

**QUALITATIVE MODELING OF CONTINUOUS CHEMICAL PROCESSES  
AND APPLICATIONS TO FAULT DIAGNOSIS**

by

**OLAYIWOLA OLUWEMIMO OYELEYE**

Bachelor of Science, Chemical Engineering  
University of Lagos, Nigeria  
1982

Master of Science, Chemical Engineering Practice  
Massachusetts Institute of Technology  
1985

Submitted in partial fulfillment of the requirements  
for the degree of

**DOCTOR OF SCIENCE**

at the

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

February 1990

© Massachusetts Institute of Technology 1989

Signature of Author \_\_\_\_\_

Department of Chemical Engineering  
October 16, 1989

Certified by \_\_\_\_\_

Dr. Mark A. Kramer  
Associate Professor, Chemical Engineering  
Thesis Supervisor

Accepted by \_\_\_\_\_

Dr. William M. Deen  
Chairman, Department Graduate Committee

v.1  
MASSACHUSETTS INSTITUTE  
OF TECHNOLOGY

MAR 28 1989

1

LIBRARIES

QUALITATIVE MODELING OF CONTINUOUS CHEMICAL PROCESSES  
AND APPLICATIONS TO FAULT DIAGNOSIS

by

OLAYIWOLA OLUWEMIMO OYELEYE

Submitted to the Department of Chemical Engineering on October 16, 1989  
in partial fulfillment of the requirements for the degree of  
Doctor of Science in Chemical Engineering.

ABSTRACT

Incorporating knowledge about process behavior in a diagnostic model is one of the most crucial steps in the development of an automated aid for process malfunction diagnosis. In continuous chemical processes, this is further complicated by the complex dynamic behaviors exhibited by these systems, including inverse and compensatory (control system) responses. Several approaches may be employed in modeling process behavior, among which are those based on quantitative models, expert system rules and "formal" qualitative models. Although qualitative models have several advantages over quantitative models and expert system rules, the inherent ambiguity of qualitative modeling has in the past degraded the performance and limited the potential application of diagnostic aids based on these models. In order to develop diagnostic aids with improved performance, the ambiguity in qualitative modeling must be addressed.

In this thesis, a study of qualitative modeling was carried out. An algorithmic procedure for developing process diagnostic models from flowsheet information for a diagnostic aid for continuous chemical and refinery processes, called MIDAS was implemented. The approach taken was to develop models that generate predictions that coincide with observable process behaviors; predicting all actual behaviors while minimizing the prediction of spurious behaviors. In diagnostic applications, this ensures that accurate diagnoses are provided without unnecessary losses in diagnostic resolution.

An investigation of steady state qualitative modeling was carried out. Steady state qualitative models can be formulated in terms of constraints derived from fundamental principles governing the process. A systematic procedure was developed for reducing the ambiguity when predicting the ultimate qualitative state of continuous processes in response to process perturbations. The ambiguity was primarily reduced by applying additional qualitative constraints associated with analytically redundant quantitative equations, some of which can only be derived from global process considerations. In an example process, the number of qualitative solutions obtained by utilizing redundant constraint information was reduced by up to two orders of magnitude. Although steady state qualitative models are not suitable for dynamic processes, the need for global constraints reveals the need for global information when modeling qualitative dynamic process behavior.

The modeling of dynamic process behavior using Signed Directed Graph (SDG) based representations of causality was investigated. In order to reduce ambiguity, previous workers have in the past limited disturbance propagation to feedforward paths in the SDG. Others adopted heuristics that make exceptions to this assumption, thus, explaining compensatory response exhibited by controlled variables in feedback control loops. A rigorous analysis was performed and indicated that the heuristics are inadequate, leading to the exclusion of inverse and compensatory response arising from interactions in non-control negative feedback loops in the SDG. Specifically, the connection between these

behaviors and the **global** topography of the SDG was established: inverse and compensatory response arise from **positive feedback** effects associated with negative feedback loops and the effect of **integrators** in negative feedback loops, respectively. Process examples in which previous heuristics have proven to be inadequate include: (i) inverse response of reactant concentration to changes in feed concentration in exothermic reactions in continuous stirred tank reactors, (ii) compensatory response of outlet flow to partial blockages in the outlet of gravity flow tanks and (iii) compensatory response of manipulated variables in control loops when external disturbances enter the loop at the manipulated variable.

A novel representation of causality for dynamic continuous processes, called the Extended Signed Directed Graph (ESDG) was developed. The ESDG represents both local causality (represented by the SDG) and "apparent" global causality (observed as inverse or compensatory response arising from negative feedback effects) in the process. The ESDG is derived from the SDG, by adding branches to account for the behavior of variables that exhibit inverse and compensatory response to negative feedback. Guidelines for deriving SDG models from algebraic and differential equations were developed. By limiting disturbance propagation to feedforward paths, the ESDG predicts all initial and ultimate behaviors while minimizing the production of spurious behaviors. The ESDG is suggested as the basis for dynamic process diagnostic models.

An algorithmic procedure for deriving the qualitative core of the diagnostic model for MIDAS was developed. Using the procedure, diagnostic process models can be systematically produced with more reliability and consistency than rule sets in conventional rule based expert systems. The effort in developing the process model is largely reduced to specifying units from a library of SDG unit models according to the process flowsheet structure. The additional expense of parameter estimation when developing models for quantitative approaches is avoided.

A novel feature of the procedure is that behaviors not apparent from local SDG models but that can be explained by additional branches of the ESDG arcs are automatically represented in the diagnostic process model. Representing knowledge about these behaviors allows an accurate diagnosis to be obtained in a dynamic process environment without unnecessary losses in diagnostic resolution. Varying amounts of process-specific information can be incorporated in the process model using the procedure, leading to improvements in diagnostic resolution. The process model explicitly relates measured variable states to malfunctions, allowing a diagnosis to be made using only existing process instrumentation.

In a case study of 77 malfunctions in a simulated dynamic chemical process, MIDAS produced an accurate initial diagnosis in 93% of the cases, an accurate final diagnosis in 100% of the cases and on average 96% of the malfunctions were eliminated by the end of the transient. The acceptable level of performance achieved by MIDAS validated the procedure for deriving diagnostic process models for dynamic continuous processes.

Thesis Supervisor: Dr. Mark A. Kramer  
Associate Professor of Chemical Engineering

## ACKNOWLEDGEMENT

I would like to express my thanks to several people, who in their own special ways have contributed to the completion of this thesis.

My thesis advisor, Dr. Mark Kramer for his untiring enthusiasm, providing valuable insights and his overall guidance and direction. I benefitted greatly from our many discussions and frequent interactions.

My thesis committee, Dr. Lawrence Evans, Dr. George Stephanopoulos and Dr. Jefferson Tester, for their incisive comments, suggestions and constructive criticism. Their wealth of experience enhanced the quality of this thesis.

My friend and colleague, Eric Finch, with whom MIDAS was developed. Without his valuable contributions, this thesis may have been of less significance. Our interaction was definitely a continuous source of stimulation.

Other members of my research group (James Leonard, Philippe Rose and Bernard Palowitch Jr.), for all their useful suggestions which proved quite valuable in advancing this thesis.

My sponsors the Nigerian Government, and the National Science Foundation which provided partial funding for this project (under grant CBT-8605253). I am grateful for their financial support.

All my friends in Cambridge, both past and present, who have made life here all the more enjoyable.

Finally, I am extremely grateful to all members of my family, for their constant words of encouragement and unwavering support. The sacrifices they have all endured on my behalf, without which this thesis, as have all my previous undertakings, would not be possible. Especially to my parents for all their love, caring and understanding. I don't believe I can find words expressive enough to thank them. I dedicate this thesis to them, as a small token of my appreciation.

*To my parents,  
for all their love, caring and understanding.*

## TABLE OF CONTENTS

Title Page .....	1
Abstract .....	2
Acknowledgement .....	4
Table of Contents .....	6
List of Figures .....	15
List of Tables .....	19
<b>Thesis Summary .....</b>	<b>22</b>
S.1. Introduction .....	22
S.1.1. Objectives .....	26
S.1.2. Summary of Results .....	27
S.2. Steady State Qualitative Modeling .....	29
S.2.1. Simulation of Steady States by Confluences .....	29
S.2.2. Resolving Ambiguity with Latent Confluences .....	30
S.2.3. Qualitative Modeling of a Chemical Reactor Process .....	33
S.3. Causal Modeling of Continuous Processes .....	37
S.3.1. Representation of Causality by Signed Directed Graphs .....	37
S.3.2. A Novel Representation of Causality: The ESDG .....	40
S.4. Process Modeling in MIDAS .....	46
S.4.1. The MIDAS Process Model .....	46
S.4.2. Developing the Process Model .....	51
S.4.3. Process Model Verification: MIDAS Case Study .....	54
S.4.3.1. Process Model of a Jacketed Reactor Process .....	54
S.4.3.2. Case Study .....	58
S.4.3.3. Case Study Results and Analysis .....	59
S.4.4. Limitations .....	63
S.5. Conclusions and Recommendations .....	67
S.6. References for Thesis Summary .....	68
<b>Chapter 1 Introduction: Qualitative Modeling and Applications to Malfunction Diagnosis .....</b>	<b>72</b>
1.1. Motivation .....	72

1.2.	Process Modeling .....	73
1.3.	Thesis Objectives .....	77
1.4.	Thesis Outline .....	78
1.5.	References for Chapter 1 .....	80
<b>Chapter 2 Steady State Qualitative Modeling .....</b>		<b>86</b>
2.1.	Review of Previous Work on Qualitative Constraint Modeling .....	86
2.2.	Simulation of Steady States by "Pure" Confluences .....	88
2.2.1.	Local Steady State Constraints .....	88
2.2.2.	Constraint Satisfaction Algorithm .....	92
2.2.3.	Resolving Ambiguity with Latent Constraints .....	95
2.2.4.	Tradeoff between Modularity and Multiplicity .....	98
2.2.5.	Example of Simulation using Confluences .....	99
2.3.	Use of Causality in Steady State Qualitative Modeling .....	102
2.3.1.	Representation of Causality by the ESDG .....	103
2.3.2.	Derivation of Confluences from the ESDG .....	104
2.3.2.1.	Nodal Balances for ESDGs Without Feedback Loops .....	104
2.3.2.2.	Systems with Negative Feedback Loops .....	106
2.3.2.3.	Global Confluences for Positive Feedback Loops .....	107
2.3.2.4.	Using Global Topology to Derive Causal Confluences .....	110
2.3.3.	Reducing Ambiguity with Causal Confluences .....	110
2.4.	Qualitative Fault Simulation of a Chemical Reactor System .....	114
2.4.1.	Using Knowledge of Control Loop Function to Reduce Ambiguity .....	121
2.5.	Discussion .....	123
2.6.	Notation for Chapter 2 .....	126
2.7.	References for Chapter 2 .....	127
<b>Chapter 3 Causal Modeling of Continuous Processes .....</b>		<b>130</b>
3.1.	Review of Previous Work on Causal Modeling .....	130
3.2.	The Single-Stage Signed Directed Graph .....	133
3.2.1.	Derivation of the SDG .....	133
3.2.2.	Determining System Behavior from the SDG .....	135
3.2.3.	Limitation of Iri's Propagation Assumptions .....	138
3.3.	Determining System Behavior by Global Qualitative Analysis .....	140

3.3.1. Initial System Behavior .....	141
3.3.2. Ultimate System Behavior .....	143
3.4. The Extended Signed Directed Graph .....	158
3.4.1. ESDG Arcs for IR .....	158
3.4.2. ESDG Arcs for CR .....	159
3.5. Transitions Across Qualitative Regimes .....	160
3.5.1. Modifying the ESDG to Account for Process Behavior During Transitions .....	161
3.5.1.1. Modification for Type A Transitions .....	162
3.6. Using the ESDG as a Dynamic Process Model .....	167
3.7. Conclusion .....	170
3.8. Notation for Chapter 3 .....	171
3.9. References for Chapter 3 .....	172
<b>Chapter 4 Guidelines for Developing SDG Models .....</b>	<b>176</b>
4.1. Introduction .....	176
4.2. Representation of the SDG .....	177
4.2.1. Variables .....	178
4.2.2. Causal Arcs .....	179
4.2.3. Root Causes .....	180
4.3. Developing SDG Models for Process Units .....	180
4.3.1. Variable Selection .....	181
4.3.2. Selecting the Malfunctions .....	183
4.3.3. Determining the Causal Structure .....	184
4.3.3.1. Independent Variables and Parameters .....	184
4.3.3.2. Ordinary Differential Equations .....	185
4.3.3.3. Partial Differential Equations .....	188
4.3.3.4. Algebraic Equations .....	188
4.3.3.5. Summary of Arc Specification Procedures .....	193
4.4. Developing SDG Models for Control System Units .....	193
4.5. Examples of SDG Models of Process and Control System Units .....	195
4.5.1. Pressure-flow Phenomena in a Pipe .....	195
4.5.2. Mixing Tank .....	198
4.5.3. Continuous Stirred Tank Reactor .....	201
4.5.4. Shell and Tube Heat Exchanger .....	203

4.5.5. Control Valve .....	207
4.5.6. PI Controller .....	209
4.6. Conclusion .....	210
4.7. Notation for Chapter 4 .....	211
4.8. References for Chapter 4 .....	212
<b>Chapter 5 The Model Integrated Diagnostic Analysis System: An Overview .....</b>	<b>213</b>
5.1. Introduction .....	213
5.2. Knowledge Representation in MIDAS .....	215
5.2.1. Programming Methodology .....	215
5.2.2. Monitors .....	216
5.2.3. Process Model .....	218
5.2.4. Hypothesis Model .....	219
5.3. Inference in MIDAS: The Event Interpreter .....	220
5.4. Implementation and Program Architecture .....	222
5.5. Conclusion .....	224
5.6. References for Chapter 5 .....	224
<b>Chapter 6 Process Modeling in MIDAS .....</b>	<b>226</b>
6.1. Introduction .....	226
6.2. Deriving Process Models for MIDAS: An Overview .....	228
6.2.1. Desirable Attributes of the Modeling Approach .....	230
6.3. The MIDAS Process Model .....	235
6.3.1. Representing Dynamic Process Behavior using the PM .....	235
6.3.2. Description of Process Model Objects .....	240
6.4. Creating the Process SDG .....	245
6.4.1. Description of Process SDG Objects .....	246
6.4.2. Algorithm for Creating the SDG .....	247
6.5. Creating the Process ESDG .....	251
6.5.1. Theoretical Basis for Topological Analysis of the SDG .....	251
6.5.1.1. Criteria for Locating ESDG Arcs .....	252
6.5.2. Additional Attributes of ESDG Objects .....	256
6.5.3. Alternative Criteria for Locating IVs and CVs .....	256
6.5.4. Algorithm for Constructing the ESDG .....	258

6.5.5. Complexity of the Algorithm for Constructing the ESDG .....	263
6.5.5.1. Reducing the Time Required to Create ESDG .....	266
6.6. Deriving Qualitative Relationships in the PM .....	273
6.6.1. Basic Tasks Performed by the Algorithm .....	273
6.6.2. Algorithm for Creating the PM .....	275
6.6.3. Features of the PM of a Heater Process .....	283
6.6.3.1. Modifying Temporal Ordering to Improve Robustness to Out of Order Events .....	
6.6.3.1.1. Modifying Temporal Ordering to Reflect Causality .....	291
6.6.4. Complexity of the Algorithm for Deriving the PM .....	292
6.7. Specifying Quantitative Constraints and Tests in the PM .....	293
6.7.1. Representing Constraints in the PM .....	294
6.7.2. Guidelines for Specifying Constraint Information .....	297
6.7.3. Specifying Test Results in the PM .....	298
6.8. Improvements to Procedures for Deriving the Qualitative Core of the PM .....	299
6.8.1. Transitions Across Qualitative Regimes .....	299
6.8.2. Improving the Algorithm for Deriving the ESDG .....	300
6.8.2.1. Devising a More Efficient Algorithm .....	301
6.8.2.2. Decomposing Large SCCs .....	301
6.8.3. Improving the Algorithm for Deriving the PM .....	302
6.8.3.1. Reducing the Connectivity of the PM .....	303
6.8.3.2. Devising a More Efficient Algorithm .....	304
6.9. Conclusion .....	305
6.10. Notation and Abbreviations for Chapter 6 .....	305
6.10.1. List of Abbreviations .....	305
6.10.2. Notation .....	306
6.11. References for Chapter 6 .....	308
<b>Chapter 7 MIDAS Case Study .....</b>	<b>310</b>
7.1. Introduction .....	310
7.2. Jacketed Reactor Process .....	312
7.2.1. Process Description .....	312
7.2.2. Developing the Process Model .....	314
7.2.2.1. Developing the Qualitative Core of the Model .....	315
7.2.2.2. Adding Constraints to the Process Model ....	323

7.2.2.3. The Process Model .....	328
7.3. Case Study .....	330
7.3.1. Process Simulation .....	330
7.3.2. Diagnosis of Simulated Malfunctions .....	331
7.4. Case Study Results and Analysis .....	332
7.4.1. Distribution of Events .....	333
7.4.2. Non Monotonic Dynamic Behavior .....	333
7.4.3. Diagnostic Performance .....	334
7.5. Limitations of Qualitative Modeling .....	340
7.6. Conclusions .....	342
7.7. Notation for Chapter 7 .....	343
7.8. References for Chapter 7 .....	343
<b>Chapter 8 Conclusions and Recommendations .....</b>	<b>345</b>
8.1. Steady State Qualitative Modeling .....	346
8.2. A Novel Representation of Causality for Continuous Process .....	347
8.3. Developing Process Diagnostic Models for MIDAS .....	349
8.4. Recommendations .....	350
<b>Chapter 9 MIDAS Model Builder .....</b>	<b>352</b>
9.1. Loading Builder Utility Files .....	353
9.2. Running a Model Building Session .....	354
9.2.1. Starting with a Partially Complete Process Model .....	355
9.2.2. Resume Model Building .....	358
9.2.2.1. Adding Units to the Process Model .....	358
9.2.2.2. Adding Sensors to the Process Model .....	360
9.2.2.3. Connecting Units and Sensors in the Flowsheet .....	361
9.2.2.4. Ending the Session .....	361
9.3. Using the Model Library to Create SDG Object Instances .....	362
9.3.1. Questions Common to all Units .....	364
9.3.2. Questions Specific to a Particular Type of Unit .....	367
9.3.2.1. Controller .....	369
9.3.2.2. Centrifugal Pump .....	371
9.3.2.3. Continuous Stirred Tank Reactor .....	373

9.3.2.4.	Control Valve .....	378
9.3.2.5.	Effluent Stream .....	380
9.3.2.6.	Feed Stream .....	381
9.3.2.7.	Continuous Stirred Tank Reactor with Jacket .....	382
9.3.2.8.	Junction .....	389
9.3.2.9.	Manual Valve .....	392
9.3.2.10.	Pipe .....	394
9.3.2.11.	Sensor .....	395
9.3.2.12.	Shell and Tube Heat Exchanger .....	397
9.3.2.13.	Tank .....	400
9.3.3.	Making Changes to the Model Library .....	404
9.3.3.1.	Increasing the Number of Chemical Species, Reactions and Catalysts .....	404
9.3.3.2.	Modifying Programs Creating Unit Models .....	405
9.3.3.3.	Adding New Unit Models to the Library .....	405
9.4.	Creating Unit SDG Objects Using Batch Files .....	406
9.4.1.	Batch File Input Language .....	407
9.4.2.	Batch Input Files for a Gravity Flow Mixing Tank Process .....	412
9.4.2.1.	Input File for Pipe-A .....	417
9.4.2.2.	Input File for Tank .....	418
9.4.2.3.	Input File for L-Sensor .....	420
9.5.	Building the Process SDG from Batch Files and the Model Library .....	421
9.6.	Naming Convention for SDG Objects .....	423
<b>Chapter 10</b>	<b>Midas Model Translator .....</b>	<b>424</b>
10.1.	Loading Top Level Translator Program .....	425
10.2.	Running a Model Translation Session .....	426
10.2.1.	Loading Frame Definitions and Demons .....	427
10.2.2.	Starting with a Partially Translated Model .....	427
10.2.3.	Deriving the ESDG from the SDG .....	430
10.2.4.	Translating the ESDG to the PM .....	434
10.2.5.	Specifying Measured Constraints and Tests in the PM .....	438
10.2.6.	Creating Screen Interface Objects .....	439
10.2.7.	Saving the PM .....	441
10.3.	Naming Convention for PM and Screen Interface Objects .....	442

10.3.1. PM Objects .....	442
10.3.2. Screen Interface Objects .....	444
10.4. Specifying Measured Constraints and Tests Using Batch Files .....	444
10.4.1. Batch File Input Language .....	444
10.4.2. Batch Input File for PM of a Hypothetical Process .....	448
10.4.2.1. Input File For Process PM .....	449
<b>Appendix A Steady State Equations for Heat Exchanger with Bypass .....</b>	<b>451</b>
<b>Appendix B Non-Causal Confluences for Continuous Stirred Tank Reactor with Recycle .....</b>	<b>453</b>
<b>Appendix C Confluences Derived from ESDG for Continuous Stirred Tank Reactor with Recycle .....</b>	<b>457</b>
<b>Appendix D Proof that Interphase Transport and Reversible Reactions cannot Result in Inverse Variables .....</b>	<b>460</b>
<b>Appendix E Attributes of Monitor and Screen Interface Objects .....</b>	<b>462</b>
<b>Appendix F Attributes of (E)SDG Objects .....</b>	<b>464</b>
<b>Appendix G Frame Objects for Pipe Unit Model .....</b>	<b>468</b>
G.1. Unit Object .....	468
G.2. Root Cause Objects .....	468
G.3. Variable Objects .....	469
G.4. Permanent Causal Arc Objects .....	472
G.5. Temporary Causal Arc Objects .....	475
<b>Appendix H Algorithms for Deriving the ESDG .....</b>	<b>476</b>
H.1. Pidgin Algol .....	476
H.2. Bipartite Matching Algorithm .....	478
H.3. Algorithm for Finding Cycles in the SDG .....	479
H.4. References for Appendix H .....	482

<b>Appendix I</b>	Evaluating Qualitative Determinants of Structures with Flow-Pressure Cycles .....	483
I.1.	Linear Flow-Pressure Cycles .....	483
I.2.	Recycle Flow-Pressure Cycles .....	483
I.3.	Bypass Flow-Pressure Cycles .....	484
<b>Appendix J</b>	Algorithms for Deriving the PM .....	485
J.1.	Algorithm for Finding Primary Deviation Paths of Root Causes .....	485
J.2.	Algorithm for finding Successor Paths from a Sensed Node .....	488
J.3.	Possible Transitions of Compiled Causal Links .....	494
<b>Appendix K</b>	Portion of the PM for Jacketed Reactor Process .....	497

## LIST OF FIGURES

### Thesis Summary

Fig. S.1.	Continuous Stirred Tank Reactor with Recycle .....	34
Fig. S.2.	Relationship Between Solutions to Different Sets of Confluences .....	36
Fig. S.3a.	Gravity Flow Tank .....	38
Fig. S.3b.	SDG of Gravity Flow Tank .....	38
Fig. S.4.	Response of Gravity Flow Tank to Outlet Blockage .....	43
Fig. S.5a.	Continuous Stirred Tank Reactor (Temperature and Level Control Loops not Shown) .....	44
Fig. S.5b.	Response to High Feed Concentration .....	44
Fig. S.6a.	Process Feed Preheater .....	45
Fig. S.6b.	Response of Process Effluent Flow through Preheater Shell to Partial Blockage in Pipe-5 .....	45
Fig. S.7.	Basic Structure of MIDAS .....	47
Fig. S.8.	Gravity Flow Tank Process .....	48
Fig. S.9.	PM for Gravity Flow Tank Process .....	49
Fig. S.10.	Steps in Constructing the PM .....	52
Fig. S.11.	Jacketed Continuous Stirred Tank Reactor Process .....	55
Fig. S.12a.	Distribution of Local Causes .....	57
Fig. S.12b.	Distribution of Compiled Causal Links .....	57
Fig. S.13.	Resolution at End of Transient .....	62
Fig. S.14.	Apparent Inverse Response .....	64
Fig. S.15.	PM of a Hypothetical Process .....	66

### Chapter 2

Fig. 2.1.	Flowchart of Constraint Satisfaction Algorithm .....	93
Fig. 2.2.	Countercurrent Heat Exchanger with Bypass .....	99
Fig. 2.3.	ESDG of Hypothetical System without Feedback Loops .....	105
Fig. 2.4.	ESDG of System with Negative Feedback Loop .....	107
Fig. 2.5.	ESDG of System with Positive Feedback Loop .....	108
Fig. 2.6.	ESDG of Countercurrent Heat Exchanger .....	112
Fig. 2.7.	ESDG for Autocatalytic Reaction .....	114
Fig. 2.8.	Continuous Stirred Tank Reactor with Recycle .....	116

Fig. 2.9.	ESDG for CSTR with Recycle .....	118
Fig. 2.10.	Relationship Between Solutions to Different Sets of Confluences .....	121

### **Chapter 3**

Fig. 3.1a.	Gravity Flow Tank .....	139
Fig. 3.1b.	SDG of Gravity Flow Tank .....	139
Fig. 3.1c.	Response to Outlet Blockage .....	139
Fig. 3.1d.	ESDG of Gravity Flow Tank .....	139
Fig. 3.2a.	SDG of CSTR .....	152
Fig. 3.2b.	Response to High Feed Concentration .....	152
Fig. 3.2c.	ESDG of CSTR .....	152
Fig. 3.3a.	SDG of Fluidized Bed Reactor .....	155
Fig. 3.3b.	ESDG of Fluidized Bed Reactor .....	155
Fig. 3.4a.	SDG of Control Loop .....	157
Fig. 3.4b.	ESDG of Control Loop .....	157
Fig. 3.5a.	SDG of CSTR with Control System .....	165
Fig. 3.5b.	ESDG of CSTR with Control System (Nominal Qualitative Regime) ....	165
Fig. 3.5c.	Response to High Feed Concentration (Saturated Control System) .....	166
Fig. 3.5d.	ESDG of CSTR with Control System (Saturated Control System) .....	166
Fig. 3.6a.	Apparent CR (overdamped response) .....	169
Fig. 3.6b.	Apparent CR (underdamped response) .....	169
Fig. 3.6c.	Apparent IR .....	169
Fig. 3.6d.	Apparent IR (CR) .....	169

