1992) in their iterated LP techniques for nonsmooth simulation. We are still not sure about how the application of these ideas could help to overcome the problems of having nonconvexities in the complementarity formulation. Research in this area is still ahead.

As mentioned before, the motivation for looking at a complementarity formulation is that only one NLP has to be solved. In general, we are presenting this approach as a very favorable alternative when the number of equations in each disjunctive term is small.

4 WORK TO DATE

4.1 MODELING

Both the WHEN statement and logical relations have already been incorporated to the ASCEND system as they were outlined above. We are currently testing this initial implementation, looking for improvements that could enhance its efficiency as modeling tools. To give an example of their application, we developed a simplified model for the superstructure given in Figure 5, taken from the work of Turkay and Grossmann (1996). In this example there are two alternative feedstocks, two possible choices of the reactor and two choices of the compression systems. Therefore, there are $2^4 = 16$ feasible configurations of the problem. All these 16 configurations are encapsulated in one ASCEND model containing 4 WHEN statements which depend on the value of 4 boolean variables. Figure 6 shows this model. The procedural section of the model and the model for each unit operation has been omitted for simplicity.

4.2 SOLUTION OF CONDITIONAL MODELS

4.2.1 Feeding a Conditional Model to an NLP solver

The mechanism suggested in section 3.2.2 has been implemented simultaneously with the

$$\begin{array}{l} h_1 \cdot h_2 + h_1 \cdot h_4 = 0 \\ h_3 \cdot h_2 + h_3 \cdot h_4 = 0 \end{array} \Rightarrow \begin{array}{l} h_1 \cdot (h_2 + h_4) = 0 \\ h_3 \cdot (h_2 + h_4) = 0 \end{array}$$

	h_1	h_2	h_3	h_4
1	$h_2 + h_4$	h_1	0	h_1
2	0	h_3	$h_2 + h_4$	h_3

FIGURE 4 Numerical singularities in factorable complementarity equations

the rows of the Jacobian corresponding to these equations. Note that the equations in Figure 4 contain the same four complementarity terms as the equations in (10), but they are grouped differently. In the case presented, if the solution of the problem is $h_2=h_4=0$, rows 1 and 2 of the Jacobian become numerically dependent as the Newton method approaches the solution (if $h_2+h_4 \sim 0$, we could get row 2 multiplying row 1 by the factor h_3/h_1). This is not the case in nonfactorable complementarity equations. There may be cases in which property 3 conflicts with property 4. Property 3 will have priority since it is what gives the simultaneous satisfaction of the equations in the same disjunctive term, a restriction that cannot be lost.

To get a formulation with all the properties outlined above is not a trivial exercise when the number of terms of the disjunction increases. However, for a disjunctive statement with two terms, the generalization to any number of equations is straightforward.

Many problems are incorporated with this kind of representation:

1. First of all, the problem grows quickly. If N is the number of equations in each term of the disjunction and D is the number of terms in the disjunction, the number of equations representing the disjunctive statement in the complementarity formulation is N(D+1).
2. The number of bilinear terms incorporated in each equation also grows with the number of equations in each term of the disjunction. The combinatorial nature of the problem is encapsulated here.
3. The performance of optimization techniques is badly affected by the introduction of nonconvexities (bilinear terms in the example) to the system of equations.
4. Numerical singularities can still arise for cases in which the solution resides on a boundary.

4. Every slack appears only once in each equation. There is no repeated slack index.

Posed in a formal way, given the disjunctive set of algebraic equations (two-term disjunction for simplicity)

$$\begin{bmatrix} f^1(x) = 0 \\ r(x) \le 0 \end{bmatrix} \vee \begin{bmatrix} f^2(x) = 0 \\ r(x) \ge 0 \end{bmatrix} \qquad (11)$$

the problem is reformulated as

$$\begin{gathered} g(x,y) = \begin{pmatrix} f^1(x) - h_1 \\ f^2(x) - h_2 \\ r(x) - h_3 + h_4 \end{pmatrix} \\ g(x,y) = 0 \\ C(h_1, h_2, h_3, h_4) = 0 \\ h_i \ge 0 \qquad i \in [1 \dots 4] \end{gathered} \qquad (12)$$

where now *hi* represents a vector of slacks variables and $C(h_1,h_2,h_3,h_4)$ is used to represent the set of complementarity equations involving the elements of these vectors. Note that in the case of inequalities, the complementarity condition is expressed in terms of the slacks defined in the same equation. Also, we will require that the equations defined by $C(h_1,h_2,h_3,h_4)$ present the features outlined above for the equations given in (10). Property 1 is intended to avoid a change in the number of degrees of freedom, while property 3 is required to ensure the simultaneous satisfaction of all the equations in the same disjunctive term. Property 1 and 3 must always be satisfied. In our example, property 2 is only a consequence of 3 and 4 rather than a requirement, and may not be present in a general case.

Newton and quasi-Newton methods are the most popular techniques for the solution of NLP problems. Property 4 is intended to decrease the possibility of numerical singularities in the Jacobian which can occur during an iterative scheme based on these methods. If a repeated index occurs and, therefore, factorization is possible for some of the slacks, numerical singularities could arise as the iterative scheme approaches some solutions of the complementarity problem. This is illustrated in Figure 4, where we show a case of factorable complementarity equations and

$$Re - 64/f = h_1 \qquad Re - (0.206307/f)^4 = h_2$$
$$Re = 2100 + h_3 - h_4 \tag{7}$$

in terms of the slacks, the disjunctive statement becomes:

$$\begin{bmatrix} h_1 = 0 \\ h_3 = 0 \\ h_2, h_4 \geq 0 \end{bmatrix} \vee \begin{bmatrix} h_2 = 0 \\ h_4 = 0 \\ h_1, h_3 \geq 0 \end{bmatrix} \tag{8}$$

and the standard complementarity formulation is given by (7) and

$$h_1 \cdot h_2 = 0$$
$$h_3 \cdot h_4 = 0 \tag{9}$$
$$h_i \geq 0 \qquad i \in [1 \ldots 4]$$

It should be noted that the disjunctive statement requires either both h_1 and h_3 or both h_2 and h_4 simultaneously to be zero, and this is not a restriction in the complementarity formulation given by (9). The following formulation for disjunctive statements is proposed instead:

$$h_1 \cdot h_2 + h_3 \cdot h_4 = 0$$
$$h_1 \cdot h_4 + h_3 \cdot h_2 = 0 \tag{10}$$
$$h_i \geq 0 \qquad i \in [1 \ldots 4]$$

Since the slacks variables are all positive, the equations in (10) not only contains the complementarity condition given in (9) but also enforces simultaneous satisfaction of all the equations defined in the same term of the disjunction.

The complementarity equations given in (10) have the following properties:

1. The number of equations is equal to the number of equations in each term of the disjunction.
2. The number of terms in each equation is also equal to the number of equations in each term of the disjunction.
3. Every slack variable is multiplied by every slack in the other term of the disjunction.

implicitly or explicitly in such a CASE are set as active.

4. All the variables incident in the active set of equations are active.
5. The current problem to be solved consists of the active set of equations and variables.

The mechanism outline above is not precisely specific for the boundary crossing algorithm. For example, a branch and bound technique would also require the switching from one problem to another. Specifically for the boundary crossing algorithm, there is another major complication that must be addressed. While analyzing a boundary, the gradient step requires information about the equations in all of the subregions neighboring the boundary. That means the more than one set of equations has to be used simultaneously. We still do not address this and some other difficulties that we may not have discovered yet. However, based on previous programming exercises, we feel optimistic of being able to overcome any difficulty involved in this implementation in the time remaining for this research project.

3.2.3 Complementarity Representation of a Conditional Model

In this section, a complementarity formulation is proposed for the representation of conditional models. This formulation not only establishes the complementarity condition among alternative sets of equations but also enforces simultaneous satisfaction of all the equations in the solution set. The formulation proposed here is an extension of that presented by More(1994) who reformulates the nonlinear complementarity problem (3) as:

$$g(x,y) = \begin{pmatrix} f(x) - h \\ H \cdot x \end{pmatrix}$$

$$g(x, y) = 0 \tag{5}$$

$$x \geq 0 \qquad y \geq 0$$

where H is the diagonal matrix diag(h_i). The next simple example will show how a complementarity formulation should change for the case of disjunctive statements. Consider the flow transition given by:

$$\begin{bmatrix} Re = 64/f \\ Re \leq 2100 \end{bmatrix} \vee \begin{bmatrix} Re = (0.206307/f)^4 \\ Re \geq 2100 \end{bmatrix} \tag{6}$$

defining slack variables h for both the equalities and inequalities:

some parameters, such as cost factors, mass balance coefficients, power of a correlation, etc. In such a case, the need of a combinatorial structural analysis is eliminated.

2. The disjunctions of the problem are disjointed, and its structural analysis may be done independently. There are two cases in which this may happen *i)* the incidence pattern of the disjunction are totally decoupled, and *ii)* the existing coupling still allows the calculation of the variables output assigned to the disjunctive equations separately in some precedence order.

We have applied all these ideas for analyzing by hand small conditional problems, and we believe that their practical implementation is possible. A computer implementation which can take advantages of them is desirable.

3.2.2 Implementation of the Boundary Crossing Algorithm

A part of this project will consist in the implementation and testing of the boundary crossing algorithm. In general, it can be considered that both the Newton and gradients steps required in this algorithm are already well defined. We are looking, however, for ideas which could give the computational efficiency needed for its application in a general purpose environment like ASCEND.

We mentioned in section 3.1.1 that the implementation of the WHEN statement in ASCEND would allow us to have available the data structure for all the variables and equations of the conditional model. Therefore, the first step in the implementation of a conditional solver is to decide how the solver is going to be fed with the correct structure and how to change this structure efficiently in an iterative solution scheme. To perform this task, we will use the notion of active equation and active variable. By active we mean "it is part of the problem currently being solved". The following steps would provide a mechanism to select the structure of the problem:

1. Initially, all the equations resulting from the compilation are considered as active.
2. Then, all the equations which are implicitly or explicitly (inside models) stated in the CASE of a WHEN statement are set as inactive. The equations set as inactive in this step constitute the variant set of equations. All the equations which remain active constitute the invariant set of equations.
3. Analyze the WHEN statements. According to the current values of the variables on which each WHEN statement depends, determine which of its CASEs applies. The equations stated

crossing algorithm.

