Verification of a Chemical Process Leak Test Procedure

Adam L. Turk, Scott T. Probst and Gary J. Powers
Department of Chemical Engineering
Carnegie Mellon University

Abstract

A leak test procedure for a combustion system which is used in the chemical industry was verified. This procedure is important since it reduces the probability of explosions. Both government and internal company standards where employed in creating the initial leak test procedure. Several major faults were discovered by the verification of a logic model of the procedure and equipment using SMV. This paper describes the leak test procedure with its corresponding combustion system pipe network, the approach employed in modeling the process, failure modes included in the process model, computational challenges, and verification results. This study indicates that the formal method, SMV, is an appropriate tool for verification of industrial processes of modest complexity

1. Introduction

The synthesis of high integrity industrial processes is becoming a more significant goal of companies. A small industrial process may contain 50 to 100 binary variables in its model. The total combinatorial space of a process model with 50 binary variables has 10^{15} states. Analysis of even a modest size system is difficult with current industrial practices, such as simulation and field testing. Automated formal methods can exhaustively verify large state spaces, typically, on the order of 10^{20} states [2], for a given property of the system.

Originally, formal methods, in particular symbolic model verification (SMV) [11], were developed for the verification of integrated circuits and communication protocols [5,6]. The power of verification using SMV is its ability to find faults in very large, sequential and finite state spaces. This ability comes from symbolically representing the set of states and state transitions. SMV internally converts a process model into a symbolic representation, an ordered binary decision diagram, which implicitly describes the state space. Recently, several industrial processes including a batch reactor, a solids transport system and a furnace standard, have been represented in a logic model and verified by SMV [1. 9, 12, 13, 14, 15, 16, 17]. One problem of verification that has laid dormant is the efficient modeling of a complete and compact representation for an industrial process. This paper describes the modeling and verification of a part of a chemical process that involves a leak test procedure and is intended to test the ability of SMV to effectively verify industrial processes. This example illustrates the importance of selecting an appropriate modeling framework to capture necessary process behavior while controlling computational difficulties.

2. Process Verification

Process flowsheets, operating procedures, control logic, human operators, and failure modes are needed to describe the behavior of a chemical process. The compactness and completeness of a model significantly influences the quality of the analysis and the computational time required for the verification. Excessively complicated models can often described a state space that is too large for efficient verification. For process verification, an appropriate level of complexity and behavior must be captured by the logic model.

The process specifications and key process properties are used as guides in identifying the behavior and complexity necessary to represent the process. These behaviors are appropriately modeled with the aid of several comprehensive strategies. The strategies involve:

- Exclusion of process behavior that does not effect the specifications being verified.
- Discretization of the continuous and dynamic process behavior into a set of finite transition relations.
- Representation of different time domains implicitly in order to reduce model complexity.
- Minimization of the number of state variables required to represent the system.
- Application of symmetry [7] and modularity in order to reduce the model complexity.
- Application of non-determinism selectively in modeling uncertain properties of the process.
- Combination of similar failure events into macro failure modes.

The union of these strategies is an attempt to tame combinatorial explosion by efficiently building logic models while remaining true to the underlying physics, chemistry, control logic, and operating goals of the process. An understanding of the physical phenomena found in the process must be developed in order to identify the behavior that may have been omitted or spuriously included.

3. Leak Test Procedure Description

The correct detection and replacement of leaking valves in a combustion system can lead to the reduction in the number of explosions from methane escaping into a shutdown furnace. The leak test procedure is a series of steps that check for leaks across shut off valves. The valve diagram for a combustion system is shown in figure 1. The original procedure (base case) pressurizes the system by igniting the pilot and main burner. Once the burners have been ignited then the hand valves *bv7* and *bv15* are closed which extinguishes the burners. The blocking valves, *ll4*, *bl3*, *ls12*, and *bs11* automatically close since they are linked to a safety interlock which is triggered when the burners are extinguished. The closure of these valves creates a series of pressurized pipe segments between the valves. Pressure in each of these segments is assumed by the test procedure to leak down stream to a lower pressure location. The system is allowed to reach equilibrium by waiting 5 minutes before proceeding with testing of the system. The main line is checked first by opening the tap valves *tp3* and then *tp2*. Each section is checked for bubbling, which indicates no leaking downstream, or for bubbling to stop, which indicates no leaking upstream. The pipe segments in-between the locking valves, l14, b13, ls12, and bs11, are tested for both conditions. If at any time, a leak is detected then the apparently leaking valve is replaced and the test procedure is started over. Once the main line has been completed the pilot line is opened allowing this section of the valve train to be re-pressurized before continuing with the procedure. The test steps outlined for the main line are repeated for the pilot line.