### **Chapter 4**

Fig. 4.1a.	Model of Flow-Pressure Phenomena in Pipes .....	197
Fig. 4.1b.	SDG of Pipe .....	197
Fig. 4.2a.	Mixing Tank .....	198
Fig. 4.2b.	SDG of Mixing Tank .....	200
Fig. 4.3a.	Continuous Stirred Tank Reactor .....	201
Fig. 4.3b.	SDG of CSTR .....	204
Fig. 4.4a.	Shell and Tube Heat Exchanger .....	205
Fig. 4.4b.	SDG of Shell and Tube Heat Exchanger .....,	206
Fig. 4.5.	SDG of Control Valve .....	208

Fig. 4.6.	SDG of Controller .....	210
-----------	-------------------------	-----

## **Chapter 5**

Fig. 5.1.	Relationship of MIDAS Objects to Event Interpreter .....	217
Fig. 5.2.	Flowsheet of MIDAS Inference Cycle .....	221
Fig. 5.3.	Structure of MIDAS Software Package .....	223

## **Chapter 6**

Fig. 6.1.	Steps in Constructing the PM .....	229
Fig. 6.2.	Gravity Flow Tank .....	236
Fig. 6.3a.	State Transition Graph for Gravity Flow Tank .....	238
Fig. 6.3b.	PM for Gravity Flow Tank .....	239
Fig. 6.4.	Flowchart of Process SDG Creation Algorithm .....	248
Fig. 6.5.	Unit Model Creation Routines .....	250
Fig. 6.6.	Flowchart of Algorithm for Constructing the ESDG .....	260
Fig. 6.7.	Continuous Stirred Tank Reactor with Recycle .....	265
Fig. 6.8a.	Linear Path in SDG .....	268
Fig. 6.8b.	Linear Path After Condensation .....	268
Fig. 6.9a.	Flow-Pressure Cycles in Linear Configuration .....	269
Fig. 6.9b.	Linear Flow-Pressure Cycles After Condensation .....	269
Fig. 6.10a.	Flow-Pressure Cycles in Recycle Configuration .....	270
Fig. 6.10b.	Recycle Flow-Pressure Cycles After Condensation .....	270
Fig. 6.11a.	Flow-Pressure Cycles in Bypass Configuration .....	271
Fig. 6.11b.	Bypass Flow-Pressure Cycles After Condensation .....	271
Fig. 6.12.	Flowchart of Algorithm for Creating PM .....	279
Fig. 6.13.	Relationship Between Sensed Variables and Sensor Signals .....	282
Fig. 6.14.	Process Feed Preheater .....	284
Fig. 6.15a.	Distribution of Local Causes .....	285
Fig. 6.15b.	Distribution of Compiled Causal Links .....	285
Fig. 6.16.	Portion of PM of Feed Preheater .....	286
Fig. 6.17a.	ESDG of Hypothetical Process .....	289
Fig. 6.17b.	PM of Hypothetical Process .....	289
Fig. 6.17c.	PM of Hypothetical Process (robust to out of order sequences) .....	289
Fig. 6.18.	Representing Dynamic Behavior of Constraint Residuals .....	296

## **Chapter 7**

Fig. 7.1.	Jacketed Continuous Stirred Tank Reactor Process .....	313
Fig. 7.2a.	Distribution of Local Causes .....	329
Fig. 7.2b.	Distribution of Compiled Causal Links .....	329
Fig. 7.3.	Distribution of Events .....	333
Fig. 7.4.	Tier of True Malfunction .....	336
Fig. 7.5.	Average Resolution .....	337
Fig. 7.6.	Average Performance .....	338
Fig. 7.7.	Resolution at End of Transient .....	339
Fig. 7.8.	PM of a Hypothetical Process .....	340

## **Chapter 9**

Fig. 9.1.	Gravity Flow Mixing Tank .....	412
Fig. 9.2.	SDG for PIPE-A .....	413
Fig. 9.3.	SDG for TANK .....	414
Fig. 9.4.	SDG for L-SENSOR .....	415
Fig. 9.5.	Legend For SDG Models .....	416

## **Chapter 10**

Fig.10.1a.	PM Showing Measured Constraints .....	448
Fig.10.1b.	PM Showing Tests .....	449

## LIST OF TABLES

### **Thesis Summary**

Table S.1	Number of Solutions Obtained Using Confluences from Various Sources .....	35
Table S.2	Average Diagnostic Performance of MIDAS .....	60

### **Chapter 2**

Table 2.1	Table of Qualitative Operations .....	90
Table 2.2	Solutions to Confluences of Heat Exchanger with Bypass .....	101
Table 2.3	Solutions to ESDG without Feedback Loops (Fig. 2.3) .....	105
Table 2.4	Solutions to Nodal Balances of Eq. (2.15) .....	109
Table 2.5	Solutions to Non-Causal Confluences for Autocatalytic Reaction .....	113
Table 2.6	ESDG Arcs for Control Loops of CSTR with Recycle .....	117
Table 2.7	Selected Faults for CSTR with Recycle .....	119
Table 2.8	Number of Solutions Obtained Using Confluences from Various Sources .....	120
Table 2.9	Results of Qualitative Simulation (Perfect Control) .....	122
Table 2.10	Information Sources for Steady State Qualitative Modeling .....	124

### **Chapter 3**

Table 3.1	Parameters for Continuous Stirred Tank Reactor in Example 1 .....	153
-----------	---	-----

### **Chapter 6**

Table 6.1	Potential Root Cause Object .....	242
Table 6.2	Potential Event Object .....	242
Table 6.3	Measured Variable Object .....	243
Table 6.4	Measured Constraint Object .....	243
Table 6.5	Compiled Causal Link Object .....	244
Table 6.6	Test Object .....	244
Table 6.7	Time Taken by Algorithm to Create ESDG of Reactor Process .....	266
Table 6.8	Time Taken to Create ESDG of Reactor Process After Condensation ....	272

## **Chapter 7**

Table 7.1	Unit Failure Modes for Jacketed CSTR Process .....	316
Table 7.2	Resolution of Ambiguity During PM Construction .....	320
Table 7.3	Relationships Between Root Causes and Constraints .....	325
Table 7.4	Average Diagnostic Performance of MIDAS .....	335

## **Chapter 9**

Table 9.1	Builder Utility Files .....	352
Table 9.2	Model Library Files .....	353
Table 9.3	Unit Library Connectivity Matrix .....	368
Table 9.4	Units in Gravity Mixing Tank Process .....	413
Table 9.5	Model Library Character Codes for Boundary Variables .....	422

## **Chapter 10**

Table 10.1	Translator Utility Files .....	424
------------	--------------------------------	-----

## **Appendix E**

Table E.1	Sensor Monitor Object .....	462
Table E.2	Constraint Monitor Object .....	462
Table E.3	Screen Interface Popup-choose Object .....	463

## **Appendix F**

Table F.1	Units Object .....	464
Table F.2	Root Causes Object .....	465
Table F.3	Variables Object .....	465
Table F.4	Causal Arcs Object .....	466
Table F.5	Strongly Connected Component Object .....	467

## **Appendix J**

Table J.1	Possible Transitions Associated with Causal Links (Source Node and Result Node Different) .....	495
Table J.1	Possible Transitions Associated with Causal Links (Source Node and Result Node Identical) .....	496

## S. Thesis Summary

### S.1. Introduction

In most continuous chemical process plants, supervisory control activities include the implementation of start-up, shut-down and change-over procedures, process monitoring and malfunction diagnosis. In the event of a process disturbance or malfunction, the operator's task is to diagnose the cause of the upset quickly and accurately so that corrective action may be taken. In modern integrated plants, strong dynamic interactions between process units may further complicate diagnosis because:

- Effects may propagate to process units located at great distances from the malfunction origin.
- Regulatory control and backup systems may obscure the symptoms of the malfunction.

Timely intervention by the operator is very important, because it may:

- Mitigate more serious incidents such as the release of toxic chemicals,
- Improve product quality, and
- Improve plant profitability by avoiding material and energy wastage due to spurious shutdowns.

In order to perform an accurate diagnosis, the operator must understand the behavior of the process. The operator's mental model of process behavior is usually developed by experience and not rigorous, and consequently the model may be inaccurate or limited to a "normal" regime of plant operation. In time-constrained situations, mental simulation of hypothetical situations can place significant cognitive load on the operator, reducing his or her effectiveness. Consequently, there are motivations to develop computer aids to assist the operator in diagnosis.

Modeling and representing process behavior is a central issue in developing diagnostic aids. Interpreting process data requires a **process model** that satisfies the following requirements:

- Contains enough information to produce the correct diagnosis.
- Can be developed with reasonable engineering effort.
- Can be easily verified.

Several types of modeling approaches may be employed in developing diagnostic aids. These include the following:

*Quantitative Approaches:*

Diagnostic aids based on dynamic quantitative models such as parameter estimation and filtering techniques [1, 2, 3 & 4], hypothesis testing of residual vectors on a bank of models [5 & 6], and comparative simulation, utilize a mathematical and statistical framework that allows considerable theoretical sophistication. Excellent reviews of diagnostic methods based on quantitative models are presented in [7 & 8]. These methods potentially produce the most accurate diagnoses. In addition, the models are explicit and well defined, therefore, directly accessible for verification. However, on the practical side, there are questions as to whether these techniques can be successfully applied to large processing plants. The excessive computational requirements and the considerable effort required to develop highly reliable and accurate models limit the applicability of the techniques.

*Rule Based Expert System Approach:*

Rule based expert systems are symbolic reasoning programs developed in the field of Artificial Intelligence that attempt to model the diagnostic knowledge of experienced process engineers and operators, expressed as a set of situation-action production rules. The rules typically specify a heuristic mapping between patterns of abnormal symptoms such as high and low states and specific process malfunctions. Expert systems have been successfully applied in domains where there is a limited fundamental understanding. For example, in the medical domain, the MYCIN system was developed for the diagnosis of

infectious blood diseases [9]. In the chemical process industry, expert systems have been applied, achieving varying degrees of success. Notable examples are the FALCON project [10] and the ESCORT system [11].

Several factors limit the applicability of rule based systems [12, 13 & 14] to chemical process diagnosis, where the fundamental principles are well understood. The rules are derived from experience which could be totally or partially missing in a new plant. In addition, the rules suffer from a lack of generality and each process requires development of unique rule base which may consist of as many as 10,000 rules [15]. Most importantly, the procedure used in developing a set of rules for an application is unstructured and often ad hoc. This potentially results in rule sets that may be unreliable, incomplete and cannot be easily verified.

### *Qualitative Modeling Approaches:*

Other symbolic diagnostic programs are based on "formal" qualitative models of process behavior, derived from the fundamental principles governing the domain. In their most general form, these models can be viewed as abstractions of the conventional differential/algebraic equation models in which some or all quantitative information is omitted. Continuous process variables are represented in qualitative models by a limited set of discrete qualitative states, and the models represent the relationships between these states.

Qualitative models can be further classified as qualitative constraint models and qualitative causal models. The former include the modeling formalisms of de Kleer and Brown [16], Pan [17], Kuipers [18] and Williams [19]. All these formalisms formulate the model in terms of constraints relating qualitative states of process variables, differing primarily in the way the constraints are derived. Constraint models have been used for diagnostic and other applications in several domains.

An important class of qualitative causal models is based on the Signed Directed Graph (SDG) and its related diagrams [20, 21, 22 & 23]. These diagrams consist of nodes symbolizing process variables, and signed directed arcs that represent the local cause and effect relationships between variables. Other representations of causality used in reasoning about process behavior are those based on the formalisms of Iwasaki and Simon [24 & 25]

in which causality is construed as the output set assignment [26] of a set of simultaneous algebraic and differential equations and the Qualitative Process Theory of Forbus [27].

The major attraction of using formal qualitative models over other modeling formalisms for diagnostic applications is based on the following factors:

- They contain much of the necessary information useful for diagnosis and can be constructed with less effort than quantitative models.
- Most of the information required in developing qualitative models is general and not process-specific.
- The computational effort in the application phase is reduced.
- Reasoning with qualitative models is similar to that of humans and thus readily understandable and explainable. Causality is clearly a basic mechanism of human reasoning about process behavior [28].
- Relative to rule-based expert systems, qualitative modeling provides a more structured approach for developing diagnostic knowledge bases.

On the other hand, qualitative modeling is fundamentally underspecified, as there is an inherent ambiguity in qualitative modeling [29]. Previous methods of qualitative modeling have tended to generate predictions that do not coincide with observable behaviors, either by including spurious behaviors or excluding actual behaviors [30]. Previous work on SDG models has not adequately addressed the role of feedback and complex dynamics. Disturbance propagation is restricted to feedforward paths in the SDG and heuristics are applied to account for complex dynamic process behavior arising due to interactions in negative feedback loops [20]. The heuristics are incomplete as they do not account for all instances of dynamic behavior [30]. In diagnostic applications, these limitations result in unnecessary losses of diagnostic resolution and the production of inaccurate diagnosis. In addition, qualitative modeling does not utilize available process-specific numerical information which might prove useful in improving diagnostic resolution.

### **S.1.1. Objectives**

The above limitations are crucial and have in the past reduced the potential application of diagnostic systems based on qualitative models. Therefore, the objectives of this thesis are to:

- 1) Understand the limitations of qualitative modeling at a fundamental level. Investigate steady state qualitative modeling, identifying the elements required to reduce the inherent ambiguity.
- 2) Investigate the limitations of previous disturbance propagation assumptions on modeling process behavior arising from interactions in feedback loops of the SDG.
- 3) Develop a representation of process causality that produces all valid process behaviors while minimizing spurious behaviors.
- 4) Develop a structured approach and implement procedures and algorithms for automatically deriving diagnostic knowledge bases for continuous processes from their constituent units and flowsheet topographies. The procedures should produce knowledge bases that have the following properties:
  - a) Can be easily verified for consistency and completeness.
  - b) Provide accurate diagnoses in processes with strong dynamic interactions without unnecessary losses in diagnostic resolution.
  - c) Can provide a diagnosis using the existing process instrumentation scheme.
  - d) Can be developed with incomplete information about process-specific parameter values. Available information about parameter values could be used to refine the diagnosis.

## S.1.2. Summary of Results

This thesis has addressed the problem of qualitative modeling and its applications in the automated diagnosis of process malfunctions in continuous chemical processes. Although emphasis has been placed on the modeling of continuous chemical processes, the concepts presented in the thesis are applicable to continuous dynamic processes in other domains.

One of the recurring concepts developed in this thesis has been the systematic use of "quantitatively redundant" global information to reduce the ambiguity inherent in qualitative modeling. In steady state qualitative modeling, ambiguity was reduced by deriving additional latent constraints, some of which can only deduced from global process topology. In causal modeling, the ambiguity was reduced by restricting disturbance propagation to feedforward paths in local SDG models and specifying additional arcs based on the global topology of the SDG.

An algorithmic technique for deriving the qualitative core of the knowledge base for an on-line diagnostic reasoning program<sup>1</sup> for continuous refinery and chemical processes was developed. Using the technique, diagnostic knowledge bases<sup>2</sup> can be systematically produced with more reliability and consistency than rule sets in conventional rule based expert systems. The effort in developing the the knowledge base is largely reduced to specifying units from a model library according to the flowsheet structure and specifying the signs of variable and parameter relationships, such as relative temperatures in adjacent process units. A novel feature of the technique is that certain behaviors not apparent from local SDG models but result from global process interactions are automatically represented in the diagnostic knowledge base. These behaviors initially represented as additional arcs derived from the global topology of the SDG are subsequently represented in a form that supports diagnostic reasoning. Representing knowledge about these behaviors allows an accurate diagnosis to be obtained in a dynamic process environment without unnecessary losses of diagnostic resolution.

The following conclusions result from this study:

---

<sup>1</sup>The Model Integrated Analysis System (MIDAS).

<sup>2</sup>The MIDAS Process Model (PM).

## **Steady state qualitative modeling**

- 1) A systematic approach was developed for reducing the ambiguity in steady state qualitative modeling.
- 2) Analytically redundant constraints, some of which can only be derived from global process considerations are required to reduce ambiguity. In an example process, the number of qualitative solutions obtained by utilizing redundant constraint information was reduced by up to two orders of magnitude.
- 3) The need for global constraints in steady state qualitative modeling points out a trade-off between modularity and solution multiplicity.

## **Causal Modeling of Continuous Chemical Processes**

- 4) A novel representation of causality<sup>3</sup> was developed for continuous processes. The ESDG produces both the initial and ultimate process response with "minimum" ambiguity when disturbance propagation is limited to feedforward paths.
- 5) A previous representation of causality (the SDG) excludes inverse and compensatory response of variables when disturbance propagation is restricted to feedforward paths even when exceptions are made for the behavior of controlled variables in feedback control loops.
- 6) Inverse and compensatory response<sup>4</sup> may be exhibited as a result of positive feedback and integral effects associated with negative feedback loops in a process.

---

<sup>3</sup>The Extended Signed Directed graph (ESDG).

<sup>4</sup>Inverse response occurs when the initial and ultimate direction of change of a variable are opposite. Compensatory response is exhibited, if the variable returns to its initial value after all transients have died out.

## **Process Modeling in MIDAS**

- 7) An algorithmic process independent technique was developed, producing MIDAS PMs with the desired properties.<sup>5</sup>
- 8) In an extensive case study, MIDAS produced an accurate initial diagnosis in 93% of the cases, an accurate final diagnosis in 100% of the cases and on average 96% of the malfunctions were eliminated by the end of the transient.
- 9) The acceptable level of performance achieved by MIDAS verified the use of the PM as a diagnostic model for dynamic continuous processes and validated the techniques for deriving the PM.

In the rest of this summary, the major contributions of the thesis are described and directions for future research are suggested.

## **S.2. Steady State Qualitative Modeling**

In order to understand the fundamental limitations of qualitative modeling, an investigation of steady state qualitative modeling was carried out. The steady state qualitative modeling problem is formally stated as that of predicting the ultimate qualitative state of a process in response to process perturbations. Qualitative models of continuous processes may be formulated in terms of qualitative constraints derived from the fundamental principles governing the process. A novel systematic procedure for reducing the inherent ambiguity provided by steady state qualitative constraint models was developed.

### **S.2.1. Simulation of Steady States by Confluences**

The traditional cornerstones of chemical process modeling are quantitative expressions describing fundamental principles such as conservation laws, transport phenomena and thermodynamic equilibrium. In the steady state limit, these expressions reduce to algebraic equations:<sup>6</sup>

---

<sup>5</sup>These properties are stated in the fourth thesis objective.

<sup>6</sup>In distributed parameter systems, algebraic equations may be obtained by discretizing the steady state differential equations.

$$f(\underline{x}, \underline{u}, \underline{p}) = 0; \quad \underline{u} = \underline{u}_0, \underline{p} = \underline{p}_0 \quad (S.1)$$

Process perturbations may be expressed by changes in values of process inputs,  $u$ , or parameters,  $p$ . Qualitative constraints can be derived from the algebraic equations by taking partial derivatives:

$$[\partial f / \partial \underline{x}] \cdot [\Delta \underline{x}] + [\partial f / \partial \underline{u}] \cdot [\Delta \underline{u}] + [\partial f / \partial \underline{p}] \cdot [\Delta \underline{p}] = 0 \quad (S.2)$$

In eq. (S.2), square brackets,  $[.]$ , represent the sign of the argument, and the partial derivatives are the mean values evaluated over process states traversed during the transient. The constraints are referred to by de Kleer and Brown as confluence equations [16].

A constraint satisfaction algorithm utilizing a combination of constraint satisfaction and generate and test was developed to obtain the ultimate qualitative state of the process,  $[\underline{x}]$ . The solution is obtained by solving a system of confluence equations according to the rules of qualitative algebra. Adding quantities of opposite sign results in ambiguity. The consequence of the ambiguity in qualitative modeling is that a set of  $n$  independent conflences in  $n$  unknowns does not necessarily result in a unique qualitative solution. Where multiple solutions result, some of the solutions to the system of conflences may be spurious while others may be genuine and occur for certain quantitative realizations of the system.

### S.2.2. Resolving Ambiguity with Latent Confluences

Qualitative modeling is fundamentally underspecified, as the set of qualitative values (+, 0, -) do not form an algebraic group under qualitative algebraic operations. Therefore, there is no general theory for predicting solution uniqueness or multiplicity.

A novel systematic procedure was developed for reducing the ambiguity inherent in qualitative modeling, thus, eliminating spurious solutions [30]. The ambiguity is reduced primarily by utilizing constraint information, redundant in the original set of quantitative equations. An overspecified set of quantitative equations is required to obtain a fully determined set of conflences. These additional equations are derived by algebraic

manipulation of the independent equation set. Latent confluences (constraints), are additional confluences derived from the redundant quantitative equations.

In general, redundant confluences containing no more, but preferable fewer equations than the independent equations from which they were derived, are less likely to yield ambiguity, thus most useful in eliminating spurious solutions. Confluences involving one or two unknowns do not result in ambiguity. The number of potential latent confluences that can be derived from a set of independent equations increases exponentially with the number of unknowns,  $O(2^n)$ . Because of analytical complexities and the large number of potential latent confluences, a heuristic approach to deriving latent confluences was suggested [30].

The following set of heuristics were employed to guide the search for useful latent confluences:

**Global confluences for conserved quantities:** Qualitative balances for conserved fundamental quantities around subsystems do not necessarily assure conservation of the quantities in the combined subsystem. Redundant confluences derived from overall global balances on combined subsystems reduce the number of spurious solutions. Examples of these fundamental quantities are total mass, species mass and energy.

**Compatibility relationships:** These latent confluences derive from equating driving forces around loops for potential driven flows. Confluences derived from pressure compatibility relationships in bypass or recycle loops are examples of this type.

**Balances from bilinear conservation relationships:** Some conserved fundamental quantities (e.g. energy) are expressed as products of extensive variables (e.g. mass) and intensive variables (e.g. temperature). Eliminating one of the extensive variables (by using an extensive variable balance) from bilinear conservation equations around the subsystem often results in useful latent confluences.

**Eliminating groups of variables:** Characteristic groupings of variables often occur in sets of equations. Eliminating these groups can result in useful latent confluences, especially when the resulting equation contains fewer variables than one of the original equations.

**Causal constraints:** The requirements of causality adds constraints on the behavior of the system. Rules for deriving causal constraints from signed directed graph based representations of causality were developed. These constraints reflect both local causality and the global properties of the signed directed graph.

We were unable to develop a general method of determining a priori, how many and which latent constraints are required to eliminate all spurious solutions. However, in addition to the causal confluences, the number of non-causal latent confluences required to eliminate all spurious solutions in cases where a unique solution is obtained is less than the number of independent equations. Latent confluences involving only one unknown variable (i.e. the analytical solution) are potentially included in the set of redundant equations. Thus, if an analytical solution to the process equations exists, the redundant set of non-causal steady state confluences is theoretically able to provide the qualitative response from all (stable and unstable) steady states without other solutions. The causal confluences provide additional constraints which help eliminate responses from unstable steady states. Causality may also provide latent constraints that are difficult to derive by algebraic manipulation of the independent equation set.

Global balances for fundamental quantities and compatibility relationships are derived using information about global process topography. In addition, some confluences reflect global properties of causal signed directed graph models. The need for confluences derived from global considerations reveal a very important limitation in qualitative modeling, involving modularity. In qualitative modeling, there is a trade-off between modularity and solution multiplicity. Modularity has been an extremely useful concept in conventional modeling and has led to the creation of modular simulation programs. However, the ability to generate global confluences corresponding to particular system topographies is required in computer programs using steady state qualitative models.

### S.2.3. Qualitative Modeling of a Chemical Reactor Process

In order to demonstrate the utility of multiple sources of constraint information in steady state qualitative modeling, the ultimate qualitative behavior of a continuous reactor process in response to malfunctions and disturbances was determined. The process is shown Fig. S.1. An irreversible heterogeneous catalytic exothermic reaction ( $A \rightarrow mB$ ), with rate expression

$$r_A = k_r C_A^n \quad , \quad n > 0 \quad (S.3)$$

takes place in a continuous stirred tank reactor. To provide temperature control, part of the reactor outlet stream is recycled to the reactor through a heat exchanger. The recycle flow rate is controlled, and residence time in the reactor is controlled by maintaining the level of reactants in the reactor.

Although more robust control schemes may exist, the process was chosen because it presents a significant challenge for qualitative modeling. The process possesses several features which tend to generate a great deal of ambiguity. It:

- Is highly non-linear.
- Has interacting control loops.
- Has multiple causal pathways with opposing tendencies between variables.
- Attains multiple steady states in the absence of temperature control.

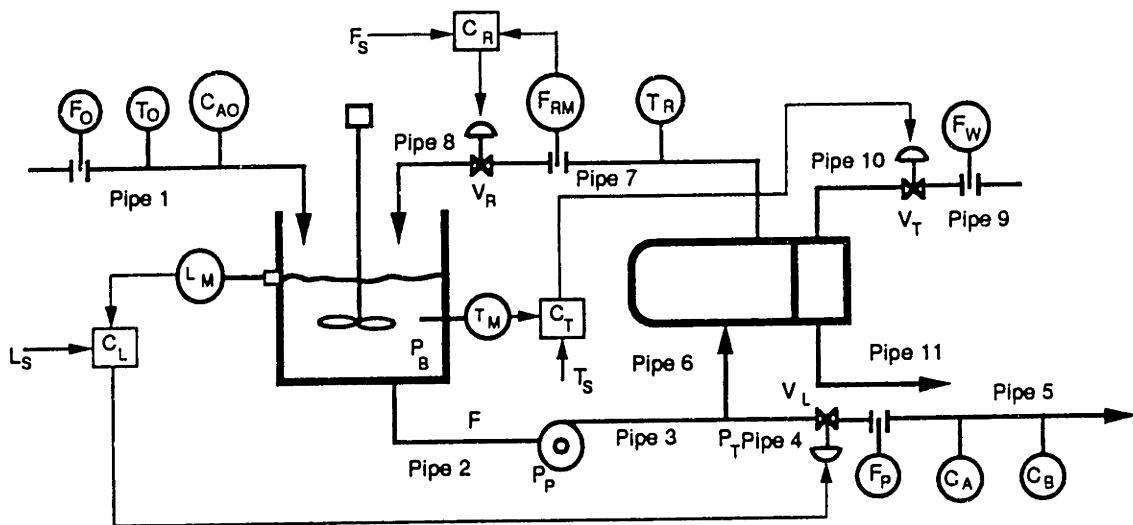


Fig. S.1. Continuous Stirred Tank Reactor with Recycle

Table S.1 shows the number of observable qualitative steady state solutions<sup>7</sup> for a set of 14 measured variables, for a selected set of malfunctions when using:

- (i) causal confluences,
- (ii) confluences derived from an independent set of equations,
- (iii) the combination of confluences from (i) and (ii), and
- (iv) non-causal latent confluences in addition to those in (iii).