3.2 SOLVING CONDITIONAL MODELS

3.2.1 Structural Analysis

The analysis required for partitioning the variables and the equations in a conditional model was outlined in section 2.2.1. In our point of view, the most serious disadvantage of this analysis is the combinatorial nature of the search consistency algorithm, which requires the analysis of all the feasible subregions resulting from the dissection of the feasible region. We think that, in general, this analysis cannot be avoided. If we want to be sure about the consistency of the conditional model, we would have to pay the price. However, based on the work of Allen and Westerberg (1976), we also think that sometimes it is possible to take considerable advantage of the structure of the problem, reducing the effort required by such an analysis.

In chemical engineering design and simulation, systems modeled with indexed equations and variables occur very often. Typical examples are a distillation column (indexed by the number of stages) and systems of equations arising in the discretization of an initial value problem (indexed by the number of time steps). If we think of a problem in which a disjunction exists in each of the repeated structures, the combinatorial complexity in the structural analysis would be unmanageable. However, intuition suggests that, in this kind of problem, the degrees of freedom analysis should not be affected by the number of repeated structures and, therefore, the analysis could be simplified. In his work, Allen used a representative incidence matrix to perform a systematic analysis of indexed equation sets. Again, this could not be applied to an arbitrary system of disjunctive equations, but if we know a priori that our problem contains an indexed repeated structure, then we may take advantage of it.

We have also observed special features of some problems which can contribute to simplifying the analysis. Even though they represent very particular cases, if found alone or in any combination, a simple checking in the computational implementation of the algorithm could give us saves in time and in the computational effort needed to perform the analysis. These are:

1. The incidence pattern of the different terms of the disjunction is the same. It happens sometimes that the alternative equations are only different because the difference in value of

(a)

```
laminar     IS_A boolean;
Re,f,k      IS_A factor;

CONDITIONAL
      condition: Re <= 2100;
END;

laminar == SATISFIED(condition);

eq1: Re = 64/f;
eq2: Re = (0.206307/f)^4;

WHEN laminar:
      CASE TRUE:
            USE eq1;
      CASE FALSE:
            USE eq2;
END;
```

(b)

```
valve_open        IS_A boolean;
pump_on           IS_A boolean;
full_tank         IS_A boolean;

valve_open == pump_on AND NOT(full_tank);
```

FIGURE 3 Examples of the syntax for the implementation of logical relations

AND, OR, and NOT are allowed. Equality in a logical equation can also be interpreted as an *if and only if* implication between two logical terms expressed in clausal form. Also, it should be noted that we can express any logical clause using the proposed syntax by simply writing the clause in one of the terms, and the constant boolean value TRUE in the other term of the logical equality. Example (a) is a more complete version of the model presented in Figure 2. In this case it can be noted that, when the value of a boolean variable depends on the truth value of a real expression, the real expression is defined and labeled inside the CONDITIONAL statement; then the logical operator SATISFIED will give the truth value of such an expression. If the truth value of the real expression is TRUE, the result of SATISFIED(label of expression) will be TRUE. The objective of the operator SATISFIED is twofold: 1) it makes the life of the compiler easier, since the logical expressions are decoupled from the real expressions on which they depend, and 2) the use of this syntax forces the user to define a boolean variable (laminar in the example) which will have the same truth value as the real expression. That is useful for browsing purposes.

It is important to emphasize that, by checking the value of the appropriate boolean variables after each iteration, the incorporation of logical relations will allow, if required, automatic change of the structure of the problem in an iterative scheme such as the boundary

the value of the symbol method. Once a configuration has been select, it will be kept unless the user decides to change it. Note that the user does not have to recompile the model to make the switching, since the data structure for both of the problems is available and, therefore, the reconfiguration of the system can be done automatically by simply changing the value of the conditional variable. An example of this application would be the synthesis of process networks.

On the other hand, in case (a) we cannot expect the boolean value of the variable laminar to be a user decision. Its value will depend on the value of the Reynolds number which is an unknown in the problem. Actually, the value of the variable laminar will be the truth value of the expression Re≤2100. Here we have a conditional model. In an iterative solution scheme, we will expect that the value of the boolean variable laminar will change, and so will the structure of the system of equations that we have to solve.

3.1.2 Incorporation of Logical Relations

Motivated by some recent applications of logic in the solution of conditional models (Raman and Grossmann, 1994; Turkay and Grossmann, 1996) and trying to expand the ASCEND capabilities for modeling, an implementation of logical relations as a modeling tool is proposed. To our knowledge, no other general purpose equation-based modeling environment provides this capability.

In the implementation of logical relations, the equation based approach is going to be maintained. That is, the user will state the logical relations that must be true at the solution to the problem but not how to solve them. Each logical equation will have a residual attached to it. This residual will indicate if the expression is satisfied or not. Therefore, this incorporation also means that we need to provide a solver which knows how to deal with logical relations. The work of Raman and Grossman (1993) can be used as a starting point for such a solver implementation. They used formal procedures for performing logic inference depending on the way in which the logical relations are present (CNF or DNF).

Figure 3 illustrates two examples of the syntax proposed for the implementation of logical relations. The symbol double equality will indicate that we have a logical relation and will separate the two logical terms of the relation. In each of the two terms, logical operators such as

whose name is given in each of the cases should be declared inside the model which contains the WHEN statement.

2 The variables in the list of variables can be of any type among boolean, integer or symbol or any combination of them. That is, we are not limited to the use of boolean variables. Obviously, The list of values in each case must be in agreement with the list of variables in the number of elements and type of each element.

3 Names of arrays of models or equations are also allowed inside the scope of each case.

The proposed extension (allowing the user to define the domain of validity of both models and equations inside the cases of a WHEN statement) enormously increases the scope of modeling in an equation based modeling environment. Of course, this implementation may be possible in other modeling environments which support object oriented concepts, but we have proposed it as an extension to the ASCEND modeling language because ASCEND is the best one locally understood. Figure 2 shows two incomplete ASCEND models in which the WHEN statement is used, each in a different manner.

```
      (a)
laminar      IS_A boolean;
Re,f,k       IS_A factor;

eq1: Re = 64/f;
eq2: Re = (0.206307/f)^4;

WHEN laminar:
      CASE TRUE:
            USE eq1;
      CASE FALSE:
            USE eq2;
END;
```

```
              (b)
method             IS_A symbol;
simplified_flash   IS_A VLE_flash;
rigorous_flash     IS_S td_VLE_flash;

WHEN method
      CASE 'rigorous':
            USE rigorous_flash;
      CASE 'simplified':
            USE simplified_flash;
END;
```

FIGURE 2 Two different applications of the WHEN statement.

In case (b) the value of the symbol method is used only to select between two alternative configurations of the problem, a flash calculation assuming constant relative volatility (VLE_flash) or a flash calculation using a more rigorous thermodynamic calculation (td_VLE_flash). The method that is going to be used and, therefore, the value of the symbol method, is a user decision. We can expect that the user will select the simplified model looking for good initial values of the variables and then he will switch to the rigorous model by changing

with the solution of these problems. Some issues involved in the implementation of the boundary crossing algorithm will be briefly explained, and an extension of a complementarity formulation to represent conditional models will be proposed.

3.1 MODELING

In this section, an extension to the ASCEND modeling language is proposed. Essentially, it consists of implementing the WHEN statement and incorporating logical relations into the system.

3.1.1 The WHEN Statement

Originally, the syntax for incorporating conditional dependence of some equations of the model in an equation-based environment was suggested by Piela (1989). That syntax is very similar to the CASE equation of *gPROMS* and suffers the same limitations. Instead, we want to take advantage of the fact that ASCEND is based on object oriented concepts where model definitions can contain parts that contain parts to any level. Furthermore, in ASCEND, a simple relation is treated as an object by itself and can have a name. Based on these ideas, the proposed syntax for the WHEN statement is:

```
WHEN list_of_variables
      CASE list_of_values_1:
                              USE name_of_equation_1;
                              USE name_of_model_1;
      CASE list_of_values_2:
                              USE name_of_equation_2;
                              USE name_of_model_2;
      CASE list_of_values_nminus1:
                              USE name_of_equation_nminu1;
                              USE name_of_model_nminus1;
     OTHERWISE:
                              USE name_of_equation_n;
                              USE name_of_model_n;
END;
```

The following are important observations about the proposed implementation:

1 We are not dealing with conditional compilation. We create and have available the data structures for all of the variables and equations in each of the models. This is actually a requirement for the solution algorithms of conditional models. All the models and equations

quite arbitrary while the other is the vector of variables. Many commonly occurring problems have a more general form:

$$\begin{gathered} f^1(x)^T \cdot f^2(x) = 0 \\ f^1(x) \geq 0 \quad f^2(x) \geq 0 \end{gathered} \tag{4}$$

This problem is called the vertical nonlinear complementarity problem (Ferris and Pang, 1995). It is possible, of course, to have more than two vectors of functions in the above equations. Later, we will propose the representation of disjunctive sets of algebraic equations as complementarity problems. It is important to mention that we are aware of many disadvantages associated with this representation and, therefore, potential solutions to these problems will be mentioned as well. Our motivation is the fact that a complementarity formulation only requires the solution of a nonlinear system of equations, avoiding all of the complications encountered in the boundary crossing algorithm and the discrete decision making of a MINLP solution.

2.2.3 Solution of Conditional Models in an Equation-Based Modeling Environment

Finally in this section, we consider again the analysis of the previous implementations of conditional modeling to mention their status in the solution of these problems. Both the IF construct of *SpeedUp* and the CASE equation of *gPROMS* are used only to dynamically change the active set of equations during a dynamic simulation by evaluating the logical expression at any point in time. They assume, however, that the initial state of the system (and hence, the initial active set of equations) is known a priori. In other words, they do not address the problem of finding a consistent set of initial values that satisfy both the logical conditions and the active set of equations.