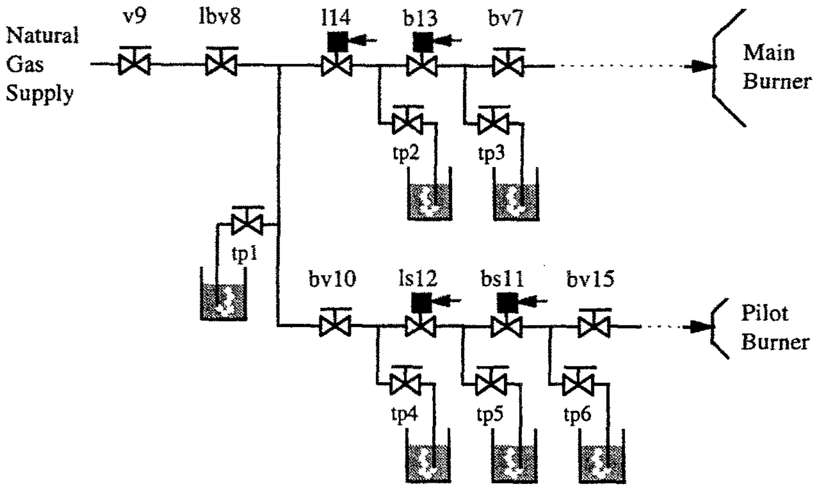

Figure 1: Piping and Valve Diagram for Combustion System

4. Properties

The properties for the process were created from quality, operability, and safety issues that concern process engineers [8,10]. General specifications developed from these issues are given below:

- Are leaking valves detected?
- Are non-leaking valves being replaced needlessly?
- Does the procedure terminate?

Specifications which represent process concerns were used as guides in abstracting and constructing process logic models.

5. Modeling of the Leak Test Procedure

5.1 Modeling Continuous Behavior

In order for the model to be comprehensive, the underlying physical phenomena of a leak must be understood either from experimentation or theory. The theory for the dynamics of a compressible fluid, such as methane gas, is well established and is based upon the following equations of continuity (eqn. 2), motion (eqn. 3), and energy (eqn. 4).

$$\frac{D\rho}{Dt} = -\rho(\nabla \bullet v) \qquad (2)$$

$$\rho\frac{Dv}{Dt} = -\nabla p - [\nabla \bullet \tau] + \rho g \qquad (3)$$

$$\rho\frac{D}{Dt}(\hat{U} + \frac{1}{2}v^2 = -(\nabla \bullet q) + \rho(v \bullet g) - (\nabla \bullet pv) - (\nabla \bullet [\tau \bullet v]\,) \qquad (4)$$

where ρ is the density,
v is the velocity,
p is the pressure,
τ is the shear forces.
g is the gravitational constant,
q is the heat flux vector,
$\hat{U}$ is the internal energy, and

A leak between two pipe sections can be simplified into two pressure vessels connected with a valve or nozzle. The effects of the nozzle on velocity and pressure in the radial direction can be removed by taking a macroscopic view of the system. It can also be assumed that no work is being performed on or by the system. Based upon these assumptions, the partial differential equation for motion can be simplified further by assuming that the variables in the z-axis do not change in the other axis directions.

$$\rho(\frac{\partial v_z}{\partial t} + v_z \frac{\partial v_z}{\partial z}) = -\frac{\partial p}{\partial z} - (\frac{1}{r}\frac{\partial}{\partial r}(r\tau_{rz}) + \frac{1}{r}\frac{\partial \tau_{\theta z}}{\partial \theta} + \frac{\partial \tau_{zz}}{\partial z}) \qquad (5)$$

This system of equations still can not be solved analytically for the leak rate of the methane gas. However, the leak procedure does not examine the leak rate of a valve directly, but checks for bubbling and/or the bubbling to have stopped from the pipe section being tested which in turn confirms the tightness of the valves isolating that pipe segment.. The rate of the gas escaping and therefore the bubbling is continuous in behavior and not discrete as the written procedure perceives it. Based upon the procedure, the system was modeled to reflect this discrete observation of bubbling and bubbling stopped. The model assumes that the operator checks for bubbling almost instantaneously after opening the tap valves and for bubbling to have stopped after a suitable time period which is not explicitly defined by the procedure. Both of these assumptions limit the behavior captured by the logic model and by the procedure. Faults caused by leak rates slower then the time period set for bubbling to stop can not be verified with the base case model. Modifications discovered during SMV verification of the procedure were added to the model which allowed this behavior to be captured.