**Table S.1 Number of Solutions Obtained Using Confluences from Various Sources**

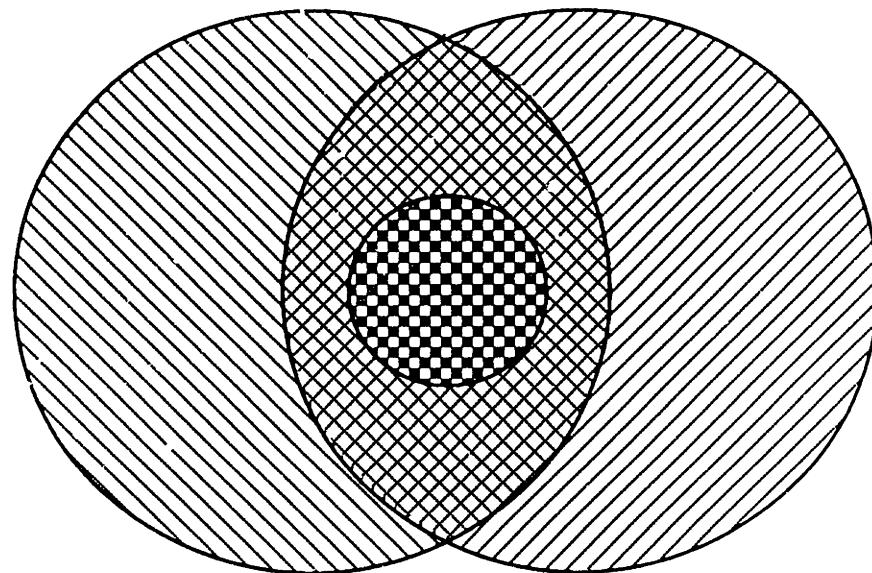
Malfn	Causal (i)	Local non-causal (ii)	Causal & local non-causal (iii)	Local, latent & causal (iv)
1	1	279	1	1
2	1079	984	481	141
3	501	1359	192	13
4	12	287	10	10
5	501	4869	192	13
6	2	279	2	2
7	4	281	4	3
8	1	279	1	1
9	984	1080	178	35
10	309	1215	138	9

The relationship between the solutions (i) - (iv) is most clearly depicted in Fig. S.2. Combining constraints from causal and non-causal sources reduces the number of solutions obtained from the causal confluences by up to a factor of 2 to 3. Adding latent non-causal confluences leads to further reductions of up to an order of magnitude. The overall reduction in the number of solutions obtained between cases (i) and (iv) is up to a factor of 35, and between cases (ii) and (iv) up to a factor of 370. This represents a reduction of over two orders of magnitude. Thus, combining causal and non-causal confluences

---

<sup>7</sup>These solutions are obtained if control loop saturation is allowed.

significantly reduces ambiguity. However, it cannot be proven (without further analysis, possibly numerical) that all solutions obtained for case (iv) are realizable [31].



- causal confluences (case i)
- local non-causal confluences (case ii)
- causal and local non-causal confluences (case iii)
- causal, local and latent non-causal confluences (case iv)

Fig. S.2. Relationship Between Solutions to Different Sets of Confluences

If perfect control is assumed,<sup>8</sup> there is a further reduction in the number of solutions. The confluences produce one solution for all, but the second malfunction. This malfunction produces 3 solutions. It can be shown by further analysis that all 3 solutions are realizable.

In summary, a systematic procedure for reducing the ambiguity in qualitative modeling was developed. It was discovered that redundant constraint information, some of which can only be deduced from global process topography is required in steady state qualitative modeling. Using the redundant constraints, the number of qualitative solutions were reduced by up to two orders of magnitude. The requirement for global constraints points out an inherent trade-off between modularity and solution multiplicity.

## S.3. Causal Modeling of Continuous Processes

Steady state qualitative models may exclude initial and transient measurement patterns and are therefore not appropriate for diagnosis in the dynamic chemical process environment. A novel representation of causality was developed to describe dynamic behavior of continuous process systems. The representation, the Extended Signed Directed Graph, represents both immediate local and "apparent global" causality in the process. Guidelines for specifying local unit causal models were developed.

### S.3.1. Representation of Causality by Signed Directed Graphs

Causality is the principle that effects must be propagated locally within the topology of a system, and is clearly a basic mechanism for human reasoning about process behavior. Causal models generally consist of a representation over which disturbances are allowed to propagate under a certain set of propagation rules. Previous work on causal modeling of continuous process systems has been based on local causal models. The explicit representations provided by the single-stage Signed Directed Graph (SDG) and its related diagrams [20, 21, 22, 23] have proven most useful in diagnostic applications.

---

<sup>8</sup>Perfect control is assumed, except for malfunctions which lead to zero-gain events in control loops.

The SDG of a process consists of nodes, symbolizing process variables and parameters and signed directed arcs representing immediate local cause and effect relationships between variables. Arc signs + and - indicate whether values of cause and effect variables tend to change in the same or opposite directions. Self cycles<sup>9</sup> are associated with variables in the SDG and have a default value of -. Self cycles + and 0 indicate whether the variable is associated with positive feedback or integral effects, respectively. The effect of a process malfunction is represented in the SDG by its initial qualitative effect on the variable that is first causally affected by the malfunction.

As an example, Fig. S.3 shows part of the SDG of a gravity flow tank. The integrator associated with level is represented by a self-cycle with sign 0. Self cycles associated with other variables are negative, and are not shown for the sake of clarity.

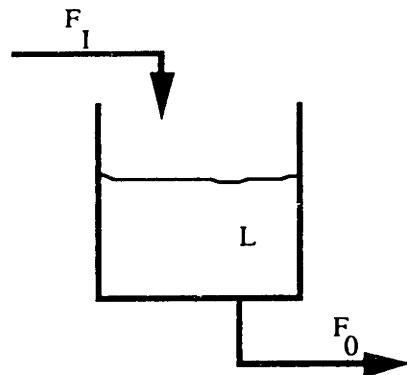


Fig. S.3a. Gravity Flow Tank

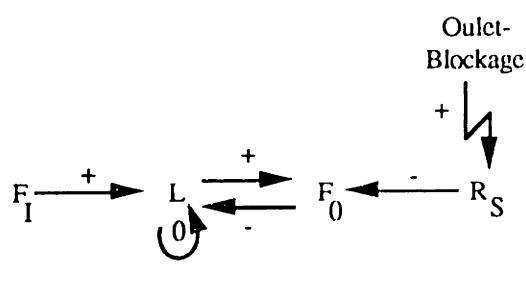


Fig. S.3b. SDG of Gravity Flow Tank

Guidelines for deriving the SDG from a set of algebraic and ordinary differential equations<sup>10</sup> describing the process were developed in Chapter 4 of the thesis. These guidelines address self cycles and provide a more satisfactory treatment for specifying SDG arcs from algebraic equations than the rules described by Palowitch [23]. The approach taken was to decompose the process into its constituent units and then to develop unit SDG models. In developing unit SDG models, a consistent set of interface variables were defined, so models of adjacent units could be linked together according to the process topography.

---

<sup>9</sup>Self cycles are not depicted in SDG models developed by other workers.

<sup>10</sup>Partial differential equations may be discretized to ordinary differential equation by the method of lines.

The guidelines involved specifying causal influences among variables related by a set of equations. For a system of equations, each variable is assigned as the "output" variable to no more than one equation. For each equation, causal arcs initiate from other variables in an equation and terminate at the output variable. Output variables are determined in the order described below. Rules for specifying the signs of arcs terminating at the variable and the associated self cycles are described.

**Independent variables and parameters:** These are variables whose values are externally specified with respect to the unit. These variables are not assigned as outputs to any equation and consequently SDG arcs do not terminate at these variables. The sign of self cycles associated with these variables is -.

**Ordinary differential equations:** The direction of causality between variables in an ordinary differential equation is from the right to the left hand side of the equation

$$\frac{dx}{dt} = f(x, u, p); \quad x = x_0, \quad u = u_0, \quad p = p_0 \quad (S.4)$$

Thus, the output variable in each equation is the variable associated with the differential term. The sign of the arc  $[\beta_{ij}]$  originating at  $x_j$  and ending at  $x_i$  is  $[\partial f_i / \partial x_j]$ . The sign of the self cycle  $[\beta_{ii}]$  associated with  $x_i$  is  $[\partial f_i / \partial x_i]$ . Knowledge of ordinal relationships among variable and parameter values such as temperature differences may be required to specify these signs. When a first order partial derivative is 0, the signs are determined from higher order partial derivatives.

**Output variables of algebraic conservation equations:** Algebraic equations representing conservation relationships derive from approximations to differential equations. For example, the flow/pressure drop relationship describing incompressible fluid flow in a pipe derives from a differential equation describing conservation of momentum (with flow as its output variable). These equations should be converted to their original differential equation form, with the arcs and self cycles specified accordingly.

**Algebraic rate transport equations:** These equations result from driving force relationships for heat and mass transfer.

$$f(x, u, p) = 0; \quad x = x_0, \quad u = u_0, \quad p = p_0 \quad (S.5)$$

Causal arcs initiate from other variables in the equation and terminate at variables denoting the transport rate. If  $x_j$  is the output variable of  $f_i$ , the sign of the arc  $[\beta_{ij}]$  originating at  $x_j$  and ending at  $x_i$  is  $[\partial f_i / \partial x_j]$ . The sign of the self cycle is -.

**Algebraic reaction rate equations:** These equations are empirical relationships relating the rate of reaction to the concentrations of species involved in chemical reactions. Causal arcs initiate from other variables in the equation and terminate at variables denoting the reaction rate. The signs of arcs and self cycles are defined identically as in transport equations.

**Other algebraic equations:** Knowledge about the origin of the equations is required when output variables cannot be assigned uniquely to the remaining algebraic equations. Once output variables have been assigned, the rules for specifying signs of arcs signs and self cycles are identical to those for transport and rate equations.

The guidelines developed in the thesis also specify other attributes of the SDG.

### S.3.2. A Novel Representation of Causality: The ESDG

An important feature that distinguishes chemical process systems from many other physical systems is the existence of complex global topologies with recycle and bypass flows of material and energy and feedback and feedforward transmission of information. These features are reflected as feedback and feedforward paths in the SDG, and result in complex process dynamics.

Previous work on SDGs has not adequately addressed the role of feedback and complex process dynamics. In order to reduce the ambiguity inherent in qualitative modeling, O'Shima [32], limits disturbance propagation to feedforward paths in the SDG. This assumption excludes non-monotonic behaviors, namely, inverse and compensatory responses that arise from global interactions in negative feedback loops. For example, process behavior is excluded in feedback control loops, where the effect of a disturbance is passed to the manipulated variable of a controller, without an apparent deviation of the controlled variable. Therefore, the diagnosis obtained by limiting propagation to feedforward paths excludes the correct failure origin in situations where controllers compensate for effects of disturbances on controlled variables. Recognizing that feedback

control is ubiquitous in chemical processes, Iri et al. [20], adopt a heuristic that makes exceptions for this instance of apparent "global" causality. Similar exceptions have been made by Tsuge et al. [33] and Ulerich and Powers [34]. However these heuristics are incomplete as they do not adequately account for all instances of non-monotonic behavior arising from global feedback effects [30].

A rigorous global analysis, resulting in theorems for identifying variables that exhibit non-monotonic behavior due to negative feedback effects was performed. The Extended Signed Directed Graph (ESDG) derives from the result of the analysis. In addition to arcs of the SDG, the ESDG contains additional arcs which account for instances of non-monotonic behavior observed in the process. The set of variables exhibiting non-monotonic behavior in response to specific disturbances determines the location of the additional arcs in the ESDG. Process behavior is determined using the ESDG by restricting disturbance propagation to feedforward paths. In the following, the global analysis is outlined.

The linearized version of the ordinary differential equations describing the process are:

$$\frac{dx}{dt} = A \cdot x + B \cdot u \quad (S.6)$$

Here,  $u$  represents process perturbations. On neglecting higher order terms, the qualitative response for each variable may be expressed as:

$$[dx_i] = [b_i t + \sum_{j \neq i} a_{ij} b_j t^2 / 2! + \sum_{k \neq j, i} \sum_{j \neq i} a_{ik} a_{kj} b_j t^3 / 3! + \dots] \cdot [du] \quad (S.7)$$

The sign of each product  $b_i, a_{ij}b_j, \dots$  in eq. (S.7) is the sign of an feedforward path of length  $r$  ( $r = 1, 2, \dots$ ) from  $u$  to  $x_i$  in the SDG. The summation covers all possible feedforward paths. The initial response is obtained by taking limits as  $t \rightarrow 0$ . From eq. (S.7), it can be deduced that:

- The SDG accounts for the initial qualitative response of the system if disturbance propagation is restricted to feedforward paths.

Assuming no transitions to other qualitative regimes (such as controller saturation), a detailed qualitative matrix analysis leads to an expression for the ultimate response.

$$\begin{aligned}
[dx_i] = & [ b_i(-1)^{n-1} |P_i| + \sum_{j \neq i} a_{ij} b_j (-1)^{n-2} |P_{ij}| \\
& + \sum_{k \neq i,j} \sum_{j \neq i} a_{ik} a_{kj} b_j (-1)^{n-3} |P_{ijk}| + \dots + \\
& + \sum_{p \neq \dots} \sum_{j \neq i} a_{ik} \dots a_{pj} b_j (-1) |P_{ijk\dots p}| \\
& + \sum_{q \neq \dots} \sum_{j \neq i} a_{ik} \dots a_{pq} a_{qj} b_j |P_{ijk\dots pq}| ]. [du]
\end{aligned} \tag{S.8}$$

Here,  $(-1)^{n-2} |P_{ij}|$  is the signed principal minor obtained by deleting elements in the  $i$ th and  $j$ th row and columns of  $A$ , etc. The ultimate response of some variables may be excluded by limiting propagation to feedforward paths if qualitative values of the signed minors are non positive. On further analysis, theorems identifying when the ultimate response may be excluded were derived. The theorems indicate situations where the ultimate response may be excluded due to the global topology of the SDG. The ultimate response is excluded in the following:

- Where **compensatory response** of a variable occurs as a result of the effect of integrators within negative feedback loops.
- When **inverse response** of a variable occurs as a result of positive feedback effects associated with a negative feedback loop.

When transitions to other qualitative regimes such as controller saturation take place, an extension to the theorems indicate that the ultimate response is excluded:

- When **inverse response** of a variable is exhibited due to the inhibition of the effect of an integrator<sup>11</sup> that suppresses positive feedback effects in a negative feedback loop.

Process examples describing situations where the previous heuristics fail are:

- Compensatory response of outlet flow to a partial blockage in the outlet of a gravity flow tank.

---

<sup>11</sup>For example, the effect of the integrator is inhibited by saturation of control loops.

- Inverse response of reactant concentration to a perturbation in reactant feed concentration for an irreversible exothermic reaction taking place in a continuous stirred tank reactor.<sup>12</sup>
- Compensatory response of a manipulated variable when an external disturbance enters a control loop at the manipulated variable (process effluent flow through the preheater in a feed preheater process).

These situations are depicted in Figs. S.4, S.5, and S.6, respectively. In these situations, the ESDG correctly accounts for process behavior.

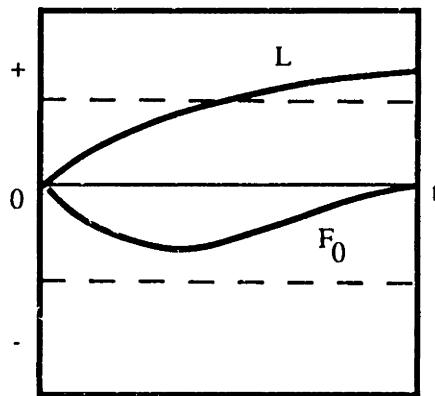


Fig. S.4. Response of Gravity Flow Tank  
to Outlet Blockage

In summary, when using previous disturbance propagation assumptions, the SDG excludes the actual behavior or produces spurious behaviors in continuous processes. In order to overcome these limitations, a novel representation of causality, the ESDG, was developed. The ESDG represents both local and apparent global causality in the process. Situations where apparent global causality may be observed were identified by a rigorous global topological analysis of the SDG. By limiting disturbance propagation to feedforward paths, the ESDG is guaranteed to predict all initial and ultimate behaviors while minimizing the production of spurious behaviors. In addition, most transient behaviors are accounted for by the ESDG. This suggests the use of the ESDG as the basis for a dynamic model of process behavior.

---

<sup>12</sup>The temperature control loop is saturated or incorrectly tuned.

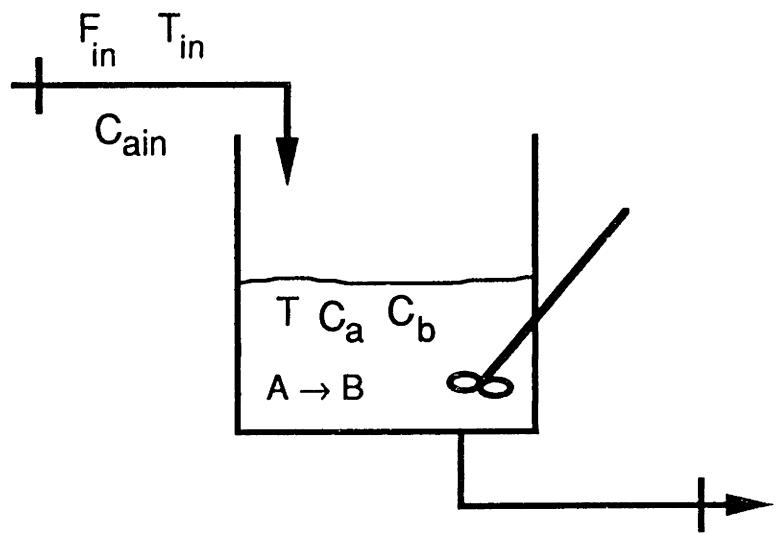


Fig. S.5a. Continuous Stirred Tank Reactor  
(Temperature and Level Control Loops not Shown)

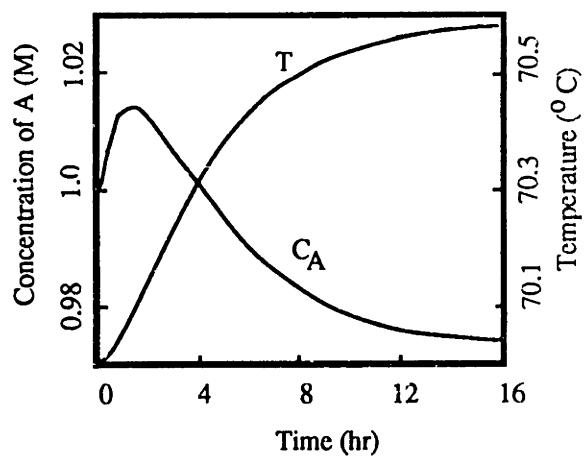


Fig. S.5b. Response to High Feed Concentration

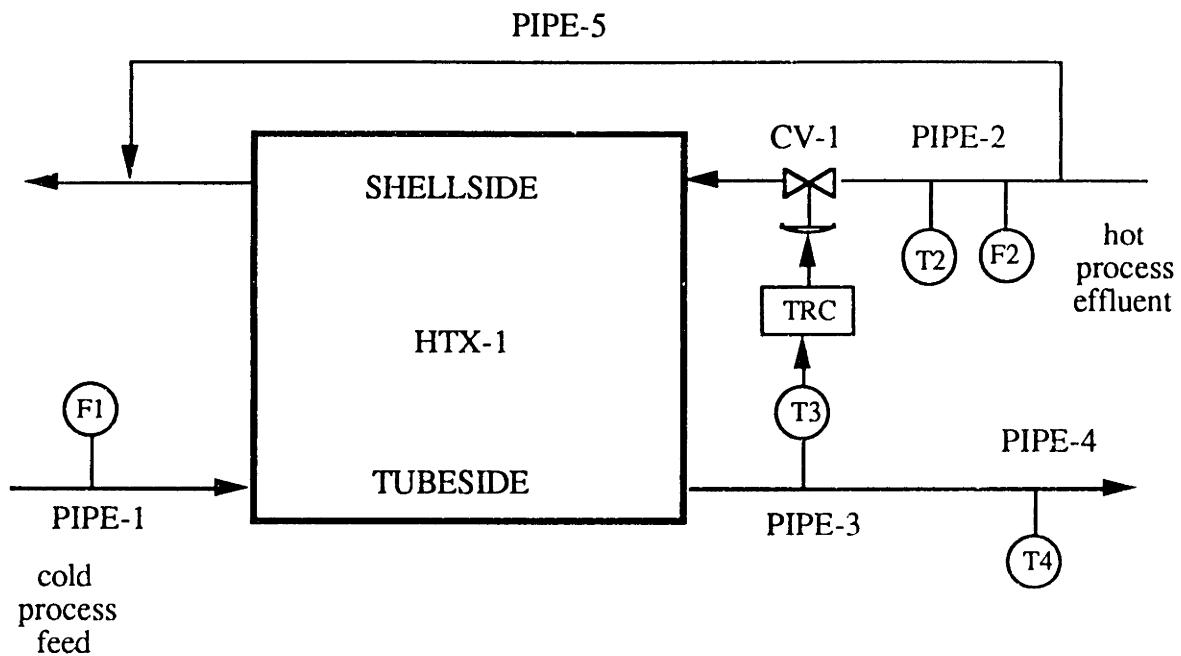


Fig. S.6a. Process Feed Preheater

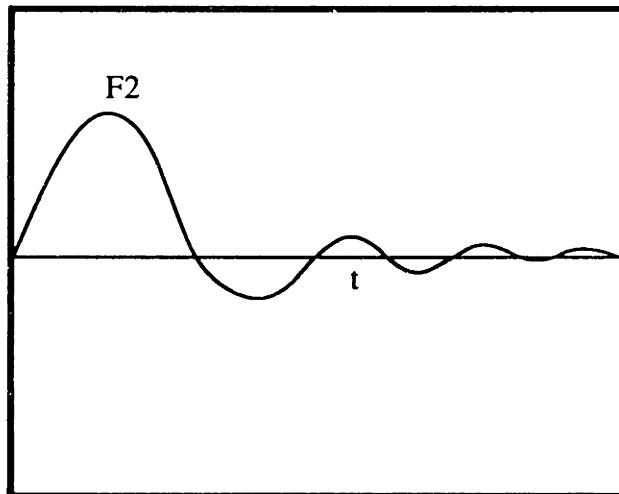


Fig. S.6b. Response of Process Effluent Flow through Preheater Shell to Partial Blockage in Pipe-5

## S.4. Process Modeling in MIDAS

The Model Integrated Diagnostic Analysis System (MIDAS) [35], is an on-line operator aid for malfunction diagnosis in continuous chemical and refinery processes. MIDAS specifically addresses problems not treated in previous diagnostic systems including the: representation of non-monotonic process dynamic behaviors and its utilization as a source of diagnostic information; diagnosis of multiple malfunctions, particularly simultaneous independent malfunctions and induced sensor failures; and the robustness of diagnostic conclusions to symptom variations. MIDAS is currently implemented in Common LISP on a 386-class PC using the Goldworks<sup>13</sup> Expert System Development Environment.

Figure S.7 shows the basic structure of MIDAS. Once incoming process data has been translated into discrete events by MIDAS monitors, the MIDAS plant independent inference strategy utilizes knowledge about relationships expressed in the process diagnostic knowledge base to make diagnostic conclusions [36].

### 4.1. The MIDAS Process Model

The MIDAS knowledge base is referred to as the Process Model (PM). The PM consists of qualitative causal and quantitative constraint relationships between process measurements and malfunctions potentially affecting the process. Relationships expressed in the PM are in a form that readily supports diagnostic reasoning.

Qualitative relationships in the PM are represented by (i) directional links between potential malfunctions and events associated with measured variables and (ii) directional links between events associated with measured variables. Specific diagnostic conditions are attached to the directional links between events. Several types of conditions may be expressed in the PM, the most common is the :not condition used to eliminate malfunctions from the current hypothesis. For example, the link between events **f2-normal-to-low** and **I-normal-to-high** in the PM (Fig. S.9) of the gravity flow tank process of Fig. S.8 indicates that **I-normal-to-high** may be observed as a consequence of **f2-normal-to-low**.

---

<sup>13</sup>Trademark, Gold Hill Computers, Cambridge, MA.