3 PROPOSED CONTRIBUTIONS

One of the goals of our research group has been to improve the ability to develop and solve process models. One of the results of this effort has been the development of ASCEND (Piela *et al*., 1991). Westerberg *et al.* (1994) discuss the essential features of the current ASCEND system and present several ways in which we can improve it to solve larger models and to increase its scope as a modeling environment. In section 3.1 we will propose the implementation of tools which will allow ASCEND to represent conditional models efficiently. Section 3.2 is concerned

In the gradient analysis at boundaries, a descent direction for the objective functions (norm of the residuals of the equations) local to each neighboring subregion is required for convergence. This direction d will be given by:

$$\begin{aligned} \min_{\alpha} \quad & d^T \cdot d \\ \text{s.t.} \quad & d = N \cdot \alpha \\ & e^T \cdot \alpha = 1 \\ & \alpha \geq 0 \end{aligned} \tag{2}$$

where the matrix N is known and is used to store the vectors of the gradients of the objective functions local to each neighboring subregion which borders a boundary, and e is a vector with all elements equal to one. The vector α, whose elements sum to one, is used to generate the direction d as the linear combination of the gradients which yields a vector of smallest length. Termination occurs if a minimum value of zero is obtained for d, which means that the gradient vectors of N cannot be confined to one side of a half-plane.

2.2.2.2 Complementarity Formulation

The class of problems known as complementarity problems has become increasingly popular as a tool for addressing practical problems arising in mathematical programming, engineering, and the sciences (Billups, 1995; Ferris and Pang, 1995). Dirske and Ferris (1995) describe the standard forms for the different classes of complementarity problems and give examples of how to formulate many popular problems as mixed complementarity problems (MCP). Billups (1995) proposes strategies which enhance the robustness of Newton-based methods for their application in these problems.

In general, the nonlinear complementarity problem is expressed as the following set of equations and inequality constraints:

$$\begin{gathered} x^T \cdot f(x) = 0 \\ x \geq 0 \qquad f(x) \geq 0 \end{gathered} \tag{3}$$

There is certain lack of symmetry in the previous formulation. One of the functions is

problem that is often solvable as a relaxed LP. They consider the nonlinear case although no guarantee of convergence is given for it. Also, based in the works of Raman and Grossmann (1993, 1994), Turkay and Grossmann (1996) address the solution of disjunctive set of nonlinear programming problems. It is shown that the use of logic can improve the efficiency and robustness of MINLP algorithms on the solution of structural optimization problems.

It is worth summarizing some of the observations and conclusions about MINLP algorithms arisen from the work of Zaher (1995): 1) it can be expected that during the solution of a MINLP, the algorithm will jump from one subregion to another which is remote from it, leading to the poor initialization of some of the nonlinear subproblems, 2) the nonlinear subproblems insist in converging the equations of what might be an incorrect subregion and 3) this method employs a highly mathematical approach which remains as a rigorous general purpose algorithm designed for problems where discrete decision making cannot be avoided.

The following two sections summarize two alternative approaches to the solution of conditional models.

2.2.2.1 Boundary Crossing Algorithm

Zaher (1991) introduced the boundary crossing algorithm as an alternative to the MINLP formulation for the solution of conditional models. In this approach, the inequalities defining the domain of each set of variant equations are considered as logical conditions which, when satisfied, will identify a particular subregion and, if violated, will identify an alternative one. In that way, every equation $r_{ik}(x)=0$ constitutes a boundary which will separate the feasible region in two neighboring subregions. In general, identification of a subregion requires a unique combination of boolean values for the collection of all logical conditions. If l is the number of boundary equations, as many as 2^l subregions can be created.

The boundary crossing algorithm has been given in Zaher (1991). Basically, the algorithm applies a Newton step for all the points not residing at boundaries. If during the iterative process a boundary is reached, then a Newton step cannot be taken because the system is nondifferentiable at that point (different subregions have different set of equations). Therefore, a gradient analysis is performed instead.

problem can be decomposed in smaller partitions which can be individually solved in some precedence order. Structural analysis is an indispensable tool in an equation-based modeling environment, and as it will be shown, conditional models make the need for this tool an even stronger requirement.

In conditional models, the sparsity pattern is expected to change from one subregion to another. This implies that a consistent set of independent variables for one subregion may not be valid for another subregion. For the same reason, block partitioning of the equations for one region may not be possible using the equations of another.

Zaher (1995) addressed the structural analysis of conditional models. A necessary condition for structural consistency in conditional models is that each of the alternative sets of equations must be structurally consistent. Hence, consistency of a conditional model is assessed by finding at least one consistent partitioning of the variables (independent-dependent) such that output assignment of all of the equations in each subregion can be performed. This requirement makes the problem combinatorial, since we have to perform the analysis for all of the possible alternative set of equations.

Regarding the equation partitioning, it is recommended one to obtain a representative incidence matrix rather than attempt to incorrectly decouple a wrong set (not the solution set) of equations. This representative incidence of equations in a disjunctive statement can be achieved by 'OR'ing the incidences of the inequalities (or boundary expressions) with each locally defined equation. Then, additional incidences are obtained for each equation in a subregion one at a time by merging the incidences of the equation with those of exactly one equation from all of the other subregions. It should be pointed out that it is also possible and valid to obtain an individual block partitioning while analyzing any particular subregion, and that the previous algorithm was proposed by Zaher when seeking efficiency.

2.2.2 Solution Algorithms for Conditional Models

As mentioned earlier, conditional models can be formulated as mixed integer programming problems. In a very recent work, Grossmann and Turkay (1996) show that, in the case of linear equations, the solution of a conditional model can be found by solving a MILP

2 BACKGROUND

2.1 BUILDING A CONDITIONAL MODEL

Previous implementations of conditional statements in an equation-based modeling environment have been reported. One such mechanism is the IF-THEN-ELSE construct of *SpeedUp* described by Pantelides (1988):

```
IF logical_condition THEN
      equation1
ELSE
      equation2
ENDIF
```

where both the logical conditions and the equations are expressed in terms of the model variables. Such a construct defines two system states corresponding to the IF and the ELSE clauses respectively. Multiple states may be described by nesting several IF statements.

Similarly, Barton (1992) and Barton and Pantelides (1994) incorporated the CASE equation to *gPROMS*. The CASE equation is used to define both the appropriate modeling equations in each state and the logical conditions for transitions among states. It covers multiple states within only one statement and has the advantage of successfully representing irreversible discontinuities.

There is a major difficulty in both of the previous approaches as tools for conditional modeling. They only allow the substitution of a list of equations (or arrays of equations) for another. In an equation-based modeling environment in which concepts like hierarchy (building complex models from small models) and inheritance (adding new features to a model) are constantly in use, that represents a severe disadvantage in efficiency. Thinking of how many equations have to be written down when substituting some of the units of a superstructure makes obvious the lack of flexibility in the scope of these approaches. Later in this paper, we will show our extension to these modeling capabilities with the incorporation of the WHEN statement to the ASCEND modeling environment.

2.2 SOLVING A CONDITIONAL MODEL

2.2.1 Structural Analysis for Conditional Models

Before attempting to solve a model, a structural analysis has to be performed to determine if the formulation is well posed. Similarly, a partitioning algorithm is applied to find if the original

as determined by the simultaneously calculated value of the flow type indicator (Reynolds or Mach number) relative to some critical value.

While many currently available equation-based modeling systems have been reported in the literature, only a few of them have given attention to conditional models (Pantelides, 1988; Piela,1989; Barton, 1992). Several problems arise when building and testing models of complex processes in an equation-based environment (providing initial guesses, analyzing degrees of freedom, efficiently computing solutions, etc.). The combinatorial nature of a conditional model makes the solution of these problems even a more difficult task. In this work, we propose the incorporation of a series of tools which enables the user of an equation-based modeling environment with the capability of dealing with complex conditional models. It is important to distinguish between two well defined and separated (but complementary) goals of this research project:

1. Efficient representation of a conditional model in an equation-based modeling environment. This is intended to be independent of any particular application, or of any solver or algorithm used for finding a solution to the system of equations. Potential applications of this capability varies from the simple substitution of one equation for another (as in the case of the laminar-turbulent flow transition), to the substitution of a section of a chemical plant for another (as can be required while analyzing and initializing a superstructure).
2. Implementation and testing of alternative approaches to the solution of a conditional model. Modeling is a user dependent task. Frequently, the same problem is formulated differently from one modeler to another and, as a consequence, different approaches and techniques could be used to find a solution to it. Conditional models can be formulated as mixed-integer programming problems. See for example Grossmann and Turkay (1996). Besides the mixed-integer formulation, some alternative approaches exist which, in principle, could avoid the need of discrete decisions. Research in this work will focus in two of them: the boundary crossing algorithm (Zaher, 1991; Zaher, 1993; Zaher, 1995) and the complementarity formulation (Billups, 1995). In particular, we will propose an approach for the implementation of the boundary crossing algorithm in an equation-based environment, and an extension to the standard complementarity formulation for the representation of conditional models.

1 INTRODUCTION

The design and simulation of engineering processes require one to find the solution to a large system of linear and/or nonlinear equations. This work is particularly concerned with the representation and solution of conditional models. Conditional models exist when the defining equations of the system depend on where the model solution lies. A conditional model consists of a system of equations expressed by two sets: a globally defined invariant set of equations and a variant (or locally defined) set of conditional equations which are expressed as disjunctions. Basically, as shown by Zaher(1993), and Grossmann and Turkay (1996), this problem reduces to representing and finding the solution to the system of disjunctive equations:

$$
\begin{gathered}
g(x) = 0 \\
\bigvee_{i \in D_k} \begin{bmatrix} h_{ik}(x) = 0 \\ r_{ik}(x) \leq 0 \end{bmatrix} \quad k \in K \\
x \in R^n
\end{gathered}
\tag{1}
$$

where $g(x)$ and $h_{ik}(x)$ represent the invariant and the variant sets of equations respectively, K represents the set of disjunctions and the index i is used to indicate the i-th term in each disjunction D_k. The equations $g(x)$ can be said to be defined over the entire feasible region, while the inequalities define the domain of validity of each variant set of equations $h_{ik}(x)$. In that way, each variant set of equations is confined to some subregion resulting from the dissection of the feasible region (Zaher, 1991). The solution of the system will be given by the vector x satisfying the invariant set $g(x)$ and exactly one set of equations for each of the disjunctions, providing that the corresponding set of inequalities is satisfied.