The leak rate of a valve affects the amount of pressure remaining in a pipe segment. The altered pressure causes the time period necessary for the pipe segment to bleed to atmospheric pressure to be shorter or longer. The model of the process assumes that the leak will not decrease or increase the time needed to normally bleed a pipe segment. However, this assumption raises the questions of how much delay time is required for detecting bubbling to stop and how small of a test time interval is needed to isolate these leaks. Both the temperature of the environment and the current pressure of the methane source will affect the delay time. These interactions makes the selection of an explicit delay time difficult. This problem can be solved by allowing the transitions to the bubbling and bubble stopped states to represent different time periods.

The time period between states was allowed to vary according to the task being performed during the transition. The only time requirements were a five minute delay, the instantaneous checking for bubbles, and the suitable delay for bubbles to stop. The verification of this leak test procedure benefits from the flexible transition time periods since it represents any leak test procedure that specifies the time necessary for an operator to wait before he can assume that the bubbling has not stopped. Time was also implicitly included in the model through the variable step which indicates the current step an operator is performing in the leak test procedure.

5.2 Modeling Pressure

Pressure was declared as a Boolean variable as it is treated in the leak test procedure. This variable as described in the procedure only needed values for atmospheric pressure at which no bubbling would occur, and a higher pressure at which bubbling would occur. A local pressure variable was created for each segment of pipe. These local variables were connected through a series of definition variables. The transition relation of these definition variables would link the pressure in the pipe segments together if they shared a common leaking or open valve. The value of the definition variable for local pressure in a pipe segment would then be mapped into the correct local pressure variable. This modeling strategy assumes the pressure to quickly equalize through the pipe network due to leaks in the connecting valves. These modeling assumptions can be traced back to the defining equations 2, 3, and 4 for validation.

5.3 Modeling the Procedure

The leak test procedure, itself, was modeled using two state variables, *step* and *status*. The variable *step* was declared as an integer range between 0 and 17 while *status* was declared as four discrete values; conducting the leak test (test), repairing a valve (repairing), valve repair done (repair_done), start pilot burner (pilot_start), start main burner (main_start), and combustion system operational (burn).

$$step = \{0..17\} \tag{6}$$

$$status = \{\text{testing}, \text{repair}, \text{repair\_done}, \text{pilot\_start}, \text{main\_start}, \text{burn}\} \tag{7}$$

The transition relation of the variable, *step*, defines the various tasks of the test procedure and their ordering in the model (fig. 2). At each step *n*, several logic constraints need to be satisfied before *step* can progress to the next interval, *n+1*. Step one allows the pressure in the pipe network to come to equilibrium before the tap valve, *tp3*, is opened at the next step. Step 4 checks for the process condition of bubbling at *tp3*. Step 4 opens *tp2* and checks for bubbling at the next pipe segment. The condition of the bubbles stopping is tested in step 5. The tasks, performed in these steps are repeated for each pipe segment in the network. If no bubbles were seen by the human operator performing the test procedure at step 3, then the variable, *status*, initiates the repair sequence for valve *bv7*.

$$\text{step}' = \begin{cases} 0 & \text{if status} = \text{repair\_done} \\ \text{step} & \text{if status} = \text{repair} \\ 1 & \text{if step} = 0 \wedge \text{status} = \text{test} \\ 2 & \text{if step} = 1 \\ 3 & \text{if step} = 2 \\ 4 & \text{if step} = 3 \wedge \text{bubbling@ tp3} \\ 5 & \text{if step} = 4 \wedge \text{bubbling@ tp4} \\ 6 & \text{if step} = 4 \wedge \text{bubbles\_stop@ tp4} \\ 7 & \text{if step} = 6 \\ . & \\ 17 & \text{if step} = 16 \wedge \text{bubbles\_stop@ tp1} \\ 0 & \text{if step} = 17 \\ \text{step} & \end{cases}$$

Figure 2: Logical Abstraction of the Transition Relation for Variable *Step*

The *status* variable defines the transition of the process from the test procedure to other conditions such as repairing a valve or igniting the combustion system (fig. 3).