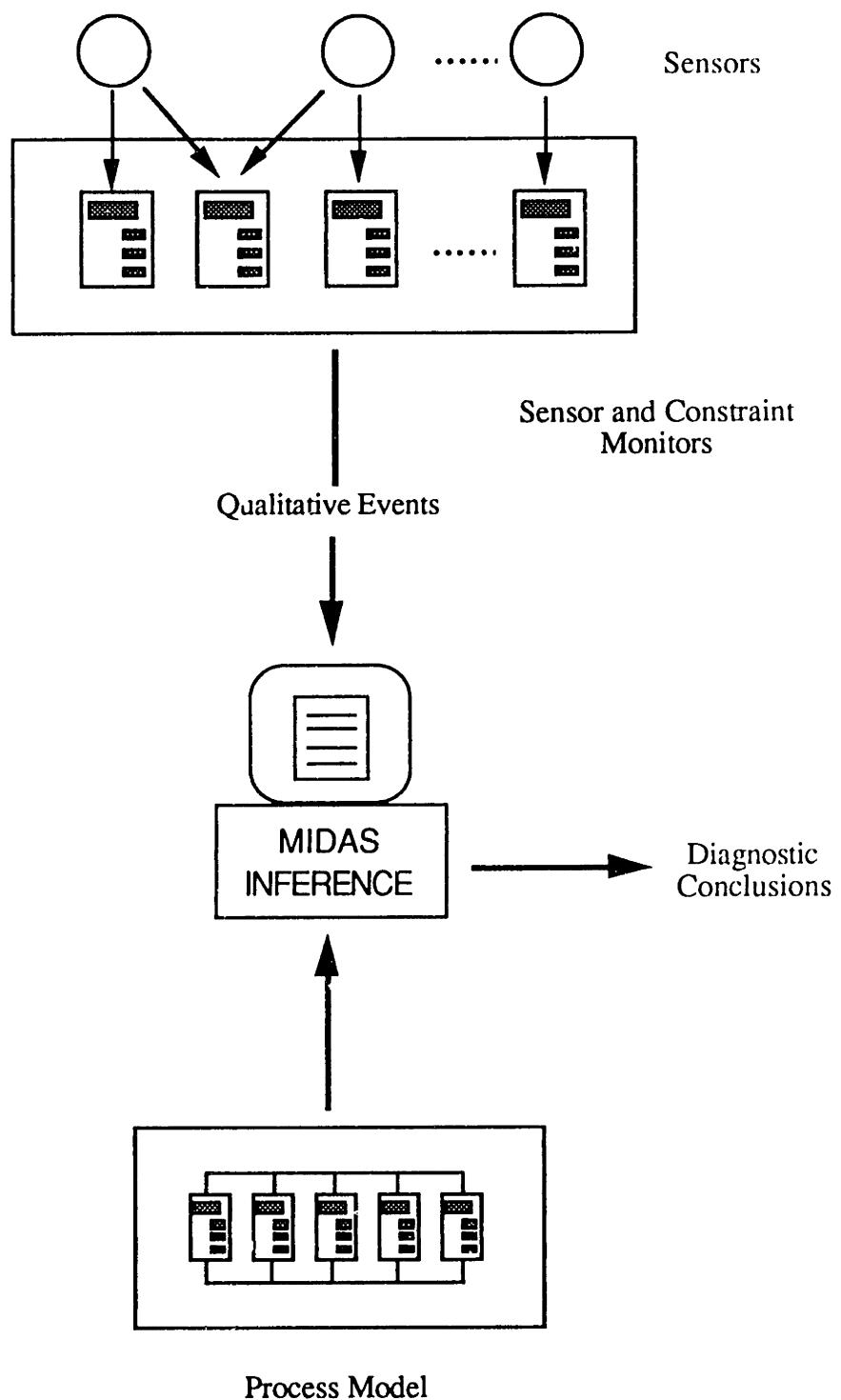


Fig. S.7. Basic Structure of MIDAS

The condition associated with the link,

(:not (tank-1-outlet-leak f2-sensor-failed-low))

is interpreted as follows: If during inference, the occurrence of **I-normal-to-high** is attributed to **f2-normal-to-low**, both malfunctions should be eliminated from the hypothesis set as malfunctions in f2-sensor and the tank leak cannot lead to the event **I-normal-to-high**.

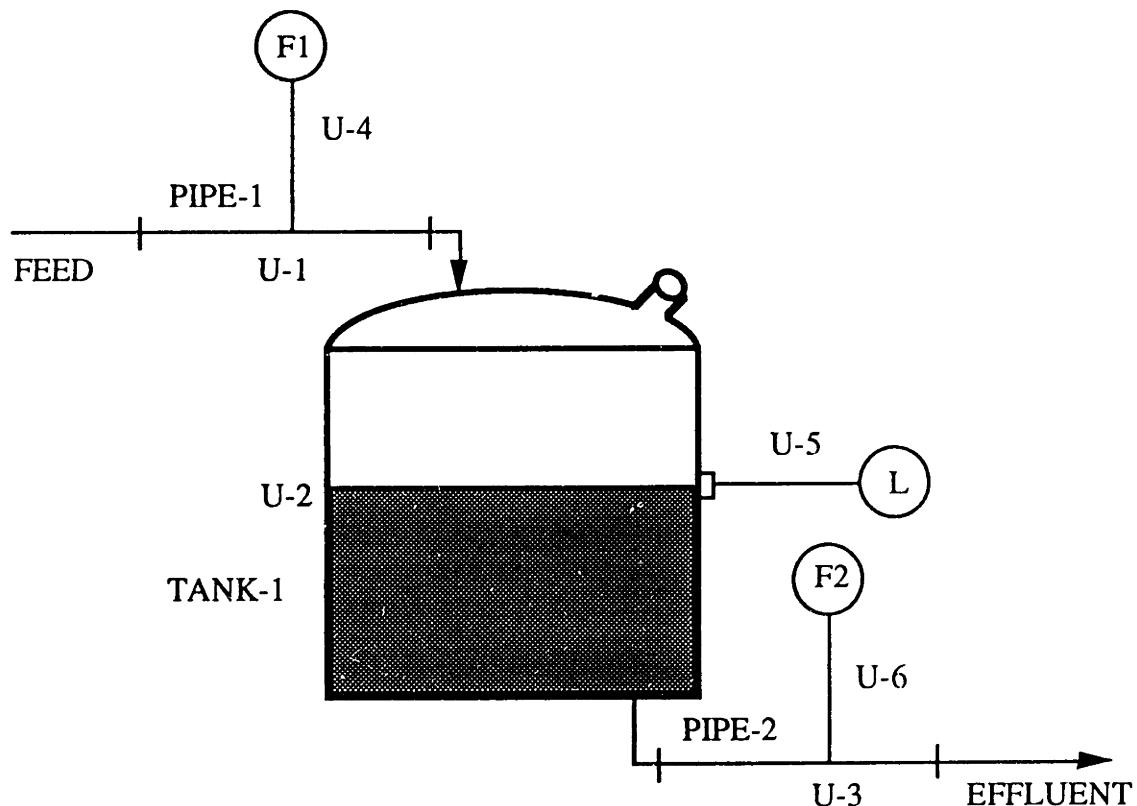


Fig. S.8. Gravity Flow Tank Process

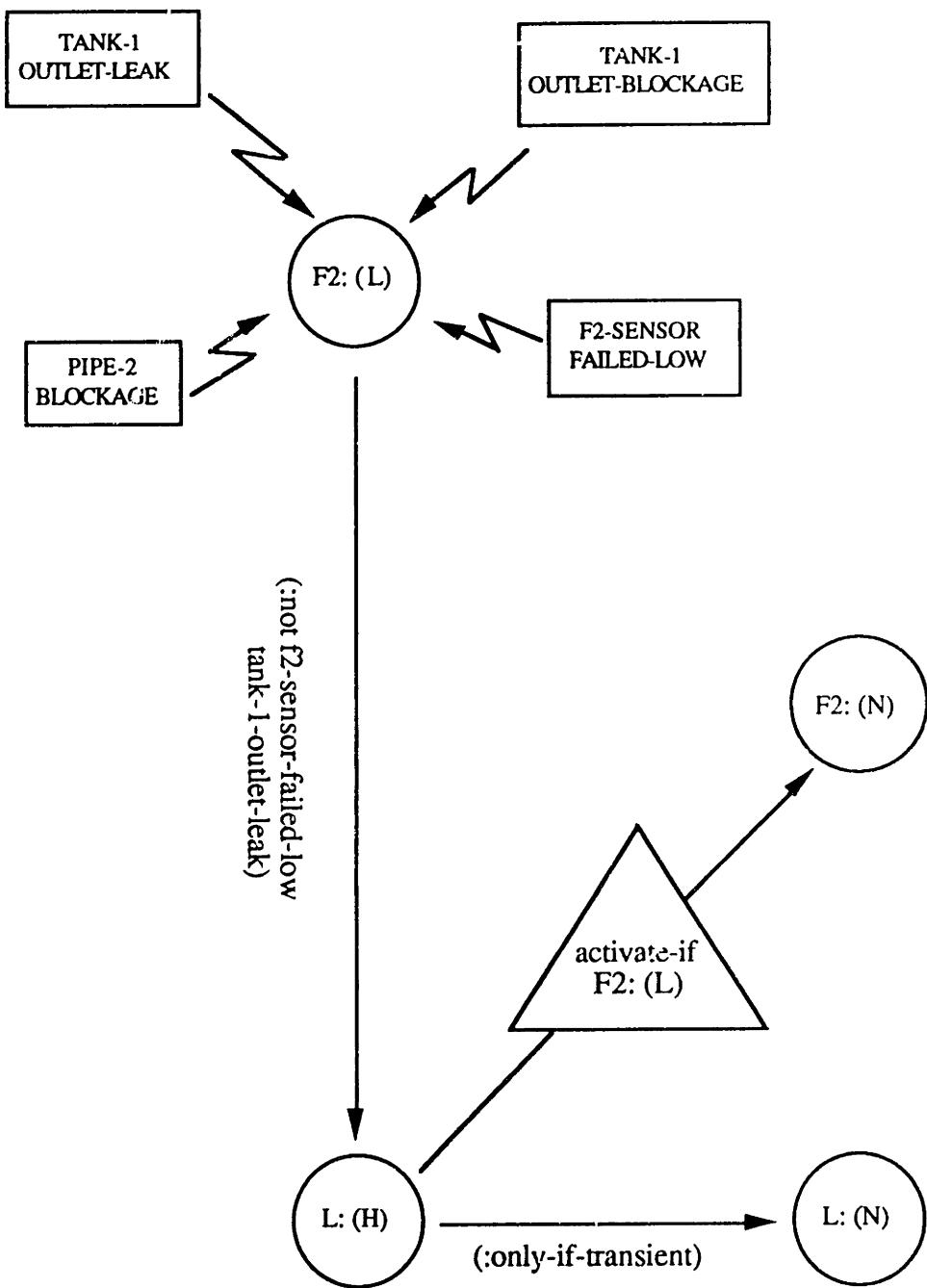


Fig. S.9. PM for Gravity Flow Tank Process

The representation of constraint information is similar to qualitative relationships in the PM. Directional links and their associated diagnostic conditions relate observable process constraint residuals to process malfunctions. Specifying constraints in the PM potentially improves diagnostic resolution when compared to the diagnosis obtained by using qualitative relationships alone.

In MIDAS the PM is implemented using the Goldworks frame-based representation scheme. The following types of frame objects comprise the PM.<sup>14</sup>

**Potential-Root-Causes:** These symbolize the malfunctions that potentially affect the process. Associated with **potential-root-causes** are the list of events observed first as a result of the malfunction. Directional local cause links<sup>15</sup> initiate from **potential-root-causes** to these events.

**Potential-Events:** These indicate events associated with measured variables and constraints. Associated with **potential-events** are the **potential-root-causes** for which the event is the first event observed. These **potential-root-causes** are known as the local causes of the event.

**Compiled-Causal-Links:** These represent directional links between potential event objects. Diagnostic conditions are attached to these links. Also attached to a subset of these links are the conditions under which each link may be activated. These links are referred to as activatable links.

The PM readily supports diagnostic reasoning. The directional links and their associated diagnostic conclusions provide explicit relationships between process observations and diagnostic conclusions. The initial hypothesis set consists of malfunctions from which links initiate to the first observed event. Diagnostic conditions associated with links between events allow the current hypothesis to be updated based upon subsequent observations of events.

---

<sup>14</sup>The following description is simplified for the clarity of presentation. The actual representation used in MIDAS includes more object types.

<sup>15</sup>Local cause links are represented as attribute values of other objects in the PM.

## S.4.2. Developing the Process Model

A structured, process independent, algorithmic technique for deriving the qualitative relationships expressed in the PM was developed. The process ESDG is derived as an intermediate step in the derivation of the PM. An input specification language is provided to supplement information expressed by the qualitative core of the PM with constraint relationships.

The procedure for developing the PM has the following advantages. It:

- Reduces the effort required in representing diagnostic information about the process. When compared with conventional rule based expert systems, the effort required in developing the PM is largely reduced to selecting and specifying units the flowsheet structure and certain variable and parameter relationships such as relative temperatures in adjacent units. The considerable effort required for parameter estimation for quantitative modeling approaches is avoided.
- Significantly improves the reliability and consistency of the diagnostic knowledge base when compared with rule based expert systems.
- Correctly represents non-monotonic process behaviors in the PM. An accurate diagnosis is obtained in a dynamic process environment without unnecessary losses of diagnostic resolution.
- Allows diagnostic conclusions to be reached based on the existing process instrumentation scheme. Values of unmeasured variables are not required for diagnostic inference.
- Allows a "family" of PMs to be created by utilizing varying amounts of process specific information. Thus, diagnostic conclusions may be "tailored" to the amount of available quantitative process information.

The procedure consists of four major steps. An overview of the procedure is shown in Fig. S.10.

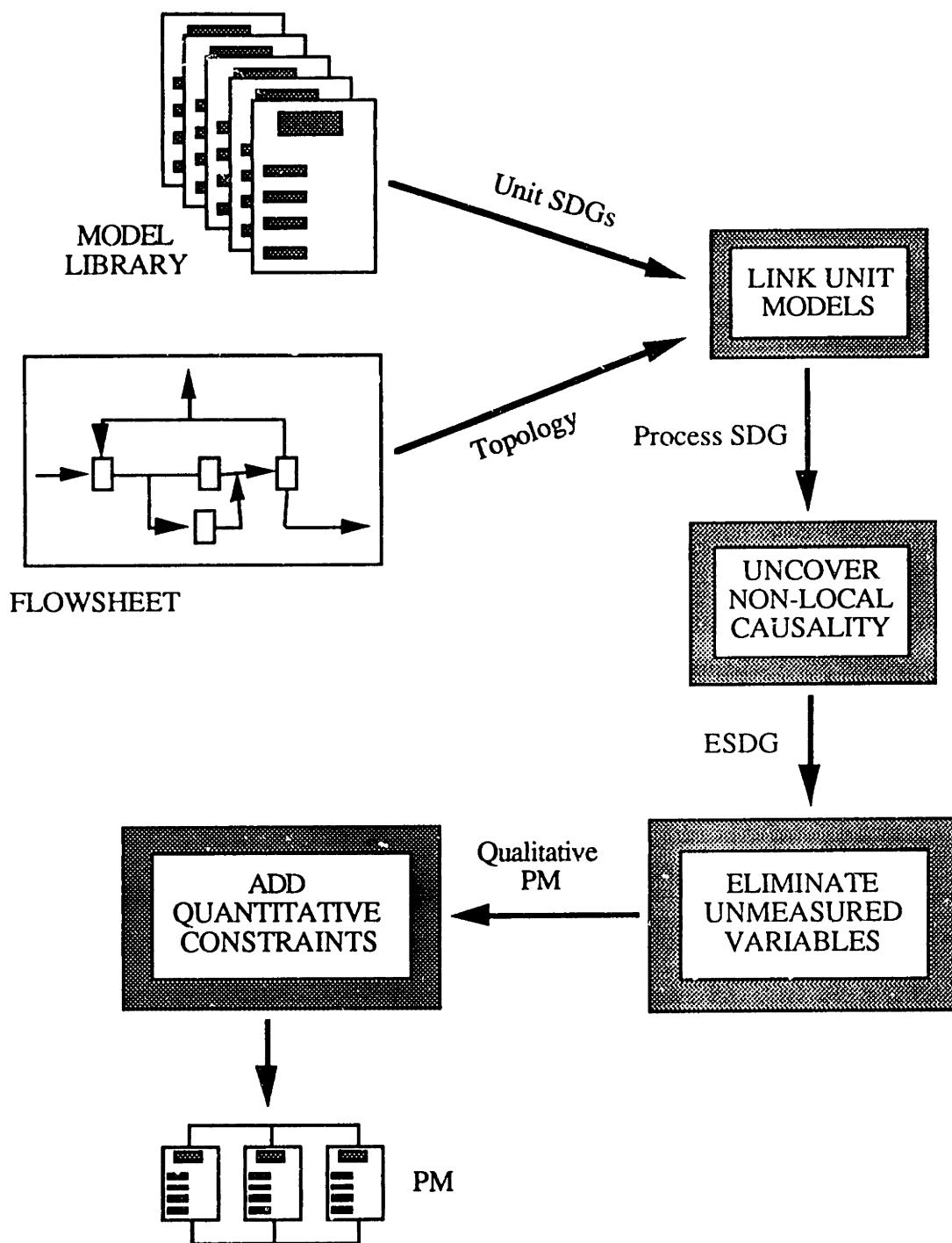


Fig. S.10. Steps in Constructing the PM

Each step is described in more detail below.

**Step 1. Developing the process SDG:** The first step in developing the PM is to specify local causal relationships among variables in the process. Local causality is represented by the SDG as was discussed in Section S.3.1. This step involves selecting, instantiating and linking process units from a model library according to the flowsheet topography. The model library consists of algorithms that create SDG models for 13 commonly occurring types of units such as heat exchangers, reactors and process controllers. While instantiating unit models information about the signs of certain variable and parameter relationships such as the relative temperatures in adjacent process units are specified. An input specification language is provided to enter unit SDG models for unit types not included in the library.

**Step 2. Creating the process ESDG:** The second step in developing the PM is to identify instances of apparent global causality that result from global feedback interactions in the process. These interactions may result in the exhibition of non-monotonic dynamic behavior by variables in the process. Apparent global causality is represented as additional arcs in the ESDG. An algorithm that identifies all instances where non-monotonic behavior may be exhibited in the process and creates additional ESDG arcs was implemented. The algorithm is based on theorems derived from the global qualitative analysis described in Section S.3.2. However, the theorems relating to process behavior during transitions between qualitative regimes (such as controller saturation) were not implemented. Representing apparent global causality by additional arcs<sup>16</sup> of the ESDG allows an accurate diagnosis to be obtained in a dynamic process environment. Restricting disturbance propagation to feedforward paths while carrying out inference with the ESDG ensures that diagnostic resolution is not degraded unnecessarily.

**Step 3. Deriving qualitative relationships in the PM:** The next step is to translate the ESDG into the qualitative relationships expressed in the PM. Unmeasured variables are eliminated from the ESDG and causal pathways coalesced to derive the qualitative relationships expressed in the PM. An algorithm for translating the ESDG to qualitative relationships in the PM was implemented. Interactive user input may be used to resolve ambiguities resulting from merging pathways of opposite sign. Eliminating unmeasured variables from the ESDG allows a diagnosis to be made with the existing

---

<sup>16</sup>These arcs serve to explain instances of non-monotonic process behavior.

process instrumentation scheme. Eliminating unmeasured variables also results in a more efficient on-line diagnosis as assumptions about states of unmeasured variables do not have to be postulated or retracted during inference.

**Step 4. Specifying quantitative constraints in the PM:** The final step in developing the PM is to enhance the qualitative relationships in the PM with relationships between constraint residuals and process malfunctions. Typical constraints include conservation equations and equipment performance specifications. An input specification language is provided to specify knowledge of these relationships in the PM. Guidelines for deriving relationships between the constraint residuals and process malfunctions were developed. Specifying knowledge about constraint relationships in the PM allow a family of PMs to be created by utilizing varying amounts of process specific information.

### S.4.3. Process Model Verification: MIDAS Case Study

In order to verify the PM as a diagnostic model for use in dynamic continuous chemical processes, an extensive case study detailing the performance of MIDAS was carried out. Based on a sample of 77 runs for a simulated jacketed reactor process, (i) an accurate diagnosis was achieved in 93% of the cases after the initial event, (ii) an accurate diagnosis was achieved in 100% of the cases, at the end of the transient, and (iii) on average, 96% of the malfunctions in the process were eliminated by the end of the transient. These results indicate very good diagnostic performance. The acceptable level of performance verifies the PM as a diagnostic model for dynamic process environments and validates the techniques for developing the PM.

#### S.4.3.1. PROCESS MODEL OF A JACKETED REACTOR PROCESS

The process selected for the study was the jacketed reactor process shown in Fig. S.11. The process consists of a continuous stirred tank reactor in which two liquid phase, catalyzed, irreversible, first order reactions ( $A \rightarrow B$ ) and ( $A \rightarrow C$ ) take place. The primary reaction, ( $A \rightarrow B$ ), is exothermic. The endothermic side reaction, ( $A \rightarrow C$ ), is slow and nominally accounts for 0.23% of the consumption of A. The reacting mixture is cooled by cooling water flowing through the external jacket. To provide temperature control,

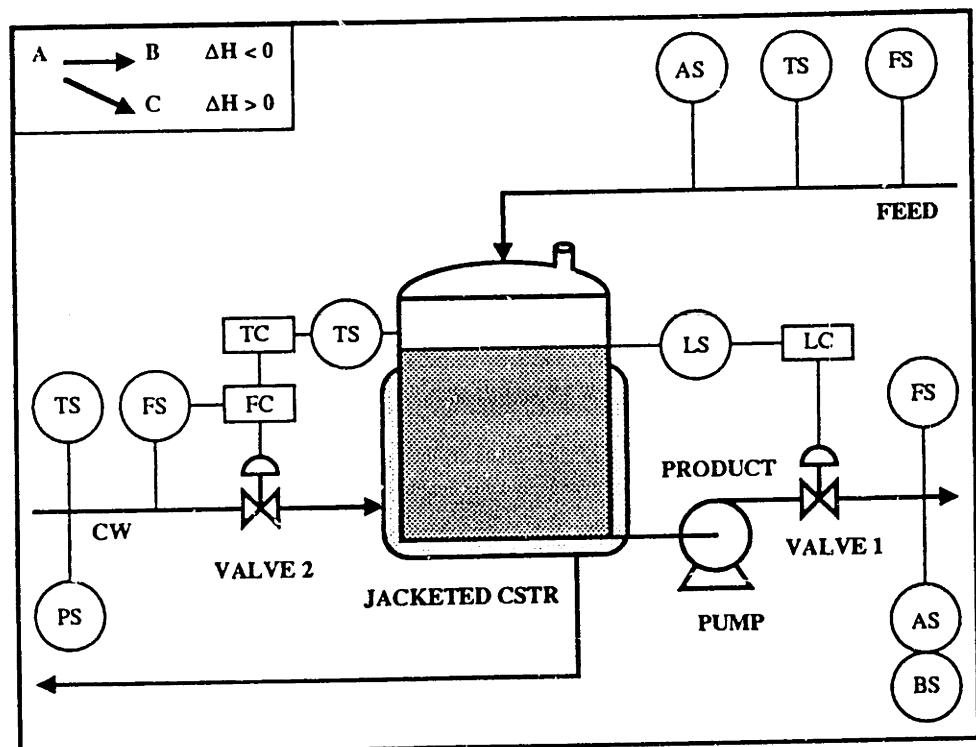


Fig. S.11. Jacketed Continuous Stirred Tank Reactor Process

a cascade control system using measurements of the reactor temperature and cooling water flowrate manipulates the flow of cooling water to the jacket. Temperature is the primary controlled variable of the cascade loop and the set point of the cooling water flow controller is determined by the output of the temperature controller. The residence time of reactants in the reactor is controlled by maintaining the level of the reactants in the reactor.

Although limited in size, this process has several features which present a challenge for dynamic process diagnosis. A unique malfunction in the process can lead to different symptoms depending on its magnitude and extent. Different malfunctions may also result in the same set of symptoms. Furthermore, the process:

- Is highly non-linear.
- Has interacting control loops.
- Has multiple causal pathways with opposing tendencies between variables. Positive feedback and non-control integrating effects are additional sources of non-monotonic dynamic behavior by the variables.
- Attains multiple steady states in the absence of temperature control.

Qualitative relationships in the PM were derived from 22 library unit models using the algorithms described in the previous section. 48.6% of the ambiguous relationships that occurred as a result of merging causal pathways of opposite sign were resolved during model creation. Creation of the qualitative PM required approximately three man-hours and seven computer-hours on a PC compatible 386/20 MHz computer. Relationships between process malfunctions and four constraint equations were specified, supplementing the qualitative relationships. The constraints were:

- An overall material balance for the reactor,
- A pressure drop equation for the cooling water stream,
- A pressure drop equation for the reactor outlet stream, and

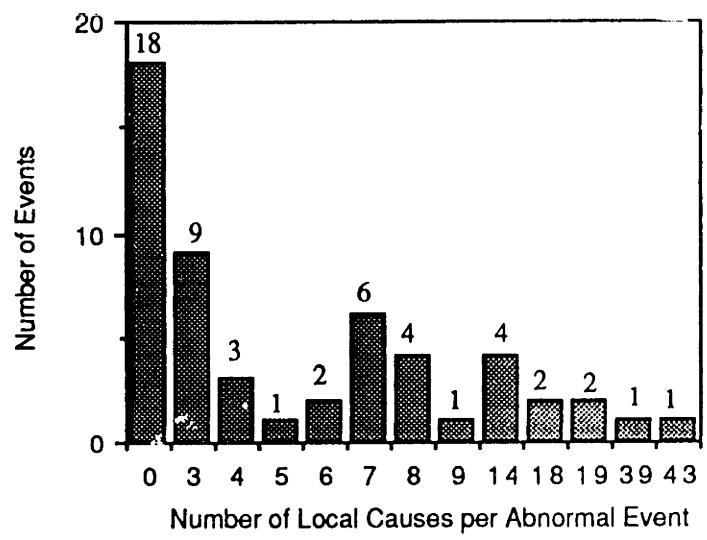


Fig. S.12a. Distribution of Local Causes

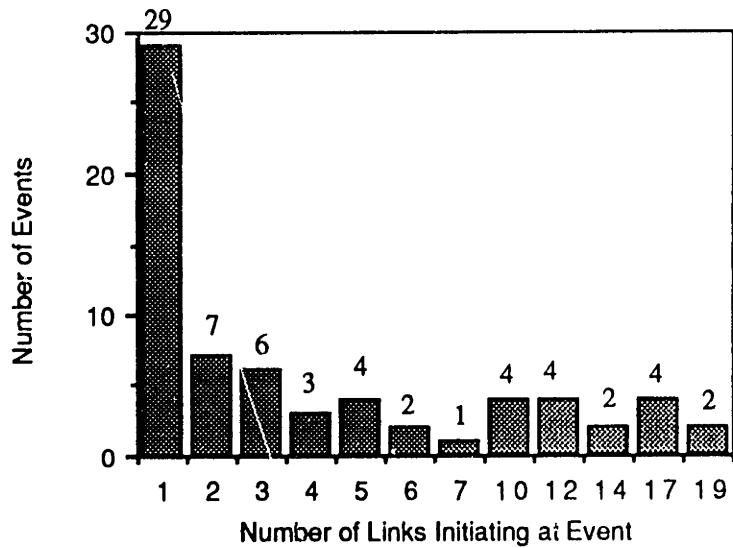


Fig. S.12b. Distribution of Compiled Causal Links

- A chemical species balance for the reactor.