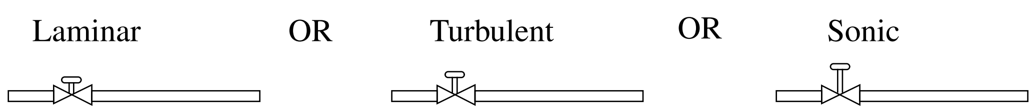


FIGURE 1 Fluid flow transition

Fluid transition represents an example commonly encountered in chemical engineering. This is illustrated in Figure 1 taken from the work of Zaher (1995). In this case, alternative equations for the transport properties and velocity bounds must be incorporated within the model

Representing, Analyzing and Solving Conditional Models in an Equation-Based Environment

by Vicente Rico-Ramírez

Advisor: Arthur W. Westerberg

Ph.D. Thesis Proposal

Department of Chemical Engineering

Carnegie Mellon University

Pittsburgh, PA

January 15, 1997

Abstract

Conditional models arise in chemical engineering when modeling systems involving physicochemical discontinuities, such as phase transitions. There are also cases where conditional cost functions are used in the optimization of process systems. Modeling tools for the efficient representation of conditional models in an equation-based environment are proposed in this paper. Also, some of the issues involved in the solution of these problems are discussed. The boundary crossing algorithm and the complementarity formulation are identified as alternatives to the MINLP algorithms for the solution of conditional models. In particular, we will propose an approach to the structural analysis of a conditional model, an extension to the standard complementarity formulation for the representation of conditional models, and an approach for the implementation of the boundary crossing algorithm in an equation-based environment.

More, J. J.; Global Methods for Nonlinear Complementarity Problems. Technical report, Argonne National Laboratory, Mathematics and Computer Science Division, April 1994.

Pantelides, C. C.; SPEEDUP-Recent Advances in Process Simulation. *Comput. Chem. Eng.*, 12(7):745–755, 1988.

Pantelides, C. C. and Barton, P. I.; Equation-Oriented Dynamic Simulation: Current Status and Future Perspectives. *Comput. Chem. Eng.*, 17S:263–285, 1993.

Piela, P., Epperly, T., Westerberg, K., Westerberg, A. W.; An Object-Oriented Computer Environment for Modeling and Analysis: The modeling language. *Comput. Chem. Eng.*, 15(1):53–72, 1991.

Piela, P.; ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis. PhD thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, April 1989.

Quesada, I. and Grossmann I. E.; A Global Optimization Algorithm for Linear Fractional and Bilinear Programs. *Journal of Global Optimization*, 6:39–76, 1995.

Raman, R. and Grossmann, I. E.; Symbolic Integration of Logic in Mixed-Integer Linear Programming Techniques for Process Synthesis. *Comput. Chem. Eng.*, 17(9):909–927, 1993.

Raman, R. and Grossmann, I. E.; Modeling and Computational Techniques for Logic Based Integer Programming. *Comput. Chem. Eng.*, 18(7):563–578, 1994.

Turkay, M. and Grossmann, I. E.; Logic-Based MINLP Algorithms for the Optimal Synthesis of Process Networks. Comput. Chem. Eng., 20(8):959–978, 1996.

Viswanathan, J. and Grossmann, I. E.; A Combined Penalty Function and Outer Approximation Method for MINLP Optimization. *Comput. Chem. Eng.*, 14(7):769–782, 1990.

Westerberg, A.W., Abbott, K. A., and Allan, B. A.; Plans for ASCEND IV: Our Next Generation Equational-Based Modeling Environment. Boston, Massachusetts, November 1994. AspenWorld'94.

Westerberg, A. W. and Director, S. W.; A Modified Least Square Algorithm for Solving Sparse n x n Sets of Nonlinear Equations. Technical report, Carnegie Mellon University, Engineering Design Research Center, DRC-06-5-79, January 1979.

Zaher, J. J.; Conditional Modeling. PhD thesis, Department of Chemical Engineering,Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1995.

Zaher, J. J.; Conditional Modeling in an Equation-Based Modeling Environment. The Annual AIChE National Meeting, 1991. paper 138c.

Zaher, J. J.; Conditional Programming. The Annual AIChE National Meeting, March 1993.

of this project. Some of the weaknesses and advantages of each of these approaches have been mentioned, but overcoming some of the difficulties and testing the approaches in large scale problems are issues still ahead as part of the future work. The following remain as some of the questions yet to be answered: *i)* How can the information provided by the structure of the problem help to simplifying the degrees of freedom analysis of a large conditional model? *ii)* How is the performance of a complementarity formulation against that of the boundary crossing algorithm in the solution of conditional models of practical size? *iii)* How is their performance in an optimization scheme? How does one overcome the difficulties incorporated by the bilinear terms in the complementarity formulation? Under what conditions can these approaches be considered a reliable alternative to MINLP algorithms in the simulation and optimization of large scale models?.

7 REFERENCES

Allen, G. L. and Westerberg, A. W.; Solution Procedures for Indexed Equation Sets: Structural Considerations. *AIChE Journal*, 22(3):549–558, May 1976.

Barton, P. I.; The Modeling and Simulation of Combined Discrete/Continuous Processes. PhD thesis, Department of Chemical Engineering, Imperial College of Science, Technology and Medicine, 1992.

Barton, P. I. and Pantelides, C. C.; Modeling of Combined Discrete/Continuous Processes. *AIChE Journal*, 40(6):966–979, June 1994.

Billups, S. C.; Algorithms for Complementarity Problems and Generalized Equations. PhD thesis, University of Wisconsin-Madison, August 1995.

Bullard, L. G. and Biegler, L. T.; Iterative Linear Programming Strategies for Constrained Simulation. *Comput. Chem. Eng.*, 15(4):239–254, 1991.

Bullard, L. G. and Biegler, L. T.; Iterated Linear Programming Strategies for Nonsmooth Simulation: Continuous and Mixed-Integer Approach. *Comput. Chem. Eng.*, 16(10), 1992

Dirske, S. P. and Ferris, M. C.; MCPLIB: A Collection of Nonlinear Mixed Complementarity Problems. Optimization Methods and Software, 5, 1995.

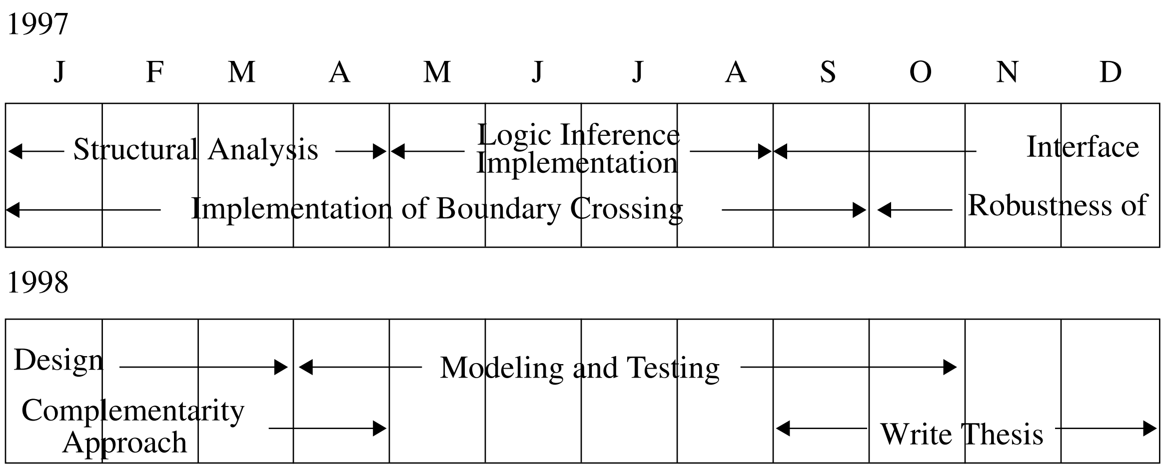
Ferris, M. C. and Pang, J. S. ; Engineering and Economic Applications of Complementarity Problems. Technical report, University of Colorado, 1995.

Grossmann, I. E. and Turkay, M.; Solution of Algebraic Systems of Disjunctive Equations. *Comput. Chem. Eng.*, 20:S339–44, 1996. Suppl. Part A.

An aspect that I have not mentioned, but that I anticipate it will be required, is the redesign of the existing interface to the ASCEND solver. I expect that the new interface tools not only will fit in with the current look of the ASCEND environment but also will provide the user with the capability of analyzing conditional models. An example of the new features that have to be added to the system is the capability of analyzing a conditional model at a boundary, where the neighboring sets of equations contain different structure.

The following is a tentative schedule for the remaining time of my research in this project.

Schedule of Proposed Work



6 CONCLUSIONS

There is a contribution to be made in the modeling world. Namely, the efficient representation of conditional models in an equation-based environment. We have described the modeling tools required to accomplish this, showing how the expressiveness of the modeling language increases with their incorporation. The computer implementation of these tools has been completed. There are surely other applications of the tools we propose that are being overlooked here. We hope to discover them.

We have also analyzed two alternative approaches to the solution of conditional models. The representation of conditional models as complementarity problems has been proposed and tested with small problems. The theoretical issues of the boundary crossing algorithm have been previously established, but an efficient computer implementation of it is considered to be a part

In the example of the linear mass balance, Turkay and Grossmann (1996) solve the problem by using a mixed-integer approach. The resulting MILP contains 18 binary variables, 66 continuous variables and 89 linear equations. In the complementarity formulation, the NLP contains 81 equations, 47 are linear, but the remaining 34 contain multiplications among slacks. The size of the problem is very similar, the difference will be in either using a branch and bound search in the MILP or dealing with the complementarity equations in the solution of one NLP.

We have thought that an interesting approach for dealing with conditional models would be to use the boundary crossing algorithm but representing some parts of the problem as a complementarity set of equations. In that way, we could obtain the advantages of both approaches. Once again, we consider this analysis to be part of the future work.