5.4 Modeling Failure Events

Failure Events for leaking valves were included in the logic model. State variables for valve leaks were defined for each valve that could leak. The initial conditions of these variables were made non-deterministic in order to capture the possible random failures of one or more values.

$$leak@bv7 = \{0,1\} \tag{8}$$

$$leak@bv7' = \begin{cases} 0 & \text{if } status = \text{repairing} \wedge step = 3 \\ leak@bv7 & \end{cases} \tag{9}$$

Variables retained their selected value until they were repaired at the correct point in the test procedure. Intermittent leaks were not included in these failure modes.

$$\text{status}' = \begin{cases} \text{test} & \text{if status} = \text{burn} \vee \text{step} = 6 \\ \text{repair} & \text{if step} = 3 \wedge \neg\text{bubbling}@\text{tp3} \\ \text{repair} & \text{if step} = 4 \wedge \neg\text{bubbling}@\text{tp4} \\ \text{repair} & \text{if step} = 5 \wedge \neg\text{bubble\_stop}@\text{tp4} \\ . & \\ \text{repair} & \text{if step} = 16 \wedge \neg\text{bubble\_stop}@\text{tp1} \\ \text{repair\_done} & \text{if status} = \text{repair} \\ \text{pilot\_start} & \text{if step} = 17 \vee (\text{step} = 0 \wedge \neg\text{pilot\_flame}) \vee \text{step} = 5 \\ \text{main\_start} & \text{if status} = \text{pilot\_start} \wedge \text{pilot\_flame} \\ \text{burn} & \text{if } \neg\text{status} = \text{test} \wedge \text{pilot\_flame} \wedge \text{main\_flame} \\ \text{status} & \end{cases}$$

Figure 3: Logical Abstraction of the Transition Relation for Variable *status*

6. Results

Table 1 gives the number of Boolean variables, reachable states, nodes in the transition relation, and the CPU time for the base case leak test models developed by Probst [16]. The time for verification and counterexamples generation is included in the values reported for the CPU time. These models revealed the general fault that leaks in the tap valves are not accounted for in the procedure and could cause non leaking valves, *bv7* and *b13*, to be replaced needlessly. In order to avoid this error, the tap valves in the pipe network must be tested prior to conducting the leak test procedure. The authors of the procedure modified the leak test method to include this additional test.

6.1 New Failure Modes

The subsequent models of the leak procedure were amended in order to captures both fast and slow valve leaks. The time scale of fast leaks was assumed to be seconds while the scale for slow leaks was assumed to be on the order of several minutes. The logic model does not need modifications in order to verify the procedure for fast leaks since the time

Table 1 Information on Verified Leak Test Procedure Models

Model Name	Boolean Variables	Reachable States	Transition Relations (OBDD Nodes)	CPU Time (sec)[a]
Base Case	24	5,944	11,221	5
Leak_tp1	25	11,859	14,804	2
Leak_tp1,tp2	26	22,263	18,734	4
Leak_tp1,tp2, tp3	27	42,154	22,961	5
Leak_tp1,tp2, tp3,tp4	28	84,313	28,215	26
Leak_tp1,tp2, tp3,tp4,tp5	29	169,049	33,701	39
Leak_tp1,tp2, tp3,tp4,tp5,tp6	30	340,193	39,087	52

[a]Computations were performed on a Hewlett-Packard 715/75 workstation

required for the leak to reduce the pressure in a particular pipe segment to atmospheric is assumed to be fast when compared with the other time constants of the system. The slow leaks are modeled by allowing the transition relation of *step* to skip testing pipe sections and their corresponding valves. It was assumed that the human operator could not observe the slowly leaking valves and this phenomena was treated as if there was no leaking valve (or equivalently that the pipe section was skipped). The pressure in the skipped pipe segment was still affected by the leaking valve and adjusted accordingly.

$$
step' = \begin{cases}
0 & \text{if } status = \text{repair\_done} \\
step & \text{if } status = \text{repair} \\
1 & \text{if } step = 0 \wedge status = \text{test} \\
2 & \text{if } step = 1 \\
\{3,4\} & \text{if } step = 2 \\
\{4,6\} & \text{if } step = 3 \wedge bubbling@tp3 \\
5 & \text{if } step = 4 \wedge bubbling@tp4 \\
6 & \text{if } step = 4 \wedge bubbles\_stop@tp4 \\
7 & \text{if } step = 6 \\
. & \\
17 & \text{if } step = 16 \wedge bubbles\_stop@tp1 \\
0 & \text{if } step = 17 \\
step &
\end{cases}
$$