The final PM for the jacketed reactor process included:

- 99 Potential-Root-Cause instances,
- 144 Potential-Event instances,
- 334 Compiled-Causal-Link instances (58 activatable links).

The connectivity of the PM is indicated in Figs. S.12a and S.12b. A distribution of local causes for event transitions from the normal state (i.e. abnormal events) is shown in Fig. S.12a. The average number of local causes per event is 9.75. Figure S.12b shows a distribution of the number of compiled causal links initiating at each event. On average, 4.91 links initiate at each event.

#### S.4.3.2. CASE STUDY

The process model for the jacketed reactor process was verified by evaluating the diagnostic performance of MIDAS using numerical data from a simulation of the process as input. A FORTRAN 77 dynamic simulation program for the process was developed by Finch [36]. The program simulates the effects of 106 malfunctions on the process and produces output files containing the simulated sensor data. Malfunction modes that may be simulated include biases and fixed failures in sensors and controllers, process malfunctions and external process disturbances. Constraint residuals are calculated by the program and their values are also recorded in output data files. Noisy process data is simulated by adding random errors to simulated measurement values.

Before performing the study, the monitors in the PM were tuned using malfunction-free data generated by the simulation program. In tuning the monitors, numerical values of parameters associated with the variable or constraint were determined.

Process data for a randomly selected set of 100 malfunctions was generated. The extents and rapidity for each malfunction were also assigned randomly. Of the 100 malfunctions, 23 could not be used to evaluate MIDAS' performance either because they were beyond the

scope of the simulation, their extents were too small to produce observable events or the malfunctions were unobservable given the existing process instrumentation.

Data from the remaining 77 simulation runs formed the basis for the case study. Data output for each run was for 90 minutes, with the malfunction introduced after 15 minutes of normal operation. The data for each simulation run was input to MIDAS using its monitors.

The recorded-event sequences produced by the data were diagnosed using the MIDAS' inference algorithms developed by Finch [36]. After each recorded-event the current state of the diagnostic hypothesis was recorded. The current state of the diagnosis includes:

- A list of clusters, each cluster representing an independent malfunction.
- A list of root cause candidates for each cluster. Associated with each candidate in the list is its relative ranking.

#### S.4.3.3. CASE STUDY RESULTS AND ANALYSIS

One objective of the techniques for developing the PM was to create PMs in which explicit knowledge about non-monotonic dynamic behavior could be represented and utilized as a source of diagnostic information. Knowledge about these kinds of behavior should be represented so that an accurate diagnosis with a high degree of resolution is produced. At least one measured variable displayed compensatory response in 31 of the runs, representing 40% of the total number of runs. Inverse response was exhibited by at least one variable in 16 (21%) of the cases. The significant number of cases for which non-monotonic behavior was exhibited provides a basis for verifying the process model and evaluating diagnostic performance in a dynamic process environment.

The results of the case study indicated that (i) an accurate diagnosis was achieved in 93% of the cases after the initial event, (ii) an accurate diagnosis was achieved in 100% of the cases, at the end of the transient, and (iii) on average, 96 % of the malfunctions in the process were eliminated by the end of the transient.

The average diagnostic performance of MIDAS is detailed in Table S.2. The entries in each row are based on the number of cases in which at least the indicated number of events is observed.

Table S.2 Average Diagnostic Performance of MIDAS

Events Observed	No of Cases	Accuracy (% of Cases) AC	Average Resolution RE	Average Performance $\Phi$
1	77	93	0.892	0.87
2	69	100	0.950	0.93
3	60	100	0.960	0.94
4	55	100	0.964	0.96
5	50	100	0.968	0.96
6	39	100	0.966	0.94
7	30	97	0.960	0.93
8	24	100	0.954	0.92
9	19	100	0.944	0.92
10	13	100	0.932	0.91

From the table, it can be seen that MIDAS produces an accurate diagnosis most of the time. Where an accurate diagnosis was obtained, the true malfunction was included in the candidate set. There were 6 cases in which an inaccurate diagnosis was obtained; 5 cases after detection of the 1st event and 1 case, after the 7th event was detected. The inaccurate diagnosis obtained in these cases was not due to errors in the PM, but due to inadequacies in the event detection scheme. In all 5 cases where an inaccurate diagnosis was produced

after the 1st event, the observed event did not occur in its expected causal order.<sup>17</sup> The case where an inaccurate diagnosis was obtained after the 7th event was a situation where **apparent inverse response** was exhibited by a variable in a control loop. This is discussed further in Section S.4.4.

There was a noticeable drop in performance for cases where more than 6 events were observed. This is indicated by the average resolution and performance observed in cases with more than 6 events.

The resolution, RE, is defined in terms of the number of root causes, Nr, that have a relative ranking equal to or greater than the relative ranking of the true malfunction.

$$RE = (N_t - Nr)/(N_t - 1) \quad (S.9)$$

In eq. (S.9), Nt is the total number of malfunctions that may affect the process. Nr corresponds to the maximum number of malfunctions that would be examined by the operator before identifying the true malfunction, if the candidate root causes were examined according to their relative rankings. The performance parameter,  $\Phi$ , is defined as the product of the accuracy and resolution.

$$\Phi = AC \cdot RE \quad (S.10)$$

A value of 1 indicates a perfect diagnosis, while 0 indicates a useless diagnosis. A useless diagnosis is obtained either when the diagnosis is inaccurate ( $AC = 0$ ) or the candidate set contains all malfunctions in the process ( $Nr = N_t$ ).

The drop in performance is primarily due to the formation of "active"<sup>18</sup> causal loops in inferred malfunction clusters [36] and the inability of the inference algorithm to identify source events in the loop. This is addressed in Section S.4.4.

---

<sup>17</sup>The robustness features of MIDAS' inference [36] did not result in an accurate diagnosis because a sub-threshold event was not detected for the first causally expected event. A remedy is to retune the affected monitors so that the sub-threshold values are closer to the nominal value.

<sup>18</sup>A causal loop was activated when a variable in the temperature control loop exhibited apparent inverse response.

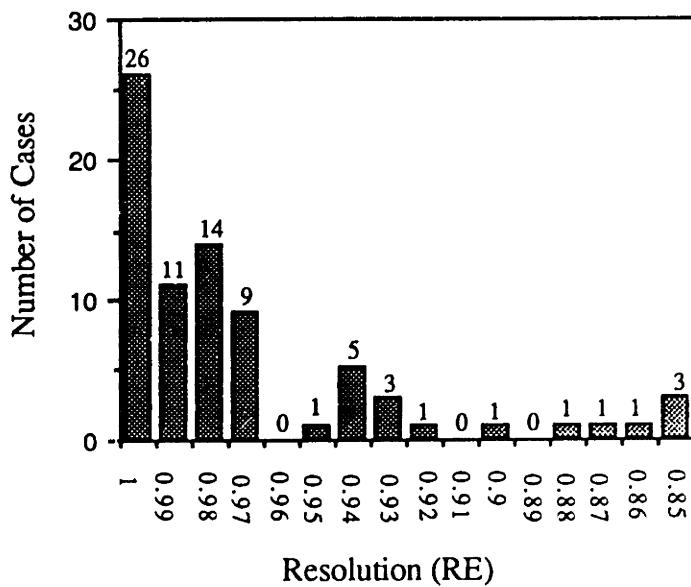


Fig. S.13. Resolution at End of Transient

Figure S.13 shows a distribution of the resolution achieved at the end of the transient. Based on all 77 cases, the average resolution was 0.97. This corresponds to an average candidate set size of 4 (i.e. 96% of the process malfunctions eliminated). Perfect resolution corresponds to a candidate set size of 1. Given the instrumentation scheme some malfunctions are qualitatively identical<sup>19</sup> and perfect resolution cannot always be achieved. In most cases where resolution was greater than or equal to 0.97, the best diagnosis given the available instrumentation was produced.

Almost all malfunction episodes in the study included at least one event that was not detected in its expected causal order. Other test cases demonstrated the performance of MIDAS in situations where multiple process malfunctions or external process transients occurred. The best diagnosis given the available instrumentation was produced in these situations.

In the final analysis, an accurate diagnosis was produced using the PM in most situations. The few cases where an inaccurate diagnosis was obtained were not due to errors in the

---

<sup>19</sup>For example pump blockage and CSTR outlet blockage.

PM, but due to limitations in the detection scheme of MIDAS. The PM correctly represents the underlying process behavior. Although perfect performance was not achieved, the level of performance achieved verifies the technique for developing the PM. The performance may be improved by methods suggested in Section S.4.4.

## S.4.4. Limitations

The acceptable diagnostic performance of MIDAS using simulated test cases has stimulated interest in applying MIDAS to industrial scale problems. Plans are underway to further evaluate MIDAS in actual or simulated process environments. However, a disappointing result that has become apparent during attempts in creating the PM of industrial scale processes is the "long" times required to create the PM. In order to construct the PM of large scale processes in reasonable time, more efficient algorithms are needed.

Furthermore, PMs derived by the current implementation of the algorithms are valid only when transitions to other qualitative regimes do not occur. Thus the criteria for identifying non-local causalities arising from controller saturation and other transitions across qualitative regimes should be implemented in the algorithms that derive the PM.

MIDAS ranked the true malfunction significantly lower than the top candidate in approximately 5% of the cases in the study of the jacketed reactor process. The major factors contributing to the suboptimal performance were:

- Apparent inverse response,
- Formation of causal loops during diagnostic inference, and
- Other fundamental limitations of qualitative models.

### **Apparent Inverse Response**

Apparent inverse response of a variable may be detected in a process even though inverse response of the variable is not a valid process behavior. This situation may arise in underdamped control loops where the controlled variable exhibits compensatory response (Fig. S.14). Inverse response does not indicate possible transient states that the process

may attain. Apparent inverse response occurs because transient states are used in diagnostic inference.

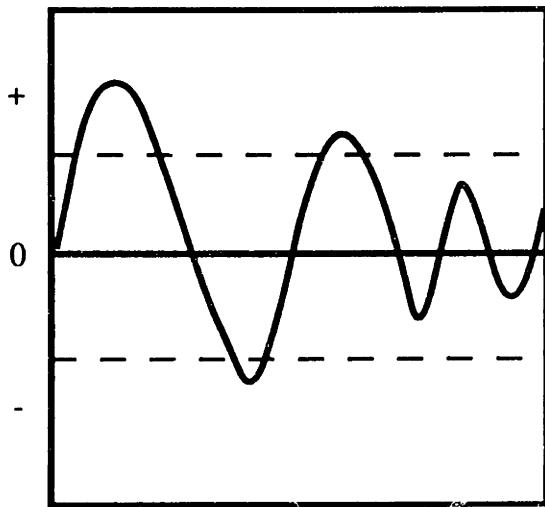


Fig. S.14. Apparent Inverse Response

During the case study apparent inverse response of the reactor temperature was observed. This resulted in an inaccurate diagnosis after the 7th event in 1 case and degraded diagnostic resolution in other cases.

One method of addressing this problem is to base the event transitions on non-stationary reference values. In underdamped systems, the amplitude of oscillation often decays by a constant ratio, D, during the transient. The value for the next transition point,  $x_e$ , may be based on the previous peak (trough),  $x_m$ , and the nominal steady state value  $x_s$ .

$$x_e = x_s \pm K \cdot (x_m - x_s) \cdot \sqrt{D} \quad ; \quad K\sqrt{D} < 1, \quad K > 1 \quad (S.11)$$

K is a constant that compensates for process non linearities. Another method would be to base event transitions on features of the trajectory shape as a whole.

## **Causal Loops**

A causal loop is a group of active causal links that connect observed events in a cycle during diagnostic inference [36]. These causal loops arise from negative feedback (control and non-control) or positive feedback loops in the process. The MIDAS inference algorithms have no mechanism for identifying source events in causal loops.

During the case study active causal loops were formed in the temperature control loop when apparent inverse response was exhibited by a variable in the loop. The inability to identify the source event in the loop led to the drop in diagnostic performance as all local causes of the source events were ranked highly.

One possible remedy is to restrict source events to be among the first event observed for each variable in the process. This "breaks" active causal loops arising from negative feedback loops, identifying a single source event. Breaking causal loops in this manner can be accomplished by modifying the inference algorithm. This would have resolved the temperature control causal loop that led to the drop in performance of MIDAS.

## **Fundamental limitations of qualitative modeling**

Other causes for suboptimal performance of MIDAS are due to fundamental limitations of qualitative modeling. These limitations arise from the inherent ambiguity in qualitative modeling and the point values and semi-infinite intervals associated with qualitative variables and relationships.

The primary effect of the ambiguity in qualitative modeling is the existence of causal links initiating from the same event and terminating at events whose consequent states indicate qualitatively opposite effects. The ambiguity results in a highly connected PM, consequently increasing the time required for diagnosis [36]. More importantly, the ambiguity leads to a loss in diagnostic resolution.

In the case study, ambiguity may have been better resolved by further use of process-specific quantitative information. The procedure for deriving the PM provides a means for specifying process specific information in the third step. Using this facility, 48.6 % of the potential ambiguities were resolved. The diagnostic resolution could have been improved by resolving a greater percentage of the ambiguities.

Another limitation of qualitative modeling arises from the point values and semi-infinite intervals associated with variables in qualitative algebra. The consequence of this is that links in the PM express **may-cause** relationships rather than **will-cause** relationships. Thus a **compiled-causal-link** can be interpreted as follows; the occurrence of the successor event may occur as a result of the precursor event. It does not imply that the successor event will necessarily occur at any finite time after the precursor event occurs.

This asymmetry limits the type of information that can be used in performing inference with the PM. The occurrence of an event may be used as a positive source of diagnostic information. However, no information is deduced from the non occurrence of an expected event. For example, in Fig. S.15, if **BS-normal-to-high** is observed after **AS-normal-to-high**,  $\beta$  may be eliminated from the diagnostic candidate set leaving  $\alpha$  as the only viable root cause. On the other hand, if **AS-normal-to-high** is observed and **BS-normal-to-high** is not observed, even after a very long time, no statement can be made about the relative likelihood of  $\alpha$  and  $\beta$ . While humans are capable of using both occurrences and non-occurrences of events in diagnostic inference, MIDAS is limited to using only event occurrences.

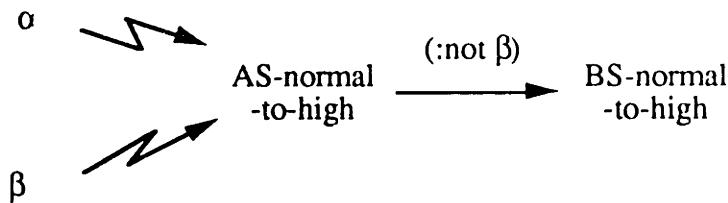


Fig. S.15. PM of a Hypothetical Process

In order to express **will-cause** relationships in the PM, knowledge of approximate gains and time delays associated with the causal links is required. Thus, if a subsequent event does not occur after a sufficient amount of time has elapsed, the non occurrence of the event may be used as a source of diagnostic information. The investigation of the use of semi-quantitative modeling approaches in deriving this information is suggested.

## S.5. Conclusions and Recommendations

This research has resulted in the development and implementation of a process independent, algorithmic technique for deriving the qualitative "core" of knowledge bases for an on-line diagnostic program (MIDAS) for continuous refinery and chemical process plants. Using the technique, the effort required in developing a knowledge base (PM) is largely reduced to specifying units from a model library according to the flowsheet structure and specifying the signs of certain variable and parameter relationships, such as relative temperatures in adjacent process units. The technique also significantly increases the reliability and reduces the time required to develop diagnostic knowledge bases, when compared to the process of interviewing process experts in rule based expert systems. Additionally, the expense of creating rigorous mathematical models and identifying process parameter values for these models is avoided.

A novel feature of the technique is that certain behaviors that are not apparent from local unit models, but result from global interactions in the process are automatically represented in the diagnostic knowledge base. Representing knowledge about these behaviors in the knowledge base, allows an accurate diagnosis to be obtained without unnecessary losses of diagnostic resolution.

Based on the limitations of the PM and the current implementation of algorithms for deriving the PM, possible avenues for future research are to:

- 1) Devise and implement more efficient algorithms for deriving the PM. Criteria for identifying non-local causalities arising from transitions to other qualitative regimes should be implemented.
- 2) Investigate the use of semi-quantitative modeling formalisms to express **will-cause** relationships rather than **may-cause** relationships in the PM.

Another logical extension of this thesis would be to:

- 3) Apply and extend the concepts developed in this thesis to the qualitative modeling of semi-continuous and batch processes.

The ability to derive process diagnostic knowledge bases for semi-continuous and batch processes will extend the range of applicability of MIDAS in industrial processes.

## S.6. References for Thesis Summary

1. Stanley, G.M. and R.S.H. Mah (1977). Estimation of flows and temperatures in process networks. A.I.Ch.E. J. 23, 642-650.
2. Park, S., and D.M. Himmelblau (1983). Fault detection and diagnosis via parameter estimation in lumped dynamic systems. Ind. Eng. Chem. Process Des. Dev. 22, 482-487.
3. Watanabe, K. and D.M. Himmelblau (1983). Fault diagnosis in nonlinear chemical processes - Part I. Theory. A.I.Ch.E. J. 29, 243-249.
4. Watanabe, K. and D.M. Himmelblau (1983). Fault diagnosis in nonlinear chemical processes - Part II. Application to a chemical reactor. A.I.Ch.E. J. 29, 250-261.
5. Lainiotis, D.G. (1971). Joint detection, estimation, and system identification. Inform. Control 19, 75-92.
6. Clark, R.N., D.C. Fosth and V.M. Walton (1975). Detecting instrument malfunctions in control systems. IEEE Trans. Aerospace Electronic Systems AES-11, no. 4, 465-473.
7. Willsky, A.S. (1976). A survey of design methods for failure detection in dynamic systems. Automatica 12, 601-611.
8. Isermann, R. (1984). Process fault detection based on modeling and estimation methods - A survey. Automatica 20, 387-404.
9. Shortliffe, E.H. (1976). MYCIN: Computer-based Medical Consultations. Elsevier, New York.

10. Dhujarti, P.S., D.E. Lamb and D. Chester (1987). Experience in the development of an expert system for fault diagnosis in a commercial scale process. Foundations of Computer Aided Process Operations. Park City, UT.
11. Sachs, P.A. and A.M. Patterson (1986). ESCORT - An expert system for complex operations in real time. Expert Systems, 3 (1), 22-28.
12. Fink, P.K. (1985). Control and integration of diverse knowledge in a diagnostic expert system. Proc. 9th IJCAI 426-431.
13. Kramer, M.A. (1986). Integration of heuristic and model-based inference in chemical process fault diagnosis. Proc. IFAC Workshop on Fault Detection and Safety in Chemical Plants, 111-115, Kyoto, Japan.
14. Kramer, M.A. (1988). Automated diagnosis of malfunctions based on object oriented programming. J. Loss Prev. Process Ind. 1, 226-232.
15. Moore, R.L. and M.A. Kramer (1986). Expert systems in on-line process control. Chemical Process Control CPC-III, 839-865, M. Morari and T.J. McAvoy (Eds), Elsevier.
16. de Kleer, J., and J.S. Brown (1984). A qualitative physics based on confluences. Artificial Intelligence 24, 7-83.
17. Pan, J.Y. (1984). Qualitative reasoning with deep-level mechanism models for diagnoses of mechanism failures. Proc. IEEE 1st Conf. on AI Appl. 295-301.
18. Kuipers, B. (1984). Common sense reasoning about causality: deriving behavior from structure. Artificial Intelligence, 24, 169-203.
19. Williams, B.C. (1984). Qualitative analysis of MOS Circuits. Artificial Intelligence 24, 281-346.
20. Iri, M., K. Aoki, E. O'Shima and H. Matsuyama (1979). An algorithm for diagnosis of system failures in the chemical process. Comput. & Chem. Eng. 3, 489-493.

21. Tsuge, Y., J. Shiozaki, H. Matsuyama and E. O'Shima (1985). Fault diagnosis algorithms based on the signed directed graph and its modifications. I. Chem. Eng. Symp. Ser. 92, 133-144.
22. Kokawa, M. and S. Shingai (1982). Failure propagating simulation and nonfailure paths search in network systems. Automatica 18, 335-341.
23. Palowitch, B.L. (1987). Fault Diagnosis of Process Plants Using Causal Models Sc.D. Thesis, Massachusetts Institute of Technology.
24. Iwasaki, Y. and H.A. Simon (1986). Causality in device behavior. Artificial Intelligence 29, 3-32.
25. Iwasaki, Y. and I. Bhandari (1988). Formal basis for commonsense abstraction of dynamic systems. Proc. 7th. Natl. Conf. on A.I., AAAI-88, 307-312.
26. Steward, D.V (1962). On an approach to techniques for the analysis of the structure of large systems of equations. Soc. Ind. Appl. Math. Rev. 4, 321-342.
27. Forbus, D.F. (1984). Qualitative process theory. Artificial Intelligence 24, 85-168.
28. Bobrow, D.G. (1985). Qualitative Reasoning about Physical Systems: An Introduction. In D. Bobrow (Ed.), Qualitative Reasoning about Physical Systems MIT Press, Cambridge, Massachusetts.
29. Kuipers, B.J. (1986). Qualitative Simulation. Artificial Intelligence 29, 289-338.
30. Oyeleye, O.O. and M.A. Kramer (1988). Qualitative simulation of chemical process systems: steady state analysis. A.I.Ch.E. J. 34, 1441-1454.
31. Kuipers, B.J. (1985). The limits of qualitative simulation. Proc 9th IJCAI 128-136.
32. O'Shima, E. (1983). Computer aided plant operation. Comput. & Chem. Eng. 6, 311-329.

33. Tsuge, Y., J. Shiozaki, H. Matsuyama, E. O'Shima, Y. Iguchi, M. Fuchigami and H. Matsushita (1985). feasibility study of a fault-diagnosis system for chemical plants. Int. Chem Eng. 25, 4, 660-667.
34. Ulerich, N.H. and G.J. Powers (1988). On-line hazard aversion and fault diagnosis in chemical processes: The digraph + fault-tree method. IEEE Trans. on Reliab. 37, 2, 171-177.
35. Kramer, M.A., F.E. Finch and O.O. Oyeleye (1989). Operating Manual and User's Guide for MIDAS (version 1.0a), Massachusetts Institute of Technology Laboratory for Intelligent Systems in Process Engineering.
36. Finch, F.E. (1989). Automated Fault Diagnosis of Chemical Process Plants using Model -Based Reasoning. Sc.D. Thesis, Massachusetts Institute of Technology.

# **1. Introduction: Qualitative Modeling and Applications to Malfunction Diagnosis**

## **1.1. Motivation**

In most continuous chemical process plants, supervisory control activities include the implementation of start-up, shut-down and change-over procedures, process monitoring and malfunction diagnosis. In the event of a process disturbance or malfunction, the operator's task is to diagnose the cause of the upset quickly and accurately so that corrective action may be taken. In modern integrated plants, strong dynamic interactions between process units may further complicate diagnosis because:

- Effects may propagate to process units located at great distances from the malfunction origin.
- Regulatory control and backup systems may obscure the symptoms of the malfunction.

Timely intervention by the operator is very important, because it may:

- Mitigate more serious incidents such as the release of toxic chemicals,
- Improve product quality, and
- Improve plant profitability by avoiding material and energy wastage due to spurious shutdowns.

In order to perform an accurate diagnosis, the operator must understand the behavior of the process. The operator's mental model of process behavior is usually developed by experience and not rigorous, and consequently the model may be inaccurate or limited to a "normal" regime of plant operation. In time-constrained situations, mental simulation of hypothetical situations can place significant cognitive load on the operator, reducing his or her effectiveness. Consequently, there are motivations to develop computer aids to assist the operator in diagnosis.

## 1.2. Process Modeling

Modeling and representing process behavior is a central issue in developing diagnostic aids. Interpreting process data requires a **process model** that satisfies the following requirements:

- Contains enough information to produce the correct diagnosis.
- Can be developed with reasonable engineering effort.
- Can be easily verified.

A brief review of the various types of models that may be used in developing diagnostic aids follows.

### *Quantitative Models*

Diagnostic methodologies based on dynamic quantitative models such as parameter estimation and filtering techniques [1, 2, 3 & 4], hypothesis testing of residual vectors on a bank of models [5 & 6], and comparative simulation, utilize a mathematical and statistical framework that allows considerable theoretical sophistication. Excellent reviews of diagnostic methods based on quantitative models are presented in [7 & 8]. These methods potentially produce the most accurate diagnoses. In addition, the models are explicit and well defined, therefore, directly accessible for verification. However, on the practical side, there are questions as to whether these techniques can be successfully applied to large processing plants because of:

- Restrictive assumptions [9],
- Excessive computational requirements,
- Considerable effort required to develop highly reliable and accurate models.

## *Rule Based Expert Systems*

Rule based expert systems are symbolic reasoning programs, developed in the field of Artificial Intelligence. These programs attempt to model the diagnostic knowledge of experienced process engineers and operators as a set of situation-action production rules. The rules typically specify a heuristic mapping between patterns of abnormal symptoms such as high and low states and specific process malfunctions. Rule based systems have been successfully applied to diagnosis in domains where there is limited fundamental understanding; for example in medical diagnosis, MYCIN [10] and INTERNIST [11]. In the chemical process industry, rule based diagnostic systems have been applied, achieving varying degrees of success. Notable examples are the FALCON project [12] and the ESCORT system [13]. Other applications are reported in [14, 15 & 16].

Several factors limit the applicability of rule based systems [17, 18 & 19] to chemical process diagnosis, where the fundamental principles are well understood.

- The rules are derived from experience rather than domain principles, and the systems often fail when solving an unfamiliar problem. The experience required could be totally or partially missing in a new plant.
- The procedure used in developing a set of rules for an application is unstructured and often ad hoc. As a result the rule set may be unreliable or incomplete and cannot be easily verified.
- The rules suffer from a lack of generality and each process requires development of unique rule base. This is a critical problem because upwards of 10,000 rules may be required in a large plant if generic rules are not used [20].