5 FUTURE WORK

In the previous two sections, we tried to emphasize the issues that still have to be addressed in both the representation and solution of conditional models; we also mentioned a description of a starting point for each of them. Making a brief summary list, we have:

1. Implementation of the boundary crossing algorithm. Issues inherently involved in this implementation (but with a much more general scope of application) include:
   i) Structural analysis in conditional models. This would include the idea of using the information provided by the structure of the problem in order to simplify the analysis, as outlined in section 3.2.1
   ii) Implementation of a logic inference algorithm.
2. Research in the field of the complementarity formulation for disjunctive systems of algebraic equations. Mainly, we need to investigate how to apply some of the ideas found in the literature to deal with the complementarity equations in an optimization mode and to improve the robustness of a NLP solver for the solution of this kind of equations.
3. Modeling and testing of large conditional models. From this analysis, we expect to be able to define the scope of application of the different approaches, to draw conclusions about the advantages and disadvantages of each of them and to combine them in order to get the best of both approaches.

be expected with problems of larger size.

Table 1: Examples of algebraic systems of disjunctive equations found in the literature

Example	Reference	Number of Equations	Number of Disjunctions	Number of Complementarity Equations	Iterations
Fluid transition	Zaher (1995)	7	1	1	11
Phase equilibria	Zaher (1995)	18	3	3	8
Heat exchanger	Zaher (1995)	56	2	4	12
Linear mass balance	Grossmann and Turkay (1996)	81	6	34	17
Pipeline network	Bullard and Biegler (1992)	250	38	76	27

The results of some of these examples are shown in Table 1. When possible, the initial values for the variables used were the same as those given in the reference (in some cases these values are not reported). To solve the examples, we used our ASCEND NLP solver which applies the modified Newton algorithm given in Westerberg and Director (1979). Since the hardware and the technique that we are using to get the solution is different, we are not comparing neither time nor number of iterations with other works. Still we consider it important to make remarks about some differences of the alternative approaches.

In the example of the pipeline network, the result of the complementarity representation is one NLP containing 250 equations, 76 of them containing 2 bilinear terms. In the same example, the number of boundaries in the boundary crossing algorithm would be 38 (that means 2^{38}=2.748779 x 10^{11} possible subregions) and the NLP to be solved in every subregion would contain 98 equations. The combinatorial complications present in examples like this impose a severe restriction to the boundary crossing algorithm. On the other hand, the reverse situation is also possible, and the boundary crossing may be clearly a better option than the complementarity formulation. For example, in a problem with only one disjunction, but 20 equations in each of the terms of the disjunction, we may not be able to solve a NLP in which 20 of the equations contain 20 bilinear terms each. However, the number of subregions in the boundary crossing algorithm would be only 2, and the possibility of applying this algorithm efficiently would be much greater.

```
MODEL flowsheet;
(* units *)
       f1                             IS_A cheap_feed;
       f2                             IS_A expensive_feed;
       c1,c2                          IS_A single_compressor;
       s1,s2                          IS_A staged_compressor;
       r1                             IS_A cheap_reactor;
       r2                             IS_A expensive_reactor;
       co1,co2                        IS_A cooler;
       h1,h2,h3                       IS_A heater;
       fl1                            IS_A flash;
       sp1                            IS_A splitter;
       m1                             IS_A mixer;
       v1                             IS_A expansion_valve;
(* boolean variables  *)
       select_feed1,   select_single1               IS_A boolean_var;
       select_cheapr1, select_single2               IS_A boolean_var;
(* define sets  *)
       m1.n_inputs :==2;
       sp1.n_outputs :== 2;
(* wire up flowsheet *)
       f1.stream, f2.stream, c1.input, s1.input     ARE_THE_SAME;
       c1.output, s1.output, m1.feed[2]             ARE_THE_SAME;
       m1.out,co1.input                             ARE_THE_SAME;
       co1.output, h1.input                         ARE_THE_SAME;
       h1.output, r1.input, r2.input                ARE_THE_SAME;
       r1.output, r2.output,v1.input                ARE_THE_SAME;
       v1.output,co2.input                          ARE_THE_SAME;
       co2.output, fl1.feed                         ARE_THE_SAME;
       fl1.liq, h2.input                            ARE_THE_SAME;
       fl1.vap, sp1.feed                            ARE_THE_SAME;
       sp1.out[1], h3.input                         ARE_THE_SAME;
       sp1.out[2],c2.input, s2.input                ARE_THE_SAME;
      c2.output, s2.output,m1.feed[1]               ARE_THE_SAME;
(* Conditional statements *)
       WHEN select_feed1
         CASE TRUE:
              USE f1;
         CASE FALSE:
              USE f2;
       END;
       WHEN select_single1
         CASE TRUE:
              USE c1;
         CASE FALSE:
              USE s1;
       END;
       WHEN select_cheapr1
         CASE TRUE:
              USE r1;
         CASE FALSE:
              USE r2;
       END;
       WHEN select_single2
         CASE TRUE:
              USE c2;
         CASE FALSE:
              USE s2;
       END;
END flowsheet;
```

FIGURE 6 ASCEND model for the superstructure of Figure 5

implementation of the WHEN statement and the logical relations. Its performance has been tested in several small problems. As an example, it was applied to the system presented in Figure 5 and Figure 6. The value of the four boolean variables will determine the structure of the problem to be solved. As mentioned above, the values of the boolean variables could be defined interactively by the user, but they also could be defined by some logic inference algorithm which would allow the automatic change of the structure of the problem. The NLP contains 137 invariant equations and 68 variant equations for a total of 205. The configuration defined by one of the feeds, two single-stage compressors and one of the reactors contains 169 equations, the 137 invariant and 32 active equations out of the 68 variant equations. Switching form one structure to another is done without the need of recompilation and, since to reconfigure the system requires only rebuilding several list of pointers, it is computationally very efficient.

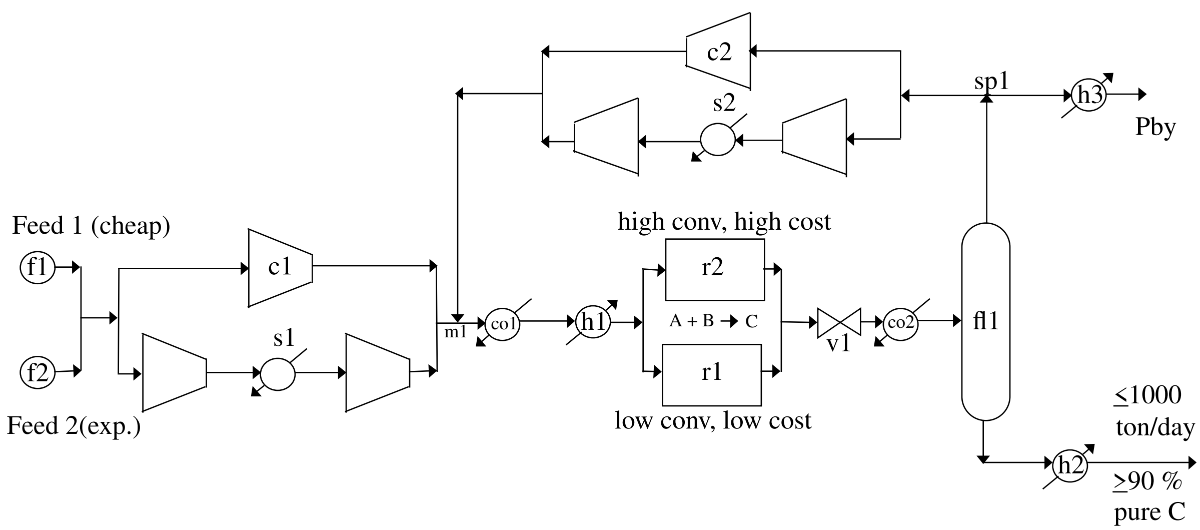


FIGURE 5 Superstructure taken from Turkay and Grossmann (1996).

4.2.2 Complementarity Formulation

Many examples of algebraic systems of disjunctive equations found in the literature have been represented and solved by using a complementarity formulation. In all the cases the results are encouraging. The solution of only one NLP is required and only a few iterations are used for convergence. However, it must be emphasized that, in all the cases, the number of equations in each disjunctive term (not the number of disjunctions) is very small; as a consequence, the complexity of the equations involving products among slack variables is not as bad as it can

Basically, a literature search has been conducted to find a possible solution for difficulty 3. To deal with nonconvexities in an optimization mode, Quesada and Grossman (1995) used linear relaxations based on the reformulation and linearization techniques for bilinear programming. Also, Viswanathan and Grossmann (1990) proposed augmenting the objective function with a penalty function that allows the violation of the linearizations of nonconvex constraints. A similar approach had been suggested by Bullard and Biegler (1991, 1992) in their iterated LP techniques for nonsmooth simulation. We are still not sure about how the application of these ideas could help to overcome the problems of having nonconvexities in the complementarity formulation. Research in this area is still ahead.

As mentioned before, the motivation for looking at a complementarity formulation is that only one NLP has to be solved. In general, we are presenting this approach as a very favorable alternative when the number of equations in each disjunctive term is small.

4 WORK TO DATE

4.1 MODELING

Both the WHEN statement and logical relations have already been incorporated to the ASCEND system as they were outlined above. We are currently testing this initial implementation, looking for improvements that could enhance its efficiency as modeling tools. To give an example of their application, we developed a simplified model for the superstructure given in Figure 5, taken from the work of Turkay and Grossmann (1996). In this example there are two alternative feedstocks, two possible choices of the reactor and two choices of the compression systems. Therefore, there are $2^4 = 16$ feasible configurations of the problem. All these 16 configurations are encapsulated in one ASCEND model containing 4 WHEN statements which depend on the value of 4 boolean variables. Figure 6 shows this model. The procedural section of the model and the model for each unit operation has been omitted for simplicity.