Figure 4: Logical Abstraction of the Transition Relation for Variable *step* with Skipping

Table 2 Information on Verified Leak Test Procedure Models

Model Name	Boolean Variables	Reachable States	Transition Relations (OBDD Nodes)	CPU Time (sec)[a]
Base_Case	24	5,944	11,495	2.80
Press_Skip1	24	14,611	14,205	7.86
Valves_Tap	30	14,611	28,102	25.65
Valves_All	38	14,611	95,100	323.49
Valves_Two	32	14,611	54,064	128.31
Remove_Press	32	14,552	37,632	71.48
Expand_Proc	32	5,153	44,688	6.45
Press_Skip2	32	15,984	49601	18.3

[a]Computations were performed on a Hewlett-Packard 812/70 workstation

Failure events for slowly leaking valves and for the human operator skipping pipe segments and not testing them for leaks were added to the logic model through a macro failure mode Both of these failure events lead to the result of a leaking valve not being detected. This failure mode was included in the model by adding non-determinism to the transition relation of the variable, *step* (fig. 4). Before testing for pressure in each pipe segment, the procedure was allowed either to continue checking the current pipe section or to skip over it.

The *Press_Skip1* model added the step skipping failure mode to the base case model (table 2). The models *Valves_tap* through *Expand_Proc* contain a series of sequential refinements to the base case model which added resolution for capturing the combined failure event. The *Valves_Tap* model redefined the tap valves as state variables instead of definition variables to see if sequential behavior of these variables was important. *Valves_All* includes all the connecting and tap valves as state variables. In table 2, the data indicates that the size, complexity level, and CPU time of the models was increasing in an exponential manner. Strategic reduction in the model was used to control complexity. The *Valve_Two* model kept all connecting valves as definition variables except for *lbv8* and *bv10,* which were changed to state variables. These valves were left as state variables since they are explicitly closed in the test procedure. *Remove_Press* simplified the pressurization and segmenting of the pipe network to one macro step (fig. 5) instead of several complicated steps since the leak test procedure did not check for faults in both the ignition and extinguish sequence of the furnace. The skipping step model uncovered that the original modeling of the leak test procedure added invalid process behavior. The representation was incrementally refined in order to eliminate this added behavior until its size and complexity began to have a negative effect on verification. A series of reductions in the representation of the model were incorporated into the leak test model. These models illustrates the struggle to balance the completeness of the model with its compactness.
Through the expansion and reduction of the original model, it was discovered that some of the procedure steps overlapped and could not be distinguished from each other. The amendments to the representation allowed the user to distinctly perceive the individual steps of the procedure. The *Expand_Proc* model expanded the test procedure so that only one task was performed at every step which increased the number of steps from 17 to 24 (fig. 6). The *Expand_Proc* model increased verification time by a factor of 2 compared to

$$status' = \begin{cases} \text{test} & \text{if } status = \text{pressurize} \\ \text{repair\_done} & \text{if } status = \text{repair} \\ \text{repair} & \text{if } step = 3 \wedge \neg bubbling@tp3 \\ \text{repair} & \text{if } step = 4 \wedge \neg bubble\_stop@tp3 \\ . & \\ \text{repair} & \text{if } step = 23 \wedge \neg bubble\_stop@tp1 \\ \text{presurize} & \text{if } step = 0 \vee step = 8 \\ status & \end{cases}$$

Figure 5: Logical Abstraction of the Transition Relation for Variable *status* with a Macro Pressurization step

the base case. However, a factor of 10 increase in the verification time was observed for *Valve_All* model before the model size and complexity was reduced. Overall, the new representation from the one in the base case model increased verification time but not as dramatically as the trend indicated by the *Valve_All* model due to reduction and modeling strategies.

The final model, *Press_Skip2*, added the combined failure mode event to the *Expand_Proc* model. The *Press_Skip2* model revealed that slow leaks in the valves *bv7*, *bv15*, and *lbv8* were neglected by the test procedure. A correction to the base case test procedure ensured that all other connecting valves were double tested by the pipe segment downstream and upstream from them. This double checking by the test procedure detected when these valves had slow leaks. The procedure was revised by explicitly checking for the starting and stopping of bubbling at each tap. These changes effectively detect slowly leaking valves.