## *Qualitative Logic Diagrams*

Fault trees, cause-consequence diagrams and event trees are logic diagrams in which a tree is developed in terms of a predefined event and its logical relationships with cause and/or consequent events [21]. Process variables, which are intrinsically continuous, are represented by a limited set of discrete qualitative states. These diagrams have been used

extensively for failure and safety analysis during the design of nuclear plants [22]. The diagrams are oriented towards multiple root causes where the combined effects lead to a hazard. An application of the fault tree methodology to the reliability and hazards analysis of a cumene hydroperoxide plant is reported in [23].

The application of these diagrams to the on-line diagnosis of process plants has been hindered because:

- Extensive effort is required in development by a team of experienced engineers. This may take up to 25 person-years to develop if constructed manually [24]. Developing these diagrams using automatic generation techniques based on unit operation models has proven difficult [25].
- The tightly integrated logic represented in the diagrams make them difficult to verify and maintain.
- It is difficult to represent process dynamic behavior and to resolve ambiguities arising from negative feedback and feedforward loops. Representing dynamic behavior and resolving these ambiguities is necessary for a "good" diagnosis.

### *Qualitative Models*

Other symbolic diagnostic programs are based on "formal" qualitative models of process behavior. The models are derived from the fundamental principles governing the domain and can easily represent information about the structural connectivity of the process. In their most general form, these models can be viewed as abstractions of the conventional differential/algebraic equation models in which some or all quantitative information is omitted. The models explicitly represent the relationship between discrete qualitative states of process variables.

Qualitative models can be further classified as qualitative constraint models and qualitative causal models. The former include the modeling formalisms of de Kleer and Brown [26], Pan [27], Kuipers [28] and Williams [29]. All these formalisms formulate the model in terms of constraints relating discrete qualitative states of process variables, differing primarily in the way the constraints are derived. Constraint models have been used for diagnostic and other applications in several domains.

An important class of qualitative causal models is based on the Signed Directed Graph (SDG) and its related diagrams [30, 31, 32 & 33]. These diagrams consist of nodes symbolizing process variables, and signed directed arcs that represent the local cause and effect relationships between variables. Other representations of causality used in reasoning about process behavior are those based on the formalisms of Iwasaki and Simon [34 & 35] in which causality is construed as the output set assignment [36] of a set of simultaneous algebraic and differential equations and the Qualitative Process Theory of Forbus [37].

The major attraction of using formal qualitative models over other modeling formalisms for diagnostic applications is based on the following factors:

- They contain much of the necessary information useful for diagnosis and can be constructed with less effort than quantitative models.
- Most of the information required in developing qualitative models is general and not process specific.
- The computational effort in the application phase is reduced.
- Reasoning with qualitative models is similar to that of humans and thus readily understandable and explainable.
- Relative to rule-based expert systems and qualitative logic diagrams, qualitative modeling provides a more structured approach for developing diagnostic knowledge bases.

On the other hand, qualitative modeling is a fundamentally underspecified problem [38]. Previous methods of qualitative modeling have tended to generate predictions that do not coincide with observable behaviors, either by including spurious behaviors or excluding

actual behaviors [39]. There is a duality between modeling and diagnosis, thus, in diagnostic applications, these limitations could lead to:

- Unnecessary losses of diagnostic resolution
- Inaccurate diagnosis.

In addition, qualitative modeling does not utilize available process-specific numerical information which might prove useful in improving diagnostic resolution.

### 1.3. Thesis Objectives

The above limitations of qualitative modeling are crucial and have in the past reduced the potential application of diagnostic systems based on these models. The objectives of this thesis are:

- 1) Understand the limitations of qualitative modeling at a fundamental level. Investigate steady state qualitative modeling, identifying the elements required to reduce the inherent ambiguity.
- 2) Investigate the limitations of previous disturbance propagation assumptions on modeling process behavior arising from interactions in feedback loops of the SDG.
- 3) Develop a representation of process causality that produces all valid process behaviors while minimizing spurious behaviors.
- 4) Develop a structured approach and implement procedures and algorithms for automatically deriving diagnostic knowledge bases for continuous processes from their constituent units and flowsheet topographies. The procedures should produce knowledge bases that have the following properties:
  - a) Can be easily verified for consistency and completeness.
  - b) Provide accurate diagnoses in processes with strong dynamic interactions without unnecessary losses in diagnostic resolution.

- c) Can provide a diagnosis using the existing process instrumentation scheme.
- d) Can be developed with incomplete information about process-specific parameter values. Available information about parameter values could be used to refine the diagnosis.

## 1.4. Thesis Outline

The objectives of the thesis have been met, ultimately leading to the development and implementation of a structured approach for deriving the qualitative behavior of a process from the flowsheet. The process model so derived serves as the "core" of an on-line diagnostic reasoning program for continuous refinery and chemical process plants. The approach significantly increases the reliability, and reduces the time and effort required to develop diagnostic knowledge bases, when compared to the process of interviewing process experts in rule based expert systems. Additionally, the expense in creating rigorous mathematical models and identifying process parameter values for these models is avoided.

One significant feature of the model is that certain behaviors not apparent from local unit models are automatically represented in the model. These behaviors result from global interactions in the flowsheet and are not satisfactorily addressed in other qualitative model-based diagnostic systems. Examples of these systems include AQUA [40], DIEX [33], MODEX [41] and YAKA [42].

This thesis consists of ten chapters. Following the introduction, the steady state qualitative modeling problem is investigated. The original publications appear as [39 & 43]. It is shown that both causal and non-causal constraints are required to predict the steady state measurement patterns generated as a result of process malfunctions with a "minimum" number of spurious patterns. In addition, it's necessary to generate additional local and global constraints associated with analytically redundant model equations. The utility of multiple sources of constraints is demonstrated with an example of a chemical reactor and its associated heat exchange and control systems, in which the number of spurious solutions is reduced by up to two orders of magnitude.

The Extended Signed Directed Graph (ESDG) is developed in Chapter 3. The ESDG is a representation of causality based on the SDG, that produces all process behaviors while minimizing spurious behaviors. An important limitation of the SDG, is that certain behaviors not apparent from the local causal models, result from global interactions in the flowsheet. Rigorous theorems for anticipating all non-local causalities are derived, the key being the location of positive feedback and integrating effects in the system. The original derivation of these theorems appear in [39]. By anticipating all non-local causalities, we can assure the inclusion of the correct failure origin without degrading diagnostic resolution in diagnostic systems based on the ESDG.

In Chapter 4, guidelines for deriving SDG models for process units from conventional differential and algebraic equation models are presented. These models are used in conjunction with the theorems developed in Chapter 3, to derive the ESDG for processes. The guidelines offer a significant improvement over those presented in [33]. In addition to a more rigorous treatment of the derivation of causal arcs from algebraic equations, rules for specifying integrating and positive feedback effects associated with nodes in the SDG are presented.

The Model Integrated Diagnostic Analysis System (MIDAS) is an on-line diagnostic aid for continuous refinery and chemical processes [44]. MIDAS is primarily based on a qualitative model derived from the ESDG but can also utilize constraint information to improve diagnostic resolution. The system is designed to maintain a separation between the diagnostic model of the process and its inference strategy. In Chapter 5, the basic features and structure of MIDAS are reviewed. The presentation parallels the developments in [45 & 46].

In Chapter 6, a structured procedure for developing the qualitative diagnostic model of the process used in MIDAS is presented. The procedure significantly increases the reliability and reduces the time and effort required in developing diagnostic knowledge bases. With the library of unit SDG models provided in the MIDAS package, the user's task is reduced to answering simple questions, such as the relative temperatures between process streams. The algorithm consists of four stages. After SDG unit models are instantiated and linked according to the flowsheet topography, the topology is analyzed to anticipate all non-local causalities, deducing the behavior of the process. The resulting ESDG is translated to the MIDAS diagnostic model by eliminating unmeasured variables. Guidelines for deriving the relationships between violated quantitative constraint relations and malfunctions are also

presented. Suggestions about ways to improve the efficiency of the algorithms used for deriving the diagnostic model are made. The original publication appears as [47].

In Chapter 7, a study of the performance characteristics of MIDAS is presented in the context of an example process. The process considered is a jacketed continuous stirred tank reactor with its associated heat exchange and control systems. Process disturbances and malfunctions considered include those directly affecting process units and malfunctions in the measurement and control systems. Statistics on the diagnostic power and accuracy obtained using MIDAS are presented. Major limitations of the basic modeling formalism are also highlighted. The material presented parallels the developments in [48 & 49].

In Chapter 8, the major contributions of this research are summarized. Directions for future research are indicated.

The final two chapters contain sections of the MIDAS Operating Manual and Users Guide [44]. Detailed instructions about how to develop the process diagnostic model from a flowsheet using MIDAS utility programs are provided. Chapter 9 presents instructions for building the process SDG from unit SDG models. The user selects unit models from a library of unit models provided in the MIDAS package and connects them according to the unit topography. For units not included in the model library, a facility is provided to input batch files (according to a specialized input format) containing the SDG models. Chapter 10 presents instructions for translating the process SDG to the MIDAS-compatible qualitative diagnostic knowledge base. A utility for entering relationships between malfunctions and violated quantitative constraints in MIDAS is provided.

## 1.5. References for Chapter 1

1. Stanley, G.M. and R.S.H. Mah (1977). Estimation of flows and temperatures in process networks. A.I.Ch.E. J. 23, 642-650.
2. Park, S., and D.M. Himmelblau (1983). Fault detection and diagnosis via parameter estimation in lumped dynamic systems. Ind. Eng. Chem. Process Des. Dev. 22, 482-487.

3. Watanabe, K. and D.M. Himmelblau (1983). Fault diagnosis in nonlinear chemical processes - Part I. Theory. A.I.Ch.E. J. 29, 243-249.
4. Watanabe, K. and D.M. Himmelblau (1983). Fault diagnosis in nonlinear chemical processes - Part II. Application to a chemical reactor. A.I.Ch.E. J. 29, 250-261.
5. Lainiotis, D.G. (1971). Joint detection, estimation, and system identification. Inform. Control 19, 75-92.
6. Clark, R.N., D.C. Fosth and V.M. Walton (1975). Detecting instrument malfunctions in control systems. IEEE Trans. Aerospace Electronic Systems AES-11, no. 4, 465-473.
7. Willsky, A.S. (1976). A survey of design methods for failure detection in dynamic systems. Automatica 12, 601-611.
8. Isermann, R. (1984). Process fault detection based on modeling and estimation methods - A survey. Automatica 20, 387-404.
9. Himmelblau, D.M. (1985). Material balance rectification via interval arithmetic. I. Chem. Eng. Symp. Ser. 92, 121-132.
10. Shortliffe, E.H. (1976). MYCIN: Computer-based Medical Consultations. Elsevier, New York.
11. Pople, E.H. Jr. (1982). Heuristic methods for imposing structure on ill-structured problems: the structuring of medical diagnostics. In P. Szolovits (Ed.) Artificial Intelligence in Medicine. Westview Press, Boulder, CO.
12. Dhujarti, P.S., D.E. Lamb and D. Chester (1987). Experience in the development of an expert system for fault diagnosis in a commercial scale process. Foundations of Computer Aided Process Operations. Park City, UT.
13. Sachs, P.A. and A.M. Patterson (1986). ESCORT - An expert system for complex operations in real time. Expert Systems, 3, 22-28.

14. Taunton, C. (1986). Expert systems go on-line at Blue Circle. Sensor Review April, 77-79.
15. Newquist, H.P. (1987). True stories: the reality of AI applications. AI Expert 2, no. 2, 63.
16. Harmon, P. (1986). Inventory and analysis of existing expert systems. Expert System Strategies 2, no. 8, 1.
17. Fink, P.K. (1985). Control and integration of diverse knowledge in a diagnostic expert system. Proc. 9th IJCAI 426-431.
18. Kramer, M.A. (1986). Integration of heuristic and model-based inference in chemical process fault diagnosis. Proc. IFAC Workshop on Fault Detection and Safety in Chemical Plants, 111-115, Kyoto, Japan.
19. Kramer, M.A. (1988). Automated diagnosis of malfunctions based on object oriented programming. J. Loss Prev. Process Ind. 1, 226-232.
20. Moore, R.L. and M.A. Kramer (1986). Expert systems in on-line process control. Chemical Process Control CPC-III, 839-865, M. Morari and T.J. McAvoy (Eds), Elsevier.
21. Andow, P.K., F.P. Lees and C.P. Murphy (1980). The propagation of faults in process plants: a state of the art review. I. Chem. E. Symp. Ser. 58, 225-243.
22. Lapp, S. and G. Powers (1977). Computer-assisted generation and analysis of fault trees. 2nd Intl. Symp. Loss Prevention and Safety Promotion in the Process Industries 377-384, Heidelberg, West Germany.
23. Ardent, J.S., M.L. Casada and J.J. Rooney (1986). Reliability and hazards analysis of a cumene hydroperoxide plant. Plant/Operations Progress 5, April 97-102.

24. Anon. (1975). Reactor Safety Study--An Assessment of Accident Risks in U.S. commercial Nuclear Power Plants. Report of the NRC, NUREG-75/014, Washington, D.C.
25. Martin-Solis, G.A., P.K. Andow and F.P. Lees (1982). Fault tree synthesis for design and real time applications. Trans. Instn. Chem. Eng. 60, 14-25.
26. de Kleer, J., and J.S. Brown (1984). A qualitative physics based on confluences. Artificial Intelligence 24, 7-83.
27. Pan, J.Y. (1984). Qualitative reasoning with deep-level mechanism models for diagnoses of mechanism failures. Proc. IEEE 1st Conf. on AI Appl. 295-301.
28. Kuipers, B. (1984). Common sense reasoning about causality: deriving behavior from structure. Artificial Intelligence, 24, 169-203.
29. Williams, B.C. (1984). Qualitative analysis of MOS Circuits. Artificial Intelligence 24, 281-346.
30. Iri, M., K. Aoki, E. O'Shima and H. Matsuyama (1979). An algorithm for diagnosis of system failures in the chemical process. Comput. & Chem. Eng. 3, 489-493.
31. Tsuge, Y., J. Shiozaki, H. Matsuyama and E. O'Shima (1985). Fault diagnosis algorithms based on the signed directed graph and its modifications. I. Chem. Eng. Symp. Ser. 92, 133-144.
32. Kokawa, M. and S. Shingai (1982). Failure propagating simulation and nonfailure paths search in network systems. Automatica 18, 335-341.
33. Palowitch, B.L. (1987). Fault Diagnosis of Process Plants Using Causal Models. Sc.D. Thesis, Massachusetts Institute of Technology.
34. Iwasaki, Y. and H.A. Simon (1986). Causality in device behavior. Artificial Intelligence 29, 3-32.

35. Iwasaki, Y. and I. Bhandari (1988). Formal basis for commonsense abstraction of dynamic systems. Proc. 7th. Natl. Conf. on A.I., AAAI-88, 307-312.
36. Steward, D.V (1962). On an approach to techniques for the analysis of the structure of large systems of equations. Soc. Ind. Appl. Math. Rev. 4, 321-342.
37. Forbus, D.F. (1984). Qualitative process theory. Artificial Intelligence 24, 85-168.
38. Kuipers, B.J. (1986). Qualitative Simulation. Artificial Intelligence 29, 289-338.
39. Oyeleye, O.O. and M.A. Kramer (1988). Qualitative simulation of chemical process systems: steady state analysis. A.I.Ch.E. J. 34, 1441-1454.
40. Ishida, Y. and L. Eshelman (1987). Integrating model-based and syndrome-based diagnosis AQUA: A qualitative approach to process diagnosis. Carnegie Mellon University, Department of Computer Science Technical Report CMU-CS-87-111.
41. Rich, S.H. and V. Venkatsubramanian (1987). Model-based reasoning in diagnostic expert systems for chemical process plants. Comput. Chem. Eng. 11, 111-122.
42. Lambert, H., L. Eshelman and Y. Iwasaki (1988). Using qualitative physics to guide the acquisition of diagnostic knowledge. Proc 3rd Intl. Conf. on Appl. A.I. in Eng.
43. Oyeleye, O.O. and M.A. Kramer (1989). The role of causal and non-causal constraints in steady state qualitative modeling. in Artificial Intelligence Simulation and Modeling, 327-358, L.E. Widman, K. Loparo and N. Nielson (Eds). John Wiley and Sons.
44. Kramer, M.A., F.E. Finch and O.O. Oyeleye (1989). Operating Manual and User's Guide for MIDAS (version 1.0a), Massachusetts Institute of Technology Laboratory for Intelligent Systems in Process Engineering.

45. Finch, F.E. and M.A. Kramer (1989). The handling of dynamics, multiple faults, and out-of-order alarms in the MIDAS diagnosis system. A.I.Ch.E. Spring National Meeting, Houston, TX.
46. Finch, F. E. and M.A. Kramer (1989). A new diagnostic algorithm for dynamic processes. In preparation.
47. Oyeleye, O.O. and M.A. Kramer (1989). Modeling aspects of the MIDAS diagnostic system. In preparation.
48. Finch, F.E., O.O. Oyeleye and M.A. Kramer (1989). A case study of process diagnosis using MIDAS. In preparation.
49. Oyeleye, O.O., F.E. Finch and M.A. Kramer (1989). Qualitative modeling and fault diagnosis of dynamic processes by MIDAS. A.I.Ch. E. National Meeting, San Francisco, CA.

## **2. Steady State Qualitative Modeling**

Previous methods of qualitative modeling have formulated models of continuous systems in terms of qualitative constraints derived from system structure. The models produced by these methods are underspecified, leading to multiple solutions (interpretations) of system behavior. In this chapter, a systematic method of reducing the number of spurious interpretations is presented.

The role of causal and non-causal constraints in steady state qualitative modeling are highlighted. One of the major concepts resulting from this study is that information that can only be deduced from the global topology of the system, is necessary in steady state qualitative modeling. Additionally, it is shown that both non-causal and causal constraints are required to model the steady state qualitative behavior of continuous systems with a "minimum" number of spurious interpretations. The utility of multiple sources of constraints is demonstrated with an example of fault simulation of a chemical reactor and its associated heat exchange and control systems, in which the number of spurious solutions is reduced by up to two orders of magnitude.

### **2.1. Review of Previous Work on Qualitative Constraint Modeling**

The traditional cornerstones of physical systems modeling are mathematical conservation laws, constitutive relationships, and correlations. Numerical solutions to these equations have significant predictive value and may be interpreted to recognize important events or provide an explanation for the behavior of the system [1]. Human cognition, however, operates at a level of abstraction where qualitative reasoning supplements or supplants numerical processing. Within certain limits, humans employing "physical insight", scientific principles, and/or experience can predict the qualitative behavior of physical systems without resort to detailed numerical analysis.

The current generation of expert systems that emulate the problem solving ability of humans generally do not exploit the underlying principles of the problem domain. Consequently, these systems often fail when confronted with an unfamiliar problem [2]. This has stimulated research on formal methods of qualitative modeling [3]. Where

qualitative models suffice, there may be great economies in avoiding theoretical developments, exhaustive collection of data, and the long computer times associated with conventional numerical simulation. However, qualitative reasoning is fundamentally underspecified and prone to production of multiple solutions. The primary objective of this chapter is to address the question of multiplicity in qualitative modeling, and to identify the elements of steady state qualitative models that are necessary to minimize spurious interpretations.

The systems of primary interest to us are continuous chemical processes, characterized by continuous inflows and outflows of mass and energy, with state variables that are continuous real-valued functions of time. Often these process are designed to operate at steady state, i.e. the process state variables are time invariant. Qualitative modeling in this context entails predicting qualitative changes that may occur in continuous systems due to external disturbances or malfunctions. When a continuous system at steady state is perturbed, state variables change during the transient and ultimately may attain a new steady state. In the current chapter, we do not attempt to predict the process behavior during the transient; we focus only on the ultimate (long time) response of processes to malfunctions. The qualitative steady state predictions are useful in malfunction diagnosis, and other applications may be found in areas such as process control, planning, and design analysis.

Previously, several qualitative modeling methods for continuous systems have been proposed, including qualitative physics methods based on confluences [4], deep-level mechanism models [5], qualitative differential equations [1,6], and Qualitative Process Theory [7]. All of these methods formulate qualitative models in terms of constraints derived from the physical structure of the system, differing primarily in the way the constraints are derived. In this chapter, constraints are derived from the conventional differential and algebraic equation description of the process. Qualitative behavior is determined by constraint satisfaction. The state description produced by the constraints is the ultimate direction of change of variables from the initial steady state, represented as (+, 0, -), corresponding to higher than, equal to, or lower than the initial steady state value.

The loss of numerical information in qualitative modeling potentially leads to multiple solutions of system behavior [4]. Some of these interpretations may be real and valid for some member of the qualitative class of the system while others may be spurious and not exhibited by any quantitative realization of the system's qualitative class. (Systems that differ only in quantitative aspects, but have an identical qualitative representations are

considered members of the same qualitative class). Ideally, interpretations produced by a qualitative model should include interpretations for all members of the system's qualitative class (completeness) and be realized by a quantitative realization of the class (realizability). Kuipers [8], has proven that qualitative models in general cannot be guaranteed to exclude spurious interpretations.

Therefore, a major challenge in qualitative modeling is elimination of spurious interpretations. We extend the previous methods of qualitative modeling by providing a systematic method of reducing the number of spurious solutions produced by these representations without resorting to quantitative information. In the following, we highlight the role of non-causal and causal constraints in Sections 2.2 and 2.3, respectively. An example of qualitative simulation of a complex chemical reactor system is presented in Section 2.4. Although our examples are drawn from chemical engineering, the elements of our theory are domain independent, and can be applied to qualitative modeling of continuous systems in other domains.

## 2.2. Simulation of Steady States by "Pure" Confluences

Quantitative mathematical models of continuous systems consist of sets of simultaneous algebraic and differential equations. In the steady state limit, these models can be reduced to algebraic equation models by eliminating time derivatives and by discretization or lumping of spatial dimensions. Disturbances and malfunctions can be represented by parametric perturbation of the steady state equations. In this section, we show how the algebraic equations describing the perturbed steady state provide qualitative constraints on the ultimate direction of change of system variables.

### 2.2.1. Local Steady State Constraints

The quantitative model describing a system at steady state can be expressed as a set of n independent non-linear equations

$$f(x, u, p) = 0 ; \quad u = u_0 , \quad p = p_0 \quad (2.1)$$

where  $\underline{x}$  is a vector of  $n$  independent state variables,  $\underline{u}$  and  $\underline{p}$  are vectors of externally specified variables (inputs) and system parameters, respectively. Because of the possibility of algebraic manipulation, this set of equations is not unique. However, in formulating the model, the modeler usually writes equations that refer to particular units or subsystems so that each equation is **local** and does not involve variables in non-adjacent subsystems.

For a finite perturbation from the initial steady state, the change in the system can be expressed as:

$$\int \partial f / \partial \underline{x} \cdot d\underline{x} + \int \partial f / \partial \underline{u} \cdot d\underline{u} + \int \partial f / \partial \underline{p} \cdot d\underline{p} = 0 \quad (2.2)$$

where the lower and upper limits of integration are the initial and final steady states, respectively. Applying the mean value theorem, eq. (2.2) becomes:

$$\partial f / \partial \underline{x} \cdot \Delta \underline{x} + \partial f / \partial \underline{u} \cdot \Delta \underline{u} + \partial f / \partial \underline{p} \cdot \Delta \underline{p} = 0 \quad (2.3)$$

where  $\partial f / \partial \underline{i}$  is the mean value of the partial derivative along the integration path. Steady state confluences can be derived by qualitative translation of eq. (2.3):

$$[\partial f / \partial \underline{x} \cdot \Delta \underline{x}] + [\partial f / \partial \underline{u} \cdot \Delta \underline{u}] + [\partial f / \partial \underline{p} \cdot \Delta \underline{p}] = 0 \quad (2.4a)$$

The square brackets, [ ], represents the qualitative value (sign) of the argument. The operations of qualitative algebra are defined in Table 2.1. Note that addition of quantities of opposite sign results in ambiguity, since relative magnitudes are not known. Under the rules of qualitative algebra, eq. (2.4a) is equivalent to:

$$[\partial f / \partial \underline{x}] \cdot [\Delta \underline{x}] + [\partial f / \partial \underline{u}] \cdot [\Delta \underline{u}] + [\partial f / \partial \underline{p}] \cdot [\Delta \underline{p}] = 0 \quad (2.4b)$$

In order to maintain consistency with the notation of de Kleer & Brown [4], eq. (2.4b) is rewritten as

$$[\partial f / \partial \underline{x}] \cdot [d\underline{x}] + [\partial f / \partial \underline{u}] \cdot [d\underline{u}] + [\partial f / \partial \underline{p}] \cdot [d\underline{p}] = 0 \quad (2.4c)$$

where the differential,  $d$ , refers to a total macroscopic rather than an infinitesimal change in a quantity.

Table 2.1 Table of Qualitative Operations

$[dX] + [dY]$

$[dY]$	$[dX]$	+	0	-	?
+		+	+	?	?
0		+	0	-	?
-		?	-	-	?
?		?	?	?	?

$[dX] - [dY]$

$[dY]$	$[dX]$	+	0	-	?
+		?	-	-	?
0		+	0	-	?
-		+	+	?	?
?		?	?	?	?

$[dX] \cdot [dY]$

$[dY]$	$[dX]$	+	0	-	?
+		+	0	-	?
0		0	0	0	0
-		-	0	+	?
?		?	0	?	?

$[dZ] = d([dX])$

$[dX]$	+	0	-	?
$[dZ]$	0	+	0	+ or 0

Equation (2.4c) is a set of equations that are sums of products of parameter/variable relationships (partial derivatives) and differential quantities, also known as "mixed" confluences [4]. These can be converted to a set of "pure" confluences (a simple sum of differential quantities) by assigning fixed signs to the partial derivatives. This is done by defining a set of inequality constraints  $\underline{C}$  under which the partial derivatives in eq. (2.2) possess the same signs as at the initial state. Usually, the signs of the partial derivatives are determined from ordinal relationships. For example, state variable and physical parameter values are usually positive, the heating stream temperature is normally higher than the cooling stream temperature at points along a heat exchanger. Within  $\underline{C}$ , the average values of the derivatives along the path of integration must have the same sign as the partial

derivative at the initial state. The ranges of state variables and parameter values which satisfy these inequalities define the range of the validity of the "pure" confluences, i.e., the scope of the system's qualitative class.