4.2 SOLUTION OF CONDITIONAL MODELS

4.2.1 Feeding a Conditional Model to an NLP solver

The mechanism suggested in section 3.2.2 has been implemented simultaneously with the

$$\begin{array}{l} h_1 \cdot h_2 + h_1 \cdot h_4 = 0 \\ h_3 \cdot h_2 + h_3 \cdot h_4 = 0 \end{array} \Rightarrow \begin{array}{l} h_1 \cdot (h_2 + h_4) = 0 \\ h_3 \cdot (h_2 + h_4) = 0 \end{array}$$

	h_1	h_2	h_3	h_4
1	$h_2 + h_4$	h_1	0	h_1
2	0	h_3	$h_2 + h_4$	h_3

FIGURE 4 Numerical singularities in factorable complementarity equations

the rows of the Jacobian corresponding to these equations. Note that the equations in Figure 4 contain the same four complementarity terms as the equations in (10), but they are grouped differently. In the case presented, if the solution of the problem is $h_2=h_4=0$, rows 1 and 2 of the Jacobian become numerically dependent as the Newton method approaches the solution (if $h_2+h_4 \sim 0$, we could get row 2 multiplying row 1 by the factor h_3/h_1). This is not the case in nonfactorable complementarity equations. There may be cases in which property 3 conflicts with property 4. Property 3 will have priority since it is what gives the simultaneous satisfaction of the equations in the same disjunctive term, a restriction that cannot be lost.

To get a formulation with all the properties outlined above is not a trivial exercise when the number of terms of the disjunction increases. However, for a disjunctive statement with two terms, the generalization to any number of equations is straightforward.

Many problems are incorporated with this kind of representation:

1. First of all, the problem grows quickly. If N is the number of equations in each term of the disjunction and D is the number of terms in the disjunction, the number of equations representing the disjunctive statement in the complementarity formulation is N(D+1).
2. The number of bilinear terms incorporated in each equation also grows with the number of equations in each term of the disjunction. The combinatorial nature of the problem is encapsulated here.
3. The performance of optimization techniques is badly affected by the introduction of nonconvexities (bilinear terms in the example) to the system of equations.
4. Numerical singularities can still arise for cases in which the solution resides on a boundary.

4. Every slack appears only once in each equation. There is no repeated slack index.

Posed in a formal way, given the disjunctive set of algebraic equations (two-term disjunction for simplicity)

$$\begin{bmatrix} f^1(x) = 0 \\ r(x) \le 0 \end{bmatrix} \vee \begin{bmatrix} f^2(x) = 0 \\ r(x) \ge 0 \end{bmatrix} \tag{11}$$

the problem is reformulated as

$$\begin{gathered} g(x,y) = \begin{pmatrix} f^1(x) - h_1 \\ f^2(x) - h_2 \\ r(x) - h_3 + h_4 \end{pmatrix} \\ g(x, y) = 0 \\ C(h_1, h_2, h_3, h_4) = 0 \\ h_i \ge 0 \qquad i \in [1 \ldots 4] \end{gathered} \tag{12}$$

where now *hi* represents a vector of slacks variables and $C(h_1,h_2,h_3,h_4)$ is used to represent the set of complementarity equations involving the elements of these vectors. Note that in the case of inequalities, the complementarity condition is expressed in terms of the slacks defined in the same equation. Also, we will require that the equations defined by $C(h_1,h_2,h_3,h_4)$ present the features outlined above for the equations given in (10). Property 1 is intended to avoid a change in the number of degrees of freedom, while property 3 is required to ensure the simultaneous satisfaction of all the equations in the same disjunctive term. Property 1 and 3 must always be satisfied. In our example, property 2 is only a consequence of 3 and 4 rather than a requirement, and may not be present in a general case.

Newton and quasi-Newton methods are the most popular techniques for the solution of NLP problems. Property 4 is intended to decrease the possibility of numerical singularities in the Jacobian which can occur during an iterative scheme based on these methods. If a repeated index occurs and, therefore, factorization is possible for some of the slacks, numerical singularities could arise as the iterative scheme approaches some solutions of the complementarity problem. This is illustrated in Figure 4, where we show a case of factorable complementarity equations and

$$Re - 64/f = h_1 \qquad Re - (0.206307/f)^4 = h_2$$
$$Re = 2100 + h_3 - h_4 \tag{7}$$

in terms of the slacks, the disjunctive statement becomes:

$$\begin{bmatrix} h_1 = 0 \\ h_3 = 0 \\ h_2, h_4 \geq 0 \end{bmatrix} \vee \begin{bmatrix} h_2 = 0 \\ h_4 = 0 \\ h_1, h_3 \geq 0 \end{bmatrix} \tag{8}$$

and the standard complementarity formulation is given by (7) and

$$\begin{aligned} h_1 \cdot h_2 &= 0 \\ h_3 \cdot h_4 &= 0 \\ h_i \geq 0 \qquad & i \in [1 \dots 4] \end{aligned} \tag{9}$$

It should be noted that the disjunctive statement requires either both h_1 and h_3 or both h_2 and h_4 simultaneously to be zero, and this is not a restriction in the complementarity formulation given by (9). The following formulation for disjunctive statements is proposed instead:

$$\begin{aligned} h_1 \cdot h_2 + h_3 \cdot h_4 &= 0 \\ h_1 \cdot h_4 + h_3 \cdot h_2 &= 0 \\ h_i \geq 0 \qquad & i \in [1 \dots 4] \end{aligned} \tag{10}$$

Since the slacks variables are all positive, the equations in (10) not only contains the complementarity condition given in (9) but also enforces simultaneous satisfaction of all the equations defined in the same term of the disjunction.

The complementarity equations given in (10) have the following properties:

1. The number of equations is equal to the number of equations in each term of the disjunction.
2. The number of terms in each equation is also equal to the number of equations in each term of the disjunction.
3. Every slack variable is multiplied by every slack in the other term of the disjunction.

implicitly or explicitly in such a CASE are set as active.

4. All the variables incident in the active set of equations are active.
5. The current problem to be solved consists of the active set of equations and variables.

The mechanism outline above is not precisely specific for the boundary crossing algorithm. For example, a branch and bound technique would also require the switching from one problem to another. Specifically for the boundary crossing algorithm, there is another major complication that must be addressed. While analyzing a boundary, the gradient step requires information about the equations in all of the subregions neighboring the boundary. That means the more than one set of equations has to be used simultaneously. We still do not address this and some other difficulties that we may not have discovered yet. However, based on previous programming exercises, we feel optimistic of being able to overcome any difficulty involved in this implementation in the time remaining for this research project.

3.2.3 Complementarity Representation of a Conditional Model

In this section, a complementarity formulation is proposed for the representation of conditional models. This formulation not only establishes the complementarity condition among alternative sets of equations but also enforces simultaneous satisfaction of all the equations in the solution set. The formulation proposed here is an extension of that presented by More(1994) who reformulates the nonlinear complementarity problem (3) as:

$$g(x,y) = \begin{pmatrix} f(x) - h \\ H \cdot x \end{pmatrix}$$

$$g(x, y) = 0 \qquad (5)$$

$$x \geq 0 \qquad y \geq 0$$

where H is the diagonal matrix diag(h_i). The next simple example will show how a complementarity formulation should change for the case of disjunctive statements. Consider the flow transition given by:

$$\begin{bmatrix} Re = 64/f \\ Re \leq 2100 \end{bmatrix} \vee \begin{bmatrix} Re = (0.206307/f)^4 \\ Re \geq 2100 \end{bmatrix} \qquad (6)$$

defining slack variables h for both the equalities and inequalities:

some parameters, such as cost factors, mass balance coefficients, power of a correlation, etc. In such a case, the need of a combinatorial structural analysis is eliminated.

2. The disjunctions of the problem are disjointed, and its structural analysis may be done independently. There are two cases in which this may happen *i)* the incidence pattern of the disjunction are totally decoupled, and *ii)* the existing coupling still allows the calculation of the variables output assigned to the disjunctive equations separately in some precedence order.

We have applied all these ideas for analyzing by hand small conditional problems, and we believe that their practical implementation is possible. A computer implementation which can take advantages of them is desirable.

3.2.2 Implementation of the Boundary Crossing Algorithm

A part of this project will consist in the implementation and testing of the boundary crossing algorithm. In general, it can be considered that both the Newton and gradients steps required in this algorithm are already well defined. We are looking, however, for ideas which could give the computational efficiency needed for its application in a general purpose environment like ASCEND.

We mentioned in section 3.1.1 that the implementation of the WHEN statement in ASCEND would allow us to have available the data structure for all the variables and equations of the conditional model. Therefore, the first step in the implementation of a conditional solver is to decide how the solver is going to be fed with the correct structure and how to change this structure efficiently in an iterative solution scheme. To perform this task, we will use the notion of active equation and active variable. By active we mean "it is part of the problem currently being solved". The following steps would provide a mechanism to select the structure of the problem:

1. Initially, all the equations resulting from the compilation are considered as active.
2. Then, all the equations which are implicitly or explicitly (inside models) stated in the CASE of a WHEN statement are set as inactive. The equations set as inactive in this step constitute the variant set of equations. All the equations which remain active constitute the invariant set of equations.
3. Analyze the WHEN statements. According to the current values of the variables on which each WHEN statement depends, determine which of its CASEs applies. The equations stated

crossing algorithm.

3.2 SOLVING CONDITIONAL MODELS

3.2.1 Structural Analysis

The analysis required for partitioning the variables and the equations in a conditional model was outlined in section 2.2.1. In our point of view, the most serious disadvantage of this analysis is the combinatorial nature of the search consistency algorithm, which requires the analysis of all the feasible subregions resulting from the dissection of the feasible region. We think that, in general, this analysis cannot be avoided. If we want to be sure about the consistency of the conditional model, we would have to pay the price. However, based on the work of Allen and Westerberg (1976), we also think that sometimes it is possible to take considerable advantage of the structure of the problem, reducing the effort required by such an analysis.

In chemical engineering design and simulation, systems modeled with indexed equations and variables occur very often. Typical examples are a distillation column (indexed by the number of stages) and systems of equations arising in the discretization of an initial value problem (indexed by the number of time steps). If we think of a problem in which a disjunction exists in each of the repeated structures, the combinatorial complexity in the structural analysis would be unmanageable. However, intuition suggests that, in this kind of problem, the degrees of freedom analysis should not be affected by the number of repeated structures and, therefore, the analysis could be simplified. In his work, Allen used a representative incidence matrix to perform a systematic analysis of indexed equation sets. Again, this could not be applied to an arbitrary system of disjunctive equations, but if we know a priori that our problem contains an indexed repeated structure, then we may take advantage of it.