$$step' = \begin{cases} 0 & \text{if } status = \text{repair\_done} \\ step & \text{if } status = \text{repair} \\ 1 & \text{if } step = 0 \\ 2 & \text{if } step = 1 \\ 3 & \text{if } step = 2 \\ 4 & \text{if } step = 3 \wedge bubbling@tp3 \\ 5 & \text{if } step = 4 \wedge bubbles\_stop@tp3 \\ 6 & \text{if } step = 5 \\ 7 & \text{if } step = 6 \wedge bubbling@tp4 \\ . & \\ 24 & \text{if } step = 23 \wedge bubbles\_stop@tp1 \\ 0 & \text{if } step = 24 \\ step & \end{cases}$$

Figure 6: Logical Abstraction of the Transition Relation for Variable *step* with Expanded Pipe Segment Testing

7. Conclusions

The formal verification of the leak test procedure, a modestly complex chemical process, discovered several significant faults that were repaired by revising the procedure. A modeling strategy that controlled the state space size while retaining process physics, procedures, and failure modes was necessary for efficient and reasonably complete verification. The original process model was refined in order to capture appropriate behavior caused by increasingly detailed failure events. Process verification using SMV has demonstrated its potential for aiding in the synthesis of high integrity industrial processes. Modeling time and completeness remain as challenges to wider application.

Model Source

The base case model by Probst is available at:

http://www.cheme.cmu.edu/who/faculty/powers/probst/html/research.html

References

[1] R. Anderson, P. Beame, S. Burns, W. Chan, F. Modugno, D Notkin, and J. Reese, Model Checking Large Software Specifications. Proceedings of the Fourth ACM Symposium on the Foundation of Software Engineering: 156, 166, October, 1996.

[2] Bryant, R. E., "Graph-Based Algorithms for Boolean Function Manipulation", *IEEE Tans. on Computers*, 35(8), 677-691, 1986.

[3] Bryant, R. E., "Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams", *Computing Surveys*, 24(3), 298-318, 1992.

[4] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hawng, Symbolic Model Checking: 10^{20} states and Beyond. Information and Computation, 98(2): 142-170, June 1992.

[5] J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan and D. L. Dill, Symbolic Model Checking for Sequential Circuit Verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13, 401- 424, 1994.

[6] E. M. Clarke, A. Emerson and A. P. Sistla, Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8 (2), 244-263, 1986.

[7] E. M. Clarke, T. Filkorn, and S. Jha, Exploiting Symmetry in Temporal Logic Model Checking. *Proceedings of the Fifth Workshop on Computer-Aided Verification*, Ed. C. Courcoubetis. June/July 1993.

[8] E. M. Clarke, O. Grumberg, K. L. McMillan, and X. Zhao, Effective Generation of Counterexamples and Witnesses in Symbolic Model Checking. Technical Report No. CMU-CS-94-204, Carnegie Mellon University, PA, 1994.

[9] V. Hartonas-Garmhausen, T. Kurfess, E. M. Clarke, and D. E. Long, Automatic Verification of Industrial Designs. Proceedings of the 1995 IEEE Workshop on Industrial -Strength Formal Specification Techniques. 88-96. IEEE Comput. Soc. Press, April 1995.

[10] M. Jackson, *Software Requirements and Specifications*, ACM and Addison-Wesley, New York, 1995.

[11] K. L. McMillan, *Symbolic Model Checking - An Approach to the State Explosion Problem*, Ph.D. Thesis, Carnegie Mellon University, 1992.

[12] I. Moon, *Automatic Verification of Discrete Chemical Process Control Systems*, Ph.D. Thesis, Carnegie Mellon University, 1992.

[13] S. T. Probst, G. J. Powers, D. E. Long, and I. Moon, Verification of a Logically Controlled Solids Transport System using Symbolic Model Checking. Submitted for publication in *Computers and Chemical Engineering*, 1994.

[14] S. T. Probst, and G. J. Powers, Automatic Verification of Control Logic in the Presence of Process Faults. Presented at the Annual AIChE Conference, San Francisco, CA, November 1994.

[15] S. T. Probst, A. L. Turk, and G. J. Powers, Formal Verification of a Furnace System Standard. Presented at the Annual AIChE Conference, Miami Beach, FL, November 1995.

[16] S. T. Probst, *Chemical Process Safety and Operability Analysis using Symbolic Model Checking*, Ph.D. Thesis, Carnegie Mellon University, 1996.

[17] T. Sreemani and J. Atlee, Feasibility of Model Checking Software Requirements: A Case Study. Technical Report CS96-05, Department of Computer Science, University of Waterloo, January, 1996.