During a transient, the inequality conditions  $C$  defining the qualitative class can be violated, leading to a transition between qualitative regimes. A separate set of confluences is required to characterize the behavior of the system for each qualitative regime. Typical qualitative transitions are phase changes, flow reversals, and controller saturation. These violations occur when either:

- The conditions under which the process equation from which the confluence was derived no longer holds, e.g. phase changes and controller saturation.
- The coefficient of one or more of the differential quantities in eq. (2.4c) change, e.g. flow reversals.

The problem of identifying potential class transitions has been addressed in [1, 4, 7 & 9]. At best these analyses provide a partial ordering but do not determine which transitions actually occur, since this determination usually requires numerical information. In this chapter, it will be assumed that the only allowable transitions between qualitative regimes is the saturation of control loops.

The functional relationships in eq. (2.1) do not have to be completely specified in order to derive confluences. For example, the equation which describes the flow of liquid for all classes of control valves is:

$$F = f^+(V, |P_I - P_O|) \cdot [(P_I - P_O)] \quad (2.5)$$

where  $f^+$  is an unspecified monotonically increasing function of the valve stem position,  $V$ , and the absolute pressure difference,  $|P_I - P_O|$ . For forward flow, ( $P_I > P_O$ ), the confluence derived from this equation, independent of the exact nature of  $f^+$ , is

$$[dF] = [dP_I] - [dP_O] + [dV] \quad (2.6)$$

Equation (2.6), states that a positive change in flowrate cannot occur without either a positive change in inlet pressure or valve stem position, or a negative change in outlet pressure. This confluence is a logical constraint on the behavior of the valve, and not an indication of causality. Causality cannot be construed from this equation, as increased flow would not cause opening of the valve, but could result in increased downstream pressure.

## 2.2.2. Constraint Satisfaction Algorithm

The set of conflences in eq. (2.4c) forms a set of inherently simultaneous qualitative equations. The strategy employed to find all possible solutions to the set of conflences is a combination of constraint satisfaction and generate and test. All qualitative patterns of variables that satisfy every confluence in the set are feasible solutions. In testing a confluence, appearance of the ambiguous value, (?), automatically satisfies the confluence.

A flowchart for the constraint satisfaction algorithm is shown in Figure 2.1. The algorithm is a modified version of the constraint satisfaction technique presented in [10]. The algorithm is implemented in Zetalisp on a Symbolics 3640 computer. Although the algorithm can be implemented in any language supporting iteration, it is most easily implemented in languages supporting recursion.

Constraint satisfaction is an NP complete problem [11]. Therefore, to improve the efficiency of the basic algorithm [10], a heuristic for selecting the order in which variables are solved is adopted. At each stage of constraint satisfaction, the values of the variables are determined in the order such that selecting the fewest variables leads to the application of the largest number of conflences. This heuristic results in fewer partial interpretations at each stage.

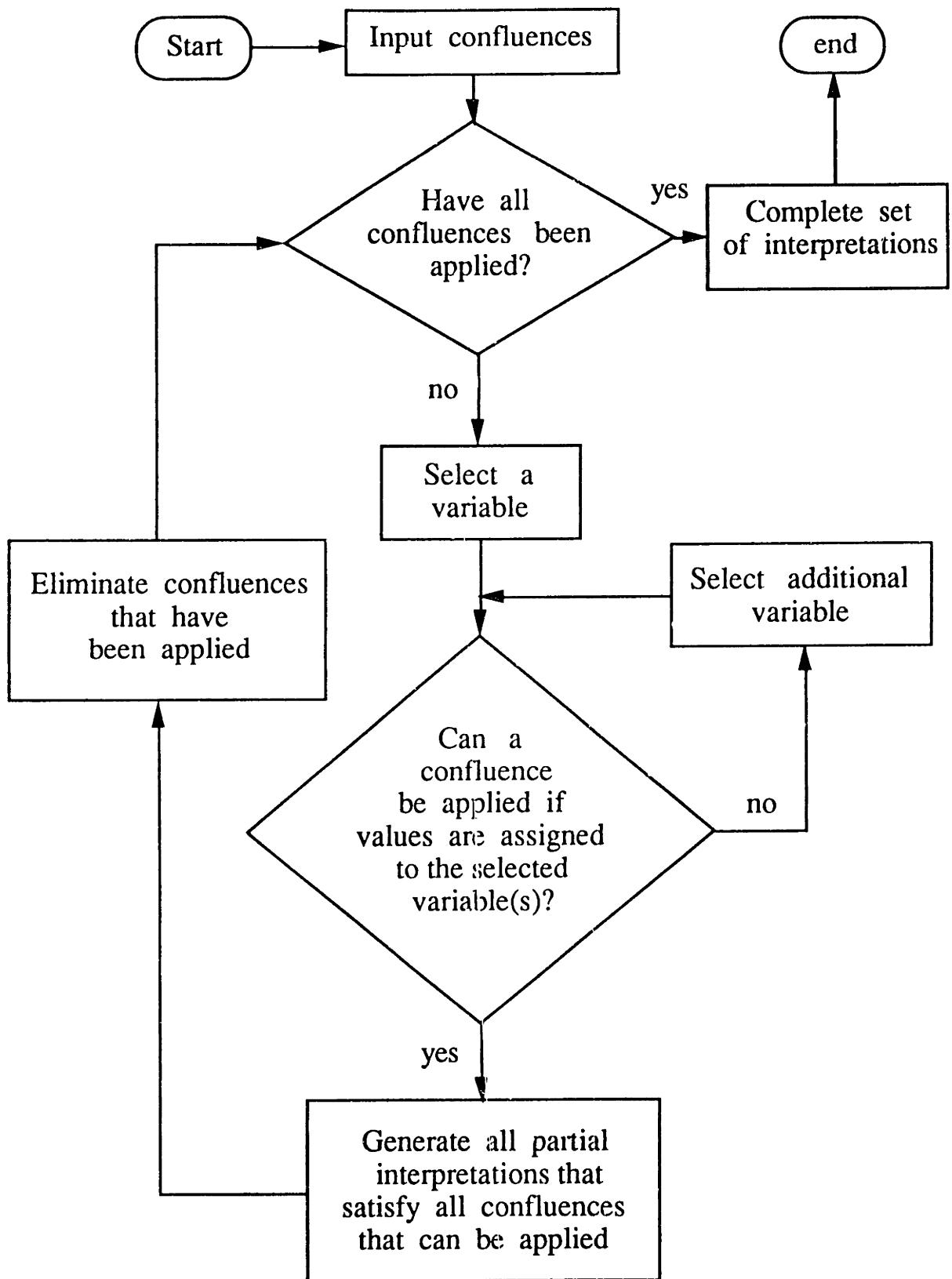


Fig. 2.1. Flowchart of Constraint Satisfaction Algorithm

The following example illustrates the order in which the confluences are applied.

*Example*

Consider a system of confluences  $e_i(x_j)$  where each confluence  $e_i$  is a qualitative expression in the variables  $x_j$ :

$$\begin{array}{ll} e_1(x_1, x_2) & e_4(x_1, x_3, x_4, x_5, x_6) \\ e_2(x_2, x_3, x_5) & e_5(x_3, x_4, x_5, x_6) \\ e_3(x_3, x_4) & e_6(x_3, x_5) \end{array}$$

A confluence can only be applied if qualitative values have been assigned to all variables appearing in the confluence. The algorithm selects the variable whose value, if known, would result in the largest number of confluences with the fewest unassigned variables.

Step 1. No assignment of values to any single variable would lead to the application of any of the confluences. However if values are assigned to  $x_3$ , 2 confluences ( $e_3$  &  $e_6$ ) will have the value of 1 variable unassigned. If values are assigned to any of  $x_1$ ,  $x_2$ ,  $x_4$  or  $x_5$ , 1 confluence will have the value of 1 variable unassigned; and if values assigned to  $x_6$ , 3 confluences will have the value of 2 variables unassigned. Select  $x_3$ .

Step 2. Assigning values to either of  $x_4$  or  $x_5$  now would lead to the application of  $e_3$  or  $e_6$ , respectively. If  $x_5$  is chosen, 2 confluences will have the value of 1 variable unassigned, and if  $x_4$  is chosen, 1 confluence will have the value of 1 variable unassigned. Thus select  $x_5$ .

Step 3. Find all combinations of values (partial interpretations) of  $x_3$  and  $x_5$  which satisfy  $e_6$  by trial and error (generate and test). Eliminate  $e_6$  from the set of confluences, and tag  $x_3$  and  $x_5$  as variables whose values have been assigned. At this stage (using  $y_j$  to indicate known variables), the following confluences remain, :

$$\begin{array}{ll} e_1(x_1, x_2) & e_4(x_1, y_3, x_4, y_5, x_6) \\ e_2(x_2, y_3, y_5) & e_5(y_3, x_4, y_5, x_6) \\ e_3(y_3, x_4) & \end{array}$$

Step 4. Assigning values to  $x_4$  will lead to application of  $e_3$ , 2 confluences ( $e_2$  &  $e_5$ ) will have 1 variable unassigned, and the remaining 2 confluences will have 2 variables unassigned. If values are assigned to  $x_2$ ,  $e_2$  can be applied, 2 confluences ( $e_1$  &  $e_3$ ) will have 1 variable unassigned, 1 confluence ( $e_5$ ) will have 2 variables unassigned and the other confluence will have 3 variables unassigned. Thus select  $x_4$ .

Step 5. For each combination of  $x_3$  &  $x_5$  (in step 3), find all values of  $x_4$  which satisfy  $e_3$ . Tag  $x_4$  as a variable whose value has been assigned and eliminate  $e_3$ .

Step 6. Assigning values to either  $x_2$  or  $x_6$  will lead to application of 1 confluence, 2 confluences will have the value of 1 variable unassigned and 1 confluence will have the value of 2 variables unassigned. Thus arbitrarily select either of  $x_2$  or  $x_6$ . Select  $x_2$ .

Step 7. For previous combinations of  $x_3$ ,  $x_5$  &  $x_4$ , find values of  $x_2$  satisfying  $e_2$ . Tag  $x_2$  as a variable whose value has been assigned and eliminate  $e_2$ .

Step 8. Assigning values to either  $x_1$  or  $x_6$  will lead to application of 1 confluence and 2 confluences will have the value of 1 variable unassigned. Arbitrarily select  $x_1$ .

Step 9. For previous combinations of  $x_3$ ,  $x_5$ ,  $x_4$  &  $x_2$ , find all values of  $x_1$  satisfying  $e_1$ . Tag  $x_1$  as variable whose value has been assigned and eliminate  $e_1$ .

Step 10. Only  $x_6$  remains. For previous combinations of  $x_3$ ,  $x_5$ ,  $x_4$ ,  $x_2$  &  $x_1$ , find values of  $x_6$  satisfying both  $e_4$  and  $e_5$ . Tag  $x_6$  and eliminate  $e_4$  and  $e_5$ . All confluences have been applied.

End.

### 2.2.3. Resolving Ambiguity with Latent Constraints

The consequence of the ambiguity in qualitative modeling is that a set of  $n$  independent confluences in  $n$  unknowns is not guaranteed to yield a unique solution. If ambiguities cannot be resolved within a set of confluences, multiple solutions result. The set of

values, (+, 0, -), do not form an algebraic group under qualitative algebraic operations, therefore there is no general theory for prediction of solution uniqueness or multiplicity.

In cases where multiple solutions result, some of the solutions to the system of independent conflences may be spurious while others may be genuine and occur for certain quantitative realizations of the system. In this section, a procedure for deriving additional non-causal conflences which reduce the number of spurious solutions is outlined.

Qualitative arithmetic in general requires more equations than unknowns to generate unique solutions, thus, it is usually necessary to begin with an **overspecified** set of quantitative equations to get a fully determined set of conflences. Latent constraints are additional conflences derived from redundant quantitative equations. These redundant equations are derived by algebraic manipulation of the independent equation set, eq. (2.1).

As a general rule, redundant equations containing no more, but preferably fewer non-zero variables than the independent equations from which they are derived, are less likely to yield ambiguity, thus, most useful in eliminating spurious solutions. This is because the greater the number of non zero terms in a qualitative equation, the more likely the additional terms will yield ambiguity. No ambiguity is possible in conflences containing one or two variables.

The number of potential latent constraints that can be derived from a set of  $n$  independent equations (unknowns) can be estimated as follows. The number of ways to select  $r$  variables from  $n$  is  $C(n,r) = n!/(n-r)!r!$ . If a particular selection of  $r$  variables is made,  $n - r$  variables are eliminated from the original set, resulting in a subset of  $r$  independent equations in  $r$  variables. For each unique selection of  $r$  variables, at least one new confluence can always be obtained. Assuming just one confluence from each selection of  $r$  variables, a conservative lower bound<sup>1</sup> on the number of latent conflences that can be derived from algebraic manipulation of a set of  $n$  independent equations is the sum from  $r = 1, \dots, n$ , less the number of independent equations:

$$C(n,1) + C(n,2) + \dots + C(n,n) - n = 2^n - n - 1 \quad (2.7)$$

---

<sup>1</sup>For sparse systems of equations, a lower estimate is obtainable.

Thus, the number of possible latent constraints increases exponentially,  $O(2^n)$ , with the number of unknowns.

Equation (2.7), shows that equations involving one unknown (i.e. the analytical solution) are included in the set of redundant equations. Thus, if an analytical solution to the system exists, the redundant set of non-causal, steady state confluences is theoretically able to provide the ultimate qualitative response from all (stable and unstable) steady states without other solutions. However, because of analytical complexities and the large number of potential latent confluences, a heuristic approach to deriving latent constraints is needed.

Although the heuristics used to derive latent confluences may be domain dependent, the following heuristics have been employed to guide the search for useful latent constraints in several domains:

- (a) **Global balances for conserved quantities.** Because of the ambiguity of qualitative arithmetic, qualitative balances for conserved fundamental quantities around subsystems do not necessarily assure conservation of the quantities in the combined system. Examples of these quantities are total mass, species mass, energy and electric current. Equations (2.11a) and (2.11b), below, are latent confluences of this type.
- (b) **Compatibility relationships.** These confluences derive from equating driving forces around loops for potential driven flows. Confluences derived from Kirchoffs voltage law and pressure compatibility relationships in bypass or recycle loops, eq. (2.11c), are examples of this type.
- (c) **Balances from bilinear conservation relationships.** In several domains, conserved fundamental quantities (e.g. energy) are expressed as products of extensive (mass or volume) and intensive variables (e.g. temperature). Eliminating one of the extensive variables (by using an extensive variable balance) from bilinear conservation equations around the subsystem often results in useful confluences, eq. (2.10c). Confluences derived from redundant component balances ( $m$  species plus an overall balance) in systems with  $m$  components (eq. (B36), Appendix B) are also examples of this type.
- (d) **Elimination of groups of variables.** Characteristic groupings of variables often occur in equations. Elimination of these groups can result in useful latent

confluences, especially when the resulting equation contains fewer variables than one of the original equations. Equation (B37), which is obtained by eliminating the reaction rate term from the reactant mass balance and reactor energy balance in the example of Section 2.4, is an example of this type.

(e) **Causal Constraints.** The requirement of causality adds two types of constraints on the behavior of the system. First, causality provides latent constraints that may otherwise be difficult to derive from the independent equation set. Second, causality provides new constraints, not latent in the set of steady state algebraic equations, which help eliminate responses from unstable steady states, as will be discussed in Section 2.3.

As yet we have no general method of determining a priori, how many and which latent confluences are required to eliminate all spurious solutions. In our experience, where a unique qualitative solution is obtained, the number of non-causal latent confluences (types (a) through (d) above) required to eliminate all spurious solutions is less than the number of independent equations.

#### 2.2.4. Tradeoff Between Modularity and Multiplicity

Strategies (a) and (b) for deriving latent constraints reveal a very important limitation in qualitative modeling, involving modularity. Modularity is the property that the behavior of a system can be deduced from the behavior of its components. In conventional modeling, the ability to divide a system into its component subsystems has been an extremely useful concept that has led to among other things, the creation of modular simulation programs. A similar property is desired in qualitative modeling. However, in general, confluences related to subsystems do not yield unambiguously the global behavior of the system.

In the subsequent example, it will be seen that satisfaction of qualitative mass (energy) balances on subsystems do not automatically result in satisfaction of the mass (energy) balances of the combined system. Similarly, qualitative pressure drop relations at the local or subsystem level do not guarantee pressure compatibility at the global level across recycle or bypass loops. The example demonstrates that **global** constraints are required in steady state qualitative modeling. Omission of "global" latent constraints of types (a) and

(b) can result in generation of spurious interpretations of system behavior. Therefore, in qualitative modeling, there is a trade-off between modularity and solution multiplicity.

## 2.2.5. Example of Simulation using Confluences

Figure 2.2 shows a countercurrent heat exchanger, with a bypass on the hot stream to control the cold stream outlet temperature. Boundary pressures, inlet temperatures and valve stem position are externally specified. Fluid physical properties are assumed constant. This system can be modeled with 10 state variables<sup>2</sup> using the equations listed in Appendix A.

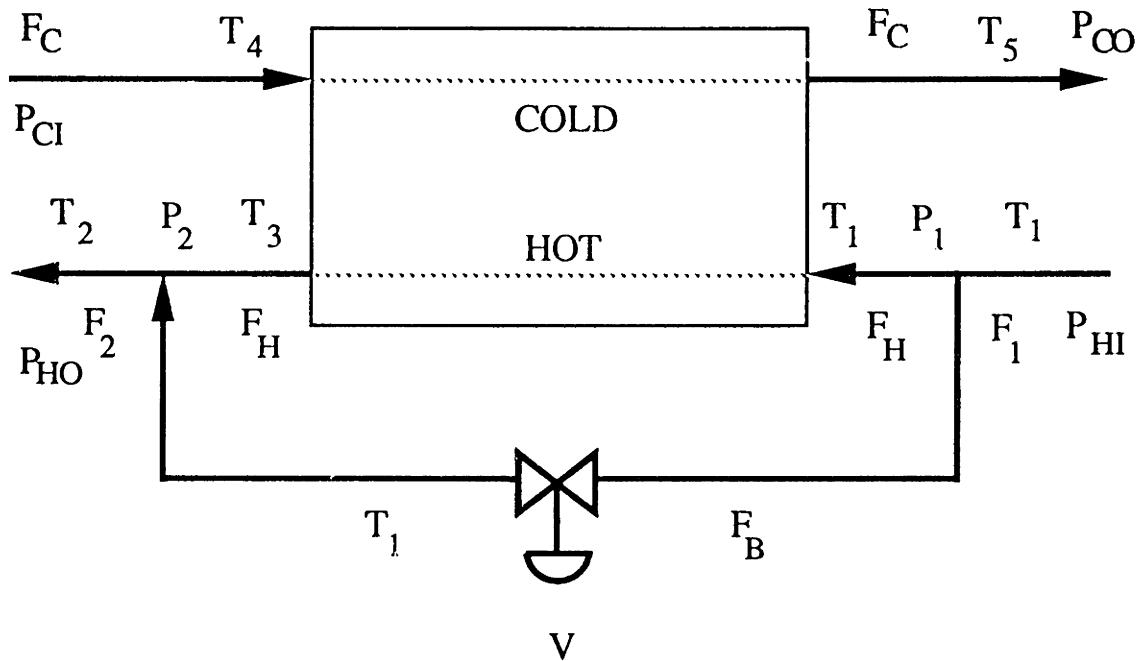


Fig. 2.2. Countercurrent Heat Exchanger with Bypass

The local (basic) steady state confluences derived from the equations in Appendix A are:

$$[dF_C] = [dP_{CI}] - [dP_{CO}] \quad (2.8a)$$

---

<sup>2</sup>Symbols representing the variables are defined in the Table of Notations at the end of the chapter.

$$[dF_1] = [dP_{HI}] - [dP_1] \quad (2.8b)$$

$$[dF_H] = [dP_1] - [dP_2] \quad (2.8c)$$

$$[dF_B] = [dV] + [dP_1] - [dP_2] \quad (2.8d)$$

$$[dF_2] = [dP_2] - [dP_{HO}] \quad (2.8e)$$

$$[dF_1] = [dF_H] + [dF_B] \quad (2.8f)$$

$$[dF_2] = [dF_H] + [dF_B] \quad (2.8g)$$

$$[dF_C] + [dT_5] - [dT_4] = [dF_H] + [dT_1] - [dT_3] \quad (2.8h)$$

$$[dF_B] + [dT_1] + [dF_H] + [dT_3] = [dF_2] + [dT_2] \quad (2.8j)$$

No confluence is derived from eq. (A.1i) because the sign of the partial derivative with respect to  $T_4$  cannot be uniquely determined from empirical ordinal relationships.

An increase in valve stem position (opening the valve) implies

$$[dV] = (+) \quad ; \quad [dU] = (0) \quad (2.9a)$$

$$[dP_{HI}] = [dP_{HO}] = [dP_{CI}] = [dP_{CO}] = [dT_1] = [dT_4] = (0) \quad (2.9b)$$

The combined system of confluentes eqs. (2.8) and (2.9), although derived from a set of equations that has a unique solution in the quantitative domain, has 105 solutions.

To reduce this solution set, three additional local latent confluentes can be derived from eqs. (A.1a) - (A.1j). These are,

$$[dT_3] = [dF_H] - [dF_C] + [dT_1] + [dT_4] - [dU] \quad (2.10a)$$

$$[dT_5] = [dF_H] - [dF_C] + [dT_1] + [dT_4] + [dU] \quad (2.10b)$$

Table 2.2 Solutions to Confluences of Heat Exchanger with Bypass

Interpr	F1	F2	F <sub>B</sub>	F <sub>C</sub>	F <sub>H</sub>	P1	P2	T2	T3	T5
1	+	+	+	0	-	-	+	+	+	-
2	+	+	+	0	-	-	+	+	0	-
3	+	+	+	0	-	-	+	+	-	0
4	+	+	+	0	-	-	+	-	-	+
5	+	+	+	0	-	-	+	0	-	+
6	+	+	+	0	-	-	+	+	-	+
7	+	+	+	0	-	-	+	+	-	-
8	+	-	-	0	+	-	-	-	+	+
9	+	-	-	0	+	-	-	0	+	+
10	+	-	-	0	+	-	-	+	+	+
11	+	+	+	0	-	-	+	-	-	-
12	+	+	+	0	-	-	+	0	-	-
13	+	0	+	0	-	-	0	0	-	-
14	+	0	+	0	-	-	0	-	-	-
15	+	0	+	0	-	-	0	+	-	-
16	+	-	+	0	-	-	-	-	-	-
17	+	-	+	0	-	-	-	0	-	-
18	+	-	+	0	-	-	-	+	-	-
19	0	+	+	0	-	0	+	-	-	-
20	0	+	+	0	-	0	+	0	-	-
21	0	+	+	0	-	0	+	+	-	-
22	-	+	-	0	+	-	+	-	+	+
23	-	+	-	0	+	-	+	0	+	+
24	-	+	-	0	+	-	+	+	+	+
25	-	+	+	0	-	-	+	-	-	-
26	-	+	+	0	-	-	+	0	-	-
27	-	+	+	0	-	-	+	+	-	-

from explicit solutions for the heat exchanger outlet temperatures, eqs. (A.2a) and (A.2b), and

$$[dF_B] + [dT_1] + [dT_3] = [dF_H] + [dT_2] \quad (2.10c)$$

from a redundant energy balance around node 2 in the bypass loop, eq. (A.2c). There are 21 solutions, Interpretations 7-27 in Table 2.2, to the basic and local latent confluences, eqs. (2.8) and (2.10).

Three global latent confluences can also be derived for this system. These are,

$$[dF_1] = [dF_2] \quad (2.11a)$$

from an overall mass balance, eq. (A.3a),

$$[dF_C] + [dT_5] - [dT_4] = [dF_1] + [dT_1] - [dT_2] \quad (2.11b)$$

from an overall energy balance, eq. (A.3b), and

$$[dF_B] = [dF_H] + [dV] \quad (2.11c)$$

from the pressure compatibility relationship around the bypass loop, eq. (A.3c). Combining these global latent confluences with the basic confluences of eqs. (2.8) and (2.9) results in 7 solutions, Interpretations 1-7 in Table 2.2.

Interpretation 7 is the only solution that satisfies all the confluences. This solution is the only possible behavior of the system and is valid for all numerical realizations of parameter values. Thus, redundant non-causal constraints derived from both local and global considerations are necessary to provide the ultimate qualitative response of systems.

## 2.3. Use of Causality in Steady State Qualitative Modeling

In this section, we show how a model of the system derived from causal considerations provides additional constraints that eliminate spurious solutions in qualitative modeling. The causal model, the Extended Signed Directed Graph (ESDG), provide constraints which are difficult to derive through algebraic manipulation of the independent equations set. More importantly, the ESDG provides new constraints that eliminate qualitative solutions corresponding to the response from unstable steady states, which are included in the set of solutions derived from the basic and latent non-causal confluences. Elimination of

solutions corresponding to unstable steady states is by purely topological arguments, and differs from the approach of Iwasaki & Simon [12], who use dynamic stability criteria. These criteria may be difficult to derive.

### 2.3.1. Representation of Causality by the ESDG

Causality is the principle that effects must be propagated locally within the topology of a system. The single-stage Signed Directed Graph (SDG), is a causal model for continuous systems, representing the immediate cause and effect relationships between process variables in the system. The SDG consists of nodes, symbolizing variables, and signed directed arcs, representing the causal influences between the variables [13]. Arc signs '+' and '-' indicate whether cause and effect variables tend to change in the same or opposite directions.

In Chapter 3, it is shown that under "optimal" disturbance propagation assumptions,<sup>3</sup> interpretations of the SDG, while guaranteed to include the initial response of the system, may exclude the ultimate response in certain systems where variables exhibit inverse or compensatory response.<sup>4</sup> Using causality to predict the ultimate response of these systems involves predicting the initial behavior combined with knowledge of variables that exhibit inverse or compensatory response.