We have also observed special features of some problems which can contribute to simplifying the analysis. Even though they represent very particular cases, if found alone or in any combination, a simple checking in the computational implementation of the algorithm could give us saves in time and in the computational effort needed to perform the analysis. These are:

1. The incidence pattern of the different terms of the disjunction is the same. It happens sometimes that the alternative equations are only different because the difference in value of

(a)

```
laminar     IS_A boolean;
Re,f,k      IS_A factor;

CONDITIONAL
      condition: Re <= 2100;
END;

laminar == SATISFIED(condition);

eq1: Re = 64/f;
eq2: Re = (0.206307/f)^4;

WHEN laminar:
      CASE TRUE:
            USE eq1;
      CASE FALSE:
            USE eq2;
END;
```

(b)

```
valve_open       IS_A boolean;
pump_on          IS_A boolean;
full_tank        IS_A boolean;

valve_open == pump_on AND NOT(full_tank);
```

FIGURE 3 Examples of the syntax for the implementation of logical relations

AND, OR, and NOT are allowed. Equality in a logical equation can also be interpreted as an *if and only if* implication between two logical terms expressed in clausal form. Also, it should be noted that we can express any logical clause using the proposed syntax by simply writing the clause in one of the terms, and the constant boolean value TRUE in the other term of the logical equality. Example (a) is a more complete version of the model presented in Figure 2. In this case it can be noted that, when the value of a boolean variable depends on the truth value of a real expression, the real expression is defined and labeled inside the CONDITIONAL statement; then the logical operator SATISFIED will give the truth value of such an expression. If the truth value of the real expression is TRUE, the result of SATISFIED(label of expression) will be TRUE. The objective of the operator SATISFIED is twofold: 1) it makes the life of the compiler easier, since the logical expressions are decoupled from the real expressions on which they depend, and 2) the use of this syntax forces the user to define a boolean variable (laminar in the example) which will have the same truth value as the real expression. That is useful for browsing purposes.

It is important to emphasize that, by checking the value of the appropriate boolean variables after each iteration, the incorporation of logical relations will allow, if required, automatic change of the structure of the problem in an iterative scheme such as the boundary

the value of the symbol method. Once a configuration has been select, it will be kept unless the user decides to change it. Note that the user does not have to recompile the model to make the switching, since the data structure for both of the problems is available and, therefore, the reconfiguration of the system can be done automatically by simply changing the value of the conditional variable. An example of this application would be the synthesis of process networks.

On the other hand, in case (a) we cannot expect the boolean value of the variable laminar to be a user decision. Its value will depend on the value of the Reynolds number which is an unknown in the problem. Actually, the value of the variable laminar will be the truth value of the expression Re$\leq$2100. Here we have a conditional model. In an iterative solution scheme, we will expect that the value of the boolean variable laminar will change, and so will the structure of the system of equations that we have to solve.

3.1.2 Incorporation of Logical Relations

Motivated by some recent applications of logic in the solution of conditional models (Raman and Grossmann, 1994; Turkay and Grossmann, 1996) and trying to expand the ASCEND capabilities for modeling, an implementation of logical relations as a modeling tool is proposed. To our knowledge, no other general purpose equation-based modeling environment provides this capability.

In the implementation of logical relations, the equation based approach is going to be maintained. That is, the user will state the logical relations that must be true at the solution to the problem but not how to solve them. Each logical equation will have a residual attached to it. This residual will indicate if the expression is satisfied or not. Therefore, this incorporation also means that we need to provide a solver which knows how to deal with logical relations. The work of Raman and Grossman (1993) can be used as a starting point for such a solver implementation. They used formal procedures for performing logic inference depending on the way in which the logical relations are present (CNF or DNF).

Figure 3 illustrates two examples of the syntax proposed for the implementation of logical relations. The symbol double equality will indicate that we have a logical relation and will separate the two logical terms of the relation. In each of the two terms, logical operators such as

whose name is given in each of the cases should be declared inside the model which contains the WHEN statement.

2 The variables in the list of variables can be of any type among boolean, integer or symbol or any combination of them. That is, we are not limited to the use of boolean variables. Obviously, The list of values in each case must be in agreement with the list of variables in the number of elements and type of each element.

3 Names of arrays of models or equations are also allowed inside the scope of each case.

The proposed extension (allowing the user to define the domain of validity of both models and equations inside the cases of a WHEN statement) enormously increases the scope of modeling in an equation based modeling environment. Of course, this implementation may be possible in other modeling environments which support object oriented concepts, but we have proposed it as an extension to the ASCEND modeling language because ASCEND is the best one locally understood. Figure 2 shows two incomplete ASCEND models in which the WHEN statement is used, each in a different manner.

```
      (a)
laminar      IS_A boolean;
Re,f,k       IS_A factor;

eq1: Re = 64/f;
eq2: Re = (0.206307/f)^4;

WHEN laminar:
      CASE TRUE:
            USE eq1;
      CASE FALSE:
            USE eq2;
END;
```

```
                (b)
method              IS_A symbol;
simplified_flash    IS_A VLE_flash;
rigorous_flash      IS_S td_VLE_flash;

WHEN method
      CASE 'rigorous':
            USE rigorous_flash;
      CASE 'simplified':
            USE simplified_flash;
END;
```

FIGURE 2 Two different applications of the WHEN statement.

In case (b) the value of the symbol method is used only to select between two alternative configurations of the problem, a flash calculation assuming constant relative volatility (VLE_flash) or a flash calculation using a more rigorous thermodynamic calculation (td_VLE_flash). The method that is going to be used and, therefore, the value of the symbol method, is a user decision. We can expect that the user will select the simplified model looking for good initial values of the variables and then he will switch to the rigorous model by changing

with the solution of these problems. Some issues involved in the implementation of the boundary crossing algorithm will be briefly explained, and an extension of a complementarity formulation to represent conditional models will be proposed.

3.1 MODELING

In this section, an extension to the ASCEND modeling language is proposed. Essentially, it consists of implementing the WHEN statement and incorporating logical relations into the system.

3.1.1 The WHEN Statement

Originally, the syntax for incorporating conditional dependence of some equations of the model in an equation-based environment was suggested by Piela (1989). That syntax is very similar to the CASE equation of *gPROMS* and suffers the same limitations. Instead, we want to take advantage of the fact that ASCEND is based on object oriented concepts where model definitions can contain parts that contain parts to any level. Furthermore, in ASCEND, a simple relation is treated as an object by itself and can have a name. Based on these ideas, the proposed syntax for the WHEN statement is:

```
WHEN list_of_variables
      CASE list_of_values_1:
                              USE name_of_equation_1;
                              USE name_of_model_1;
      CASE list_of_values_2:
                              USE name_of_equation_2;
                              USE name_of_model_2;
      CASE list_of_values_nminus1:
                              USE name_of_equation_nminu1;
                              USE name_of_model_nminus1;
     OTHERWISE:
                              USE name_of_equation_n;
                              USE name_of_model_n;
END;
```

The following are important observations about the proposed implementation:

1 We are not dealing with conditional compilation. We create and have available the data structures for all of the variables and equations in each of the models. This is actually a requirement for the solution algorithms of conditional models. All the models and equations

quite arbitrary while the other is the vector of variables. Many commonly occurring problems have a more general form:

$$f^1(x)^T \cdot f^2(x) = 0$$
$$f^1(x) \geq 0 \quad f^2(x) \geq 0 \tag{4}$$

This problem is called the vertical nonlinear complementarity problem (Ferris and Pang, 1995). It is possible, of course, to have more than two vectors of functions in the above equations. Later, we will propose the representation of disjunctive sets of algebraic equations as complementarity problems. It is important to mention that we are aware of many disadvantages associated with this representation and, therefore, potential solutions to these problems will be mentioned as well. Our motivation is the fact that a complementarity formulation only requires the solution of a nonlinear system of equations, avoiding all of the complications encountered in the boundary crossing algorithm and the discrete decision making of a MINLP solution.

2.2.3 Solution of Conditional Models in an Equation-Based Modeling Environment

Finally in this section, we consider again the analysis of the previous implementations of conditional modeling to mention their status in the solution of these problems. Both the IF construct of *SpeedUp* and the CASE equation of *gPROMS* are used only to dynamically change the active set of equations during a dynamic simulation by evaluating the logical expression at any point in time. They assume, however, that the initial state of the system (and hence, the initial active set of equations) is known a priori. In other words, they do not address the problem of finding a consistent set of initial values that satisfy both the logical conditions and the active set of equations.

3 PROPOSED CONTRIBUTIONS

One of the goals of our research group has been to improve the ability to develop and solve process models. One of the results of this effort has been the development of ASCEND (Piela *et al*., 1991). Westerberg *et al.* (1994) discuss the essential features of the current ASCEND system and present several ways in which we can improve it to solve larger models and to increase its scope as a modeling environment. In section 3.1 we will propose the implementation of tools which will allow ASCEND to represent conditional models efficiently. Section 3.2 is concerned

In the gradient analysis at boundaries, a descent direction for the objective functions (norm of the residuals of the equations) local to each neighboring subregion is required for convergence. This direction d will be given by:

$$\begin{aligned} \min_{\alpha} \quad & d^T \cdot d \\ \text{s.t.} \quad & d = N \cdot \alpha \\ & e^T \cdot \alpha = 1 \\ & \alpha \geq 0 \end{aligned} \tag{2}$$

where the matrix N is known and is used to store the vectors of the gradients of the objective functions local to each neighboring subregion which borders a boundary, and e is a vector with all elements equal to one. The vector α, whose elements sum to one, is used to generate the direction d as the linear combination of the gradients which yields a vector of smallest length. Termination occurs if a minimum value of zero is obtained for d, which means that the gradient vectors of N cannot be confined to one side of a half-plane.

2.2.2.2 Complementarity Formulation

The class of problems known as complementarity problems has become increasingly popular as a tool for addressing practical problems arising in mathematical programming, engineering, and the sciences (Billups, 1995; Ferris and Pang, 1995). Dirske and Ferris (1995) describe the standard forms for the different classes of complementarity problems and give examples of how to formulate many popular problems as mixed complementarity problems (MCP). Billups (1995) proposes strategies which enhance the robustness of Newton-based methods for their application in these problems.

In general, the nonlinear complementarity problem is expressed as the following set of equations and inequality constraints:

$$\begin{gathered} x^T \cdot f(x) = 0 \\ x \geq 0 \qquad f(x) \geq 0 \end{gathered} \tag{3}$$

There is certain lack of symmetry in the previous formulation. One of the functions is

problem that is often solvable as a relaxed LP. They consider the nonlinear case although no guarantee of convergence is given for it. Also, based in the works of Raman and Grossmann (1993, 1994), Turkay and Grossmann (1996) address the solution of disjunctive set of nonlinear programming problems. It is shown that the use of logic can improve the efficiency and robustness of MINLP algorithms on the solution of structural optimization problems.

It is worth summarizing some of the observations and conclusions about MINLP algorithms arisen from the work of Zaher (1995): 1) it can be expected that during the solution of a MINLP, the algorithm will jump from one subregion to another which is remote from it, leading to the poor initialization of some of the nonlinear subproblems, 2) the nonlinear subproblems insist in converging the equations of what might be an incorrect subregion and 3) this method employs a highly mathematical approach which remains as a rigorous general purpose algorithm designed for problems where discrete decision making cannot be avoided.

The following two sections summarize two alternative approaches to the solution of conditional models.

2.2.2.1 Boundary Crossing Algorithm

Zaher (1991) introduced the boundary crossing algorithm as an alternative to the MINLP formulation for the solution of conditional models. In this approach, the inequalities defining the domain of each set of variant equations are considered as logical conditions which, when satisfied, will identify a particular subregion and, if violated, will identify an alternative one. In that way, every equation $r_{ik}(x)=0$ constitutes a boundary which will separate the feasible region in two neighboring subregions. In general, identification of a subregion requires a unique combination of boolean values for the collection of all logical conditions. If l is the number of boundary equations, as many as 2^l subregions can be created.

The boundary crossing algorithm has been given in Zaher (1991). Basically, the algorithm applies a Newton step for all the points not residing at boundaries. If during the iterative process a boundary is reached, then a Newton step cannot be taken because the system is nondifferentiable at that point (different subregions have different set of equations). Therefore, a gradient analysis is performed instead.

problem can be decomposed in smaller partitions which can be individually solved in some precedence order. Structural analysis is an indispensable tool in an equation-based modeling environment, and as it will be shown, conditional models make the need for this tool an even stronger requirement.

In conditional models, the sparsity pattern is expected to change from one subregion to another. This implies that a consistent set of independent variables for one subregion may not be valid for another subregion. For the same reason, block partitioning of the equations for one region may not be possible using the equations of another.

Zaher (1995) addressed the structural analysis of conditional models. A necessary condition for structural consistency in conditional models is that each of the alternative sets of equations must be structurally consistent. Hence, consistency of a conditional model is assessed by finding at least one consistent partitioning of the variables (independent-dependent) such that output assignment of all of the equations in each subregion can be performed. This requirement makes the problem combinatorial, since we have to perform the analysis for all of the possible alternative set of equations.

Regarding the equation partitioning, it is recommended one to obtain a representative incidence matrix rather than attempt to incorrectly decouple a wrong set (not the solution set) of equations. This representative incidence of equations in a disjunctive statement can be achieved by 'OR'ing the incidences of the inequalities (or boundary expressions) with each locally defined equation. Then, additional incidences are obtained for each equation in a subregion one at a time by merging the incidences of the equation with those of exactly one equation from all of the other subregions. It should be pointed out that it is also possible and valid to obtain an individual block partitioning while analyzing any particular subregion, and that the previous algorithm was proposed by Zaher when seeking efficiency.

2.2.2 Solution Algorithms for Conditional Models

As mentioned earlier, conditional models can be formulated as mixed integer programming problems. In a very recent work, Grossmann and Turkay (1996) show that, in the case of linear equations, the solution of a conditional model can be found by solving a MILP

2 BACKGROUND

2.1 BUILDING A CONDITIONAL MODEL

Previous implementations of conditional statements in an equation-based modeling environment have been reported. One such mechanism is the IF-THEN-ELSE construct of *SpeedUp* described by Pantelides (1988):

```
IF logical_condition THEN
      equation1
ELSE
      equation2
ENDIF
```

where both the logical conditions and the equations are expressed in terms of the model variables. Such a construct defines two system states corresponding to the IF and the ELSE clauses respectively. Multiple states may be described by nesting several IF statements.

Similarly, Barton (1992) and Barton and Pantelides (1994) incorporated the CASE equation to *gPROMS*. The CASE equation is used to define both the appropriate modeling equations in each state and the logical conditions for transitions among states. It covers multiple states within only one statement and has the advantage of successfully representing irreversible discontinuities.

There is a major difficulty in both of the previous approaches as tools for conditional modeling. They only allow the substitution of a list of equations (or arrays of equations) for another. In an equation-based modeling environment in which concepts like hierarchy (building complex models from small models) and inheritance (adding new features to a model) are constantly in use, that represents a severe disadvantage in efficiency. Thinking of how many equations have to be written down when substituting some of the units of a superstructure makes obvious the lack of flexibility in the scope of these approaches. Later in this paper, we will show our extension to these modeling capabilities with the incorporation of the WHEN statement to the ASCEND modeling environment.

2.2 SOLVING A CONDITIONAL MODEL

2.2.1 Structural Analysis for Conditional Models

Before attempting to solve a model, a structural analysis has to be performed to determine if the formulation is well posed. Similarly, a partitioning algorithm is applied to find if the original

as determined by the simultaneously calculated value of the flow type indicator (Reynolds or Mach number) relative to some critical value.

While many currently available equation-based modeling systems have been reported in the literature, only a few of them have given attention to conditional models (Pantelides, 1988; Piela,1989; Barton, 1992). Several problems arise when building and testing models of complex processes in an equation-based environment (providing initial guesses, analyzing degrees of freedom, efficiently computing solutions, etc.). The combinatorial nature of a conditional model makes the solution of these problems even a more difficult task. In this work, we propose the incorporation of a series of tools which enables the user of an equation-based modeling environment with the capability of dealing with complex conditional models. It is important to distinguish between two well defined and separated (but complementary) goals of this research project:

1. Efficient representation of a conditional model in an equation-based modeling environment. This is intended to be independent of any particular application, or of any solver or algorithm used for finding a solution to the system of equations. Potential applications of this capability varies from the simple substitution of one equation for another (as in the case of the laminar-turbulent flow transition), to the substitution of a section of a chemical plant for another (as can be required while analyzing and initializing a superstructure).
2. Implementation and testing of alternative approaches to the solution of a conditional model. Modeling is a user dependent task. Frequently, the same problem is formulated differently from one modeler to another and, as a consequence, different approaches and techniques could be used to find a solution to it. Conditional models can be formulated as mixed-integer programming problems. See for example Grossmann and Turkay (1996). Besides the mixed-integer formulation, some alternative approaches exist which, in principle, could avoid the need of discrete decisions. Research in this work will focus in two of them: the boundary crossing algorithm (Zaher, 1991; Zaher, 1993; Zaher, 1995) and the complementarity formulation (Billups, 1995). In particular, we will propose an approach for the implementation of the boundary crossing algorithm in an equation-based environment, and an extension to the standard complementarity formulation for the representation of conditional models.

1 INTRODUCTION

The design and simulation of engineering processes require one to find the solution to a large system of linear and/or nonlinear equations. This work is particularly concerned with the representation and solution of conditional models. Conditional models exist when the defining equations of the system depend on where the model solution lies. A conditional model consists of a system of equations expressed by two sets: a globally defined invariant set of equations and a variant (or locally defined) set of conditional equations which are expressed as disjunctions. Basically, as shown by Zaher(1993), and Grossmann and Turkay (1996), this problem reduces to representing and finding the solution to the system of disjunctive equations:

$$
\begin{gathered}
g(x) = 0 \\
\bigvee_{i \in D_k} \begin{bmatrix} h_{ik}(x) = 0 \\ r_{ik}(x) \le 0 \end{bmatrix} \quad k \in K \\
x \in R^n
\end{gathered}
\qquad (1)
$$

where $g(x)$ and $h_{ik}(x)$ represent the invariant and the variant sets of equations respectively, K represents the set of disjunctions and the index i is used to indicate the i-th term in each disjunction D_k. The equations $g(x)$ can be said to be defined over the entire feasible region, while the inequalities define the domain of validity of each variant set of equations $h_{ik}(x)$. In that way, each variant set of equations is confined to some subregion resulting from the dissection of the feasible region (Zaher, 1991). The solution of the system will be given by the vector x satisfying the invariant set $g(x)$ and exactly one set of equations for each of the disjunctions, providing that the corresponding set of inequalities is satisfied.

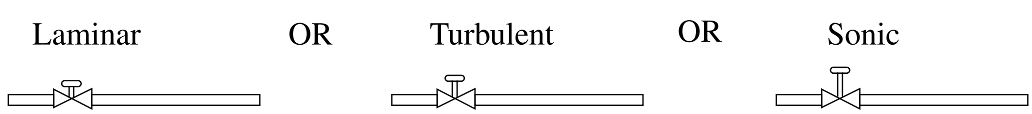


FIGURE 1 Fluid flow transition

Fluid transition represents an example commonly encountered in chemical engineering. This is illustrated in Figure 1 taken from the work of Zaher (1995). In this case, alternative equations for the transport properties and velocity bounds must be incorporated within the model

Representing, Analyzing and Solving Conditional Models in an Equation-Based Environment

by Vicente Rico-Ramírez

Advisor: Arthur W. Westerberg

Ph.D. Thesis Proposal

Department of Chemical Engineering

Carnegie Mellon University

Pittsburgh, PA

January 15, 1997

Abstract

Conditional models arise in chemical engineering when modeling systems involving physicochemical discontinuities, such as phase transitions. There are also cases where conditional cost functions are used in the optimization of process systems. Modeling tools for the efficient representation of conditional models in an equation-based environment are proposed in this paper. Also, some of the issues involved in the solution of these problems are discussed. The boundary crossing algorithm and the complementarity formulation are identified as alternatives to the MINLP algorithms for the solution of conditional models. In particular, we will propose an approach to the structural analysis of a conditional model, an extension to the standard complementarity formulation for the representation of conditional models, and an approach for the implementation of the boundary crossing algorithm in an equation-based environment.