Theorems for determining variables that exhibit inverse and compensatory response from the global topology of the SDG are derived in Chapter 3. The location of these variables leads to the construction of the ESDG for the process. In addition to arcs of the SDG, the ESDG contains additional non-physical arcs, forming feedforward paths that serve to explain inverse and compensatory responses. Disturbance propagation in feedforward paths of the ESDG provides the ultimate response of the system.

---

<sup>3</sup>These assumptions limit propagation to feedforward paths in the SDG, thus, minimizing the production of spurious interpretations.

<sup>4</sup>Inverse response occurs when the initial and ultimate direction of change of a variable are opposite. Compensatory response is exhibited if the variable returns to its initial value after all transients have died out.

## 2.3.2. Derivation of Confluences from the ESDG

The ESDG, presented initially as a graph, can be represented as an equivalent set of confluences. These confluences derive from two sources: local balances around nodes of the ESDG, and the global topology of the ESDG. Conversion of the ESDG to confluences provides a convenient uniform representation for both causal and non-causal constraints.

### 2.3.2.1. NODAL BALANCES FOR ESDGS WITHOUT FEEDBACK LOOPS

The response of a system to a disturbance is obtained by propagating its effect through feedforward paths in the ESDG. Each feedforward path represents a choice for fault propagation paths. Restriction of fault propagation to feedforward paths converts an ESDG into a set of trees branching from the root node. In these trees, the sign of each successive node is determined from the signs of the nodes immediately upstream. Therefore, each interpretation is valid only if the value of each node is consistent with transmission of effects from the root node through a feedforward path.

Starting with the root node, the qualitative value of the root node,  $[dX_0]$  is the sign of the primary disturbance [.] :

$$[dX_0] = [.] \quad (2.12)$$

A necessary condition for an interpretation of the ESDG to be valid is that the sign of other nodes must be the qualitative sum of its input effects. Mathematically, this condition can be expressed as **nodal balances** around each ESDG node:

$$[dX_j] = [\beta_{j1}] \cdot [dX_{i1}] + [\beta_{j2}] \cdot [dX_{i2}] + \dots + [\beta_{jn}] \cdot [dX_{in}] \quad (2.13)$$

where  $X_{ik}$  ( $k = 1, \dots, n$ ) are inputs to node  $j$  and  $[\beta_{jk}]$  is the sign of the arc from  $X_{ik}$  to  $X_j$ .

In an ESDG without feedback loops, all disturbed inputs to a node,  $X_j$ , are always upstream of the node in feedforward (and only feedforward) paths from the root node.

Therefore, the solution to a set of nodal balances as in eq. (2.13) will give exactly the set of all interpretations from feedforward paths, and no others.

In summary, the confluentes required to determine steady state system behavior for an ESDG without feedback loops are:

- A confluence describing the primary deviation.
- Nodal balances such as eq. (2.13) for all other nodes in the ESDG.

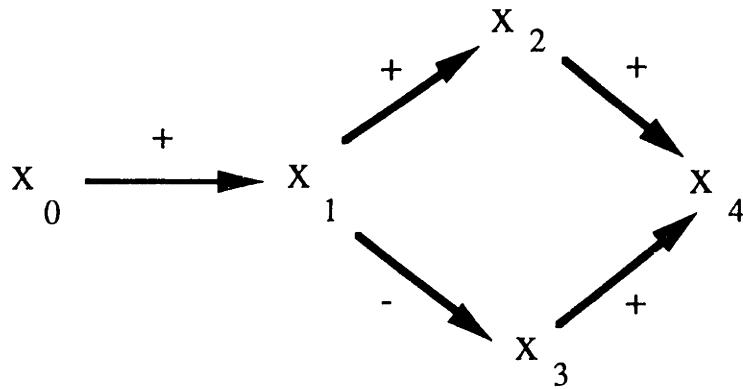


Fig. 2.3. ESDG of Hypothetical System without Feedback Loops

Table 2.3 Solutions to ESDG without Feedback Loops (Fig. 2.3)

Interpr.	[dX1]	[dX2]	[dX3]	[dX4]
1	+	+	-	+
2	+	+	-	0
3	+	+	-	-

For example, Fig. 2.3 shows the ESDG of a hypothetical system. Three feedforward paths can be enumerated for this system, corresponding to  $X_4$  receiving dominant inputs

from  $X_2$  or  $X_3$  or jointly from  $X_2$  and  $X_3$ . For a positive primary deviation in  $X_0$ , the solutions to these paths are shown in Table 2.3.

The set of nodal balances for this system is

$$[dX_0] = (+)$$

$$[dX_1] = [dX_0]$$

$$[dX_2] = [dX_1] \quad (2.14)$$

$$[dX_3] = - [dX_1]$$

$$[dX_4] = [dX_2] + [dX_3]$$

These confluences are interpreted according to the qualitative operations in Table 2.1. Appearance of the ambiguous value, (?), on one side of the equality in a confluence automatically satisfies the confluence, and variables whose values are undetermined, are assigned each of the possible values (+), (0) and (-). It can easily be shown that the only solutions that satisfy all the confluences in eq. (2.14) are the interpretations of the feedforward paths in Table 2.3.

### 2.3.2.2. SYSTEMS WITH NEGATIVE FEEDBACK LOOPS

In the preceding section, it was shown that for a system without feedback loops, a set of nodal balances gives exactly the set of interpretations obtained by propagation in all feedforward paths in the ESDG. In systems with feedback loops, an input node to a node  $X_j$ , may be in a loop. Thus the nodal balances could allow propagation in loops, leading to spurious interpretations.

In a negative feedback loop, disturbance propagation to the loop variable ( $X_j$ ) just downstream of the disturbance variable, from a variable in the loop results in inverse or compensatory response. Cases of inverse and compensatory response caused by disturbance propagation in loops in negative feedback loops result in local inconsistencies of nodal balances in the ESDG. For example, if inverse response displayed by  $X_1$  with

respect to a positive perturbation of  $X_0$ , is considered in light of the ESDG of Fig. 2.4, there is no positive causal effect that explains the positive deviation of  $X_2$ . That is, the only non-zero causal effect on  $X_2$  at the steady state is the negative effect transmitted from  $X_1$ .

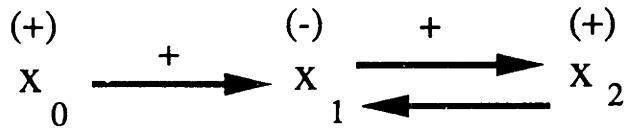


Fig. 2.4. ESDG of System with Negative Feedback Loop

In general, disturbance propagation around negative feedback loops leads to an inconsistency in the nodal balance for the loop variable not exhibiting inverse response just downstream of loop variables exhibiting inverse response. Therefore, nodal balances are sufficient to determine the steady-state behavior for the ESDG of these systems, as they do not allow additional spurious interpretations obtained by propagating around negative feedback loops.

In summary, the confluences required to determine steady state system behavior for an ESDG with negative feedback loops are:

- A confluence describing the primary deviation.
- Nodal balances such as eq. (2.13) for all other nodes in the ESDG.

### 2.3.2.3. GLOBAL CONFLUENCES FOR POSITIVE FEEDBACK LOOPS

In systems with positive feedback loops, nodal balances admit some spurious interpretations. These interpretations are associated with propagation paths (loops) that are causally isolated from the disturbance origin. Additional confluences are required to eliminate these interpretations.

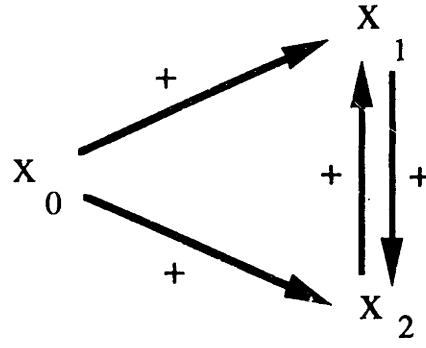


Fig. 2.5. ESDG of System with Positive Feedback Loop

Consider the ESDG of the system in Fig. 2.5. When there is no disturbance to the system the nodal balances for the system are:

$$[dX_0] = (0)$$

$$[dX_1] = [dX_2] + [dX_0] \quad (2.15)$$

$$[dX_2] = [dX_1] + [dX_0]$$

The solutions that satisfy eq. (2.15) are shown in Table 2.4. The second and third interpretations are spurious, and are clearly not solutions of the ESDG. These interpretations are due to propagation in the isolated positive feedback loop between  $X_1$  and  $X_2$ . For asymptotically stable physical systems, these interpretations cannot persist as a sustained input is required to maintain an output.

In order to eliminate these interpretations, an additional confluence

$$[dX_1] = [dX_0] \text{ or } [dX_2] = [dX_0] \quad (2.16a)$$

is needed. The confluence is related to the global topology of the ESDG. It is equivalent to the condition that the feedback loop must have an external input to sustain a signal. Equation (2.16a) eliminates spurious interpretations in eq. (2.15) (both for  $[dX_0] = (0)$  and  $[dX_0] \neq (0)$ ).

Table 2.4 Solutions to Nodal Balances of Eq. (2.15)

Interpr.	[dX1]	[dX2]
1	0	0
2	+	+
3	-	-

Disjunction of confluentes presents some computational difficulties. In order to avoid using a disjunction of confluentes, eq. (2.16a) can also be rewritten as

$$\delta([dX_1] - [dX_0]) + \delta([dX_2] - [dX_0]) = (+) \quad (2.16b)$$

The  $\delta$  "qualitative Dirac delta" function is interpreted as in Table 2.1.

It is readily proven that eqs. (2.16a) and (2.16b) have identical solutions. In general, for systems with feedback loops, an additional topological confluence is required for each positive feedback loop that may be isolated from the disturbance origin. These confluentes are of the form,

$$\sum \delta([dX_{jk}] - [\beta_{jk}].[dX_{ik}]) = (+) \quad (2.17)$$

where  $ik$  ( $k = 1,2,\dots,n$ ) are disturbance variables to nodes  $jk$  within the loop. The confluentes in eqs. (2.12), (2.13), and (2.17) exactly determine the valid steady state solutions under the ESDG.

In summary, the confluentes required to determine steady state system behavior for an ESDG with positive feedback loops are:

- A confluence describing the primary deviation.
- Nodal balances such as eq. (2.13) for all other nodes in the ESDG.

- Global topological confluences such as eq. (2.17) for each positive feedback loop.

#### 2.3.2.4. USING GLOBAL TOPOLOGY TO DERIVE CAUSAL CONFLUENCES

In summary, global causal topology is used in two ways in deriving confluences from causal information. First, global topology provides necessary conditions for certain variables exhibiting inverse and compensatory response, leading to the addition of non-physical arcs to the SDG to form the ESDG. These arcs ensure that the ultimate response of the system is not excluded in cases of inverse and compensatory response. Second, global topological confluences are derived from the positive feedback loops (PFBs) of the ESDG. These confluences eliminate solutions not consistent with propagation along feedforward paths of the ESDG.

### 2.3.3. Reducing Ambiguity with Causal Confluences

Confluences derived from causal considerations provide additional constraints that eliminate spurious steady state interpretations of system behavior. First, causal confluences provide constraints that are difficult to derive through algebraic manipulation. Second, they eliminate solutions (admitted by non-causal confluences) that correspond to the response from unstable steady states.

#### *Example 1*

The following example, illustrates the utility of latent constraints (derived from causal considerations) that are difficult to derive by algebraic manipulation of independent equations. An important latent constraint in counter-current heat exchangers involves the hot fluid outlet temperature:

$$[dT_{HO}] = [dT_{HI}] + [dT_{CI}] + [dF_H] - [dF_C] - [dU] \quad (2.18)$$

Beginning with the heat balances for the hot and cold process streams, several complex steps involving partial solution, substitution, and simplifications are required to arrive at the

redundant equation relating the hot side outlet temperature to inlet conditions in a countercurrent heat exchanger:

$$T_{HO} = \frac{T_{HI}(1 - R) + (e^\phi - 1)T_{CI}}{(e^\phi - R)} \quad (2.19)$$

where

$$R = (\rho C_p F)_H / (\rho C_p F)_C, \quad R \neq 1$$

$$\phi = UA_C(1 - R) / (\rho C_p F)_H, \quad \phi \neq 0$$

Determination of the signs in eq. (2.18) from eq. (2.19) involves evaluation of partial derivatives such as,

$$\frac{\partial T_{HO}}{\partial F_C} = \frac{R(1 + (\phi - 1)e^\phi)(T_{CI} - T_{HI})}{F_C(e^\phi - R)^2} \quad (2.20)$$

This partial derivative involves the difference of positive terms and, in general, it may not be possible to unambiguously determine the sign of such quantities for all numerical realizations of parameter values. For eq. (2.20), detailed analysis shows that this term must always be negative.

While it is difficult to derive the analytical relationship between outlet temperatures and inlet temperatures and flowrates, this information is readily available from a nodal balance around  $T_{HO}$  in the ESDG, shown in Fig. 2.6. The ESDG reflects the well-known behavior of counter-current heat exchangers. The ESDG contains the information which is obtained only with difficulty from the process equations.

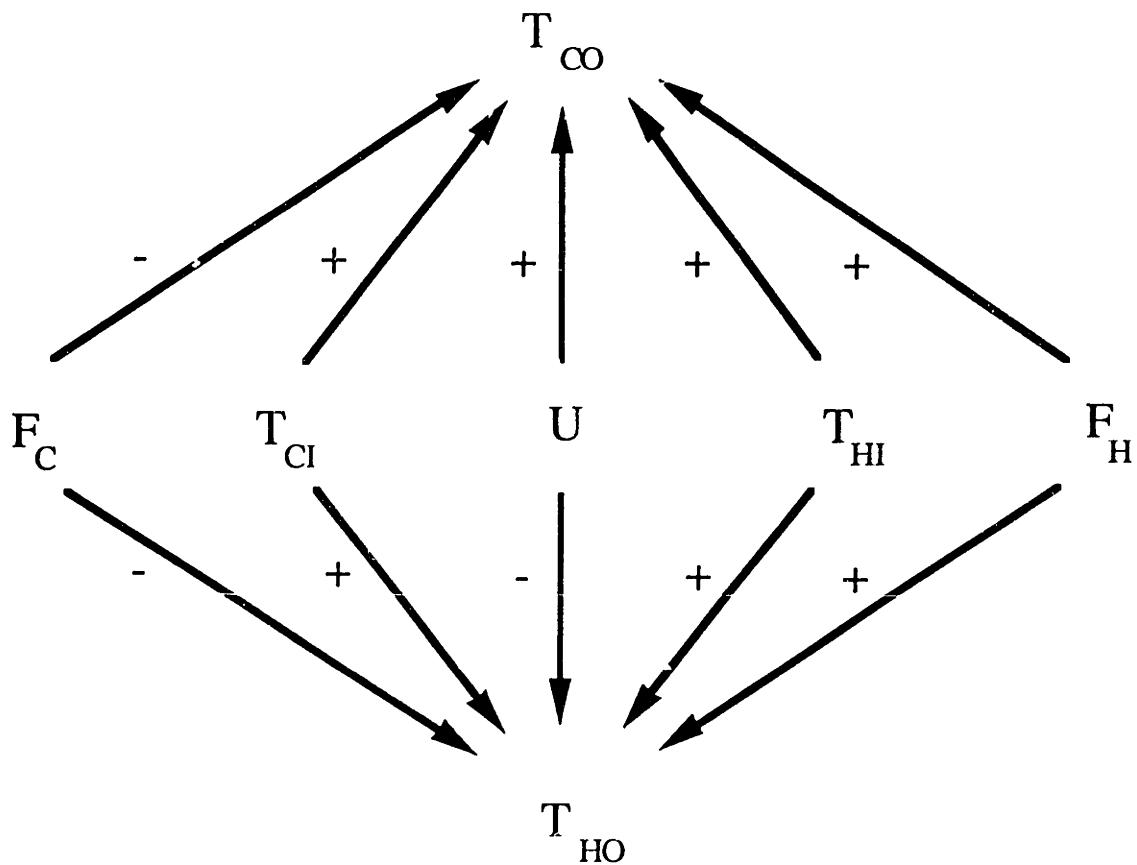


Fig. 2.6. ESDG for Countercurrent Heat Exchanger

### *Example 2*

The following example, illustrates the use of causal confluences in eliminating spurious solutions corresponding to unstable steady states. Consider an isothermal, irreversible, autocatalytic reaction ( $A \rightarrow B$ ), with rate expression

$$r_A = k_r C_A C_B^2 \quad (2.21)$$

taking place in a continuous stirred tank reactor (CSTR), with constant space velocity. The dynamic process equations are:

$$\frac{dC_A}{dt} = F(C_{A0} - C_A)/V - k_r C_A C_B^2 \quad (2.22a)$$

$$\frac{dC_B}{dt} = k_r C_A C_B^2 - F C_B / V \quad (2.22b)$$

For this system, ( $C_{A0} > C_A$ ). The basic and latent non-causal confluences are:

$$[dC_{A0}] = [dC_A] + [dC_B] \quad (2.23a)$$

$$[dC_A] + [dC_B] = 0 \quad (2.23b)$$

Table 2.5 Solutions to Non-Causal Confluences for Autocatalytic Reaction

Interpr.	[dC <sub>A0</sub> ]	[dC <sub>A</sub> ]	[dC <sub>B</sub> ]
1	+	+	-
2	+	-	+

The solutions to these confluences for an increase in  $C_{A0}$  are listed in Table 2.5. These solutions correspond to the ultimate response from the two steady state solutions of eqs. (2.22), for  $C_{A0}^2 > 4F/k_r V$ :

$$C_{AS} = \frac{C_{A0} - (C_{A0}^2 - 4F/k_r V)^{1/2}}{2} \quad (2.24)$$

The first steady state (invoking the plus sign in eq. (2.24)) is unstable. Thus, the former solution in Table 2.5 is spurious and cannot be observed in a physical system.

The ESDG for this system is shown in Fig. 2.7. The non-physical arc between  $C_{A0}$  and  $C_B$  accounting for possible inverse response of  $C_A$  to changes in  $C_{A0}$  is constructed according to the procedure in Chapter 3. Nodal balances for the ESDG yield the following confluences:

$$[dC_A] = [dC_{A0}] - [dC_B] \quad (2.25a)$$

$$[dC_B] = [dC_A] + [dC_{A0}] \quad (2.25b)$$

The first of these is the same as eq. (2.23a), however, eq. (2.25b) is a new latent constraint which eliminates the latter solution in Table 2.5. The response from the stable steady state is the only one that satisfies both non-causal confluences and confluences derived from causality.

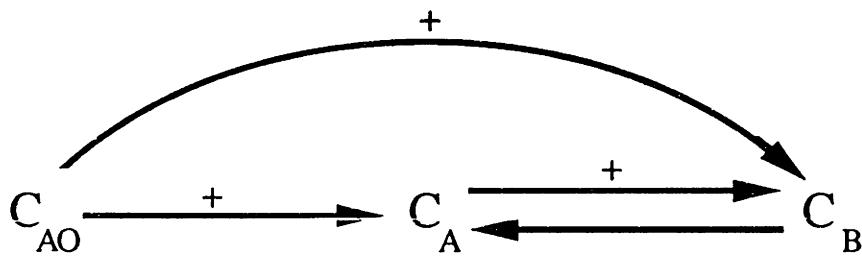


Fig. 2.7. ESDG for Autocatalytic Reaction

## 2.4. Qualitative Fault Simulation of a Chemical Reactor System

To demonstrate the use of both causal and non-causal confluences in determining the steady state qualitative behavior of continuous process systems, we consider the process shown in Fig. 2.8. An irreversible heterogeneous catalytic exothermic reaction ( $A \rightarrow mB$ ), with rate expression

$$r_A = k_f C_A^n \quad , \quad n > 0 \quad (2.26)$$

takes place in a continuous stirred tank reactor. To provide temperature control, part of the reactor outlet stream is recycled to the reactor through a heat exchanger. The recycle flow rate is controlled, and residence time in the reactor is controlled by maintaining the level of reactants in the reactor. Constant boundary pressures of all input and output streams and

constant physical properties are assumed. The objective is to determine the ultimate system response to various faults and disturbances.

Although more robust control schemes may exist, this system was chosen because it presents a significant challenge for qualitative modeling. The system possesses several features which tend to generate a great deal of ambiguity. It:

- Is highly non-linear.
- Has interacting control loops.
- Has multiple causal pathways with opposing tendencies between variables.
- Attains multiple steady states in the absence of temperature control.

The system can be modeled by as many as 166 independent state variables, but by eliminating obvious equalities and introducing definitions for 3 parameters, 27 unknowns (24 variables) result. By using empirical ordinal relationships, 26 basic confluences can be derived for the system.

Non-causal confluences (under fault-free conditions) and conditions at which they are valid are presented in Appendix B. Confluences (B10, B11, B30 & B31) are valid when the reactor outlet flowrate ( $F$ ) is greater than the recycle flowrate ( $F_R$ ). Similarly, (B37 & B38) are valid when the reactor temperature ( $T$ ) is higher than the recycle stream temperature ( $T_R$ ). These conditions are violated only in the extreme cases of reverse flow in the product stream or the cooling system acting as a heat source, respectively.

Transitions between qualitative regimes due to violation of these conditions will not occur for most faults. Depending on fault magnitudes, perfect control of controlled variables may be maintained or the control loop may become saturated at the new steady state. Control loop saturation will be allowed, and the relevant controller confluences (B19, B22 & B25), account for these possible transitions. The only other transitions that may occur are to regimes where process equations are no longer valid, such as boiling. Thus, the assumption of no transitions between qualitative regimes (except for control loop saturation), discussed in Section 2.2.1 is not very limiting.

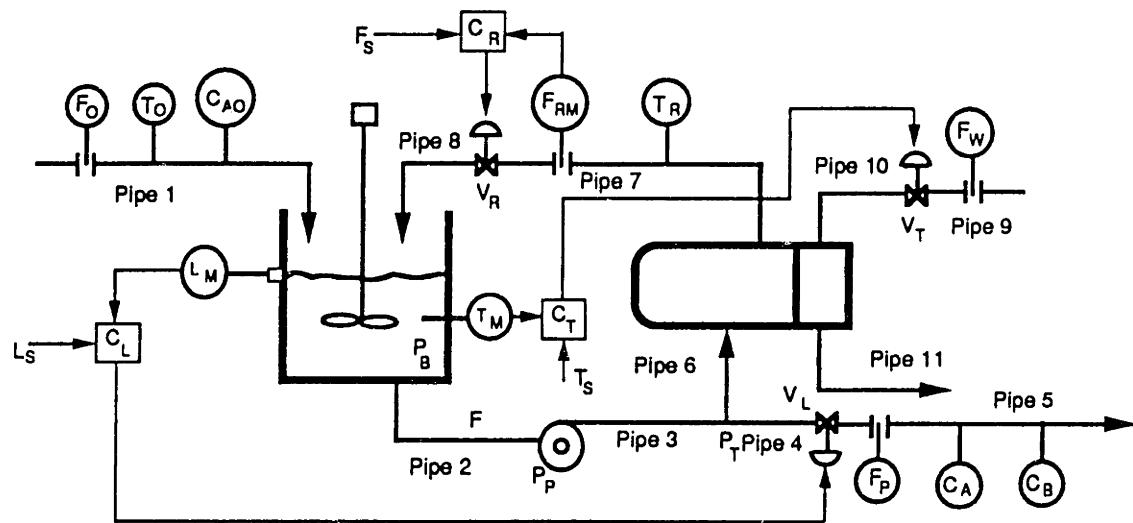


Fig. 2.8. Continuous Stirred Tank Reactor with Recycle

Table 2.6 ESDG Arcs for Control Loops of CSTR with Recycle

Controlled variable	Initial node	Terminal node	Arc sign
FR	P <sub>0R</sub>	C <sub>R</sub>	+δ([dFR])
FR	P <sub>T</sub>	C <sub>R</sub>	-δ([dFR])
L	F <sub>0</sub>	C <sub>L</sub>	+δ([dV])
L	P <sub>0P</sub>	C <sub>L</sub>	+δ([dF <sub>P</sub> ])
L	F	C <sub>L</sub>	-δ([dV])
L	F	C <sub>L</sub>	+δ([dPT])
L	F <sub>R</sub>	C <sub>L</sub>	+δ([dV])
L	F <sub>R</sub>	C <sub>L</sub>	-δ([dPT])
L	P <sub>0R</sub>	C <sub>L</sub>	-δ([dFR])
L	P <sub>P</sub>	C <sub>L</sub>	-δ([dF])
L	V <sub>R</sub>	C <sub>L</sub>	+δ([dFR])
T	F <sub>0</sub>	C <sub>T</sub>	+δ([dT])
T	F <sub>0</sub>	C <sub>T</sub>	-δ([dT])
T	F <sub>R</sub>	C <sub>T</sub>	+δ([dTR])
T	F <sub>R</sub>	C <sub>T</sub>	-δ([dT])
T	P <sub>IW</sub>	C <sub>T</sub>	-δ([dFW])
T	P <sub>OW</sub>	C <sub>T</sub>	+δ([dFW])
T	r <sub>A</sub>	C <sub>T</sub>	+δ([dT])
T	T <sub>0</sub>	C <sub>T</sub>	+δ([dT])
T	T <sub>WI</sub>	C <sub>T</sub>	+δ([dTR])
T	U	C <sub>T</sub>	-δ([dTR])
T	V	C <sub>T</sub>	+δ([dT])

The ESDG for the system is shown in Fig. 2.9. Positive arcs between C<sub>A0</sub> and r<sub>A</sub>, and F<sub>0</sub> and T are non-physical feedforward paths which explain possible inverse response of reactant concentration to changes in feed concentration and flowrate, respectively. The non-physical conditional arc between V<sub>L</sub> and P<sub>T</sub>, δ([dF<sub>P</sub>]), is dependent on the value of F<sub>P</sub>, and accounts for compensatory response of product flowrate to changes in the level control valve stem position. Non-physical arcs that explain compensatory response in control loops (not shown in Fig. 2.9) are listed in Table 2.6. All other arcs of the ESDG represent local causal relationships expressed in the process SDG. Nodal balances and global topological confluentes for positive feedback loops (under fault-free conditions) derived from the ESDG are presented in Appendix C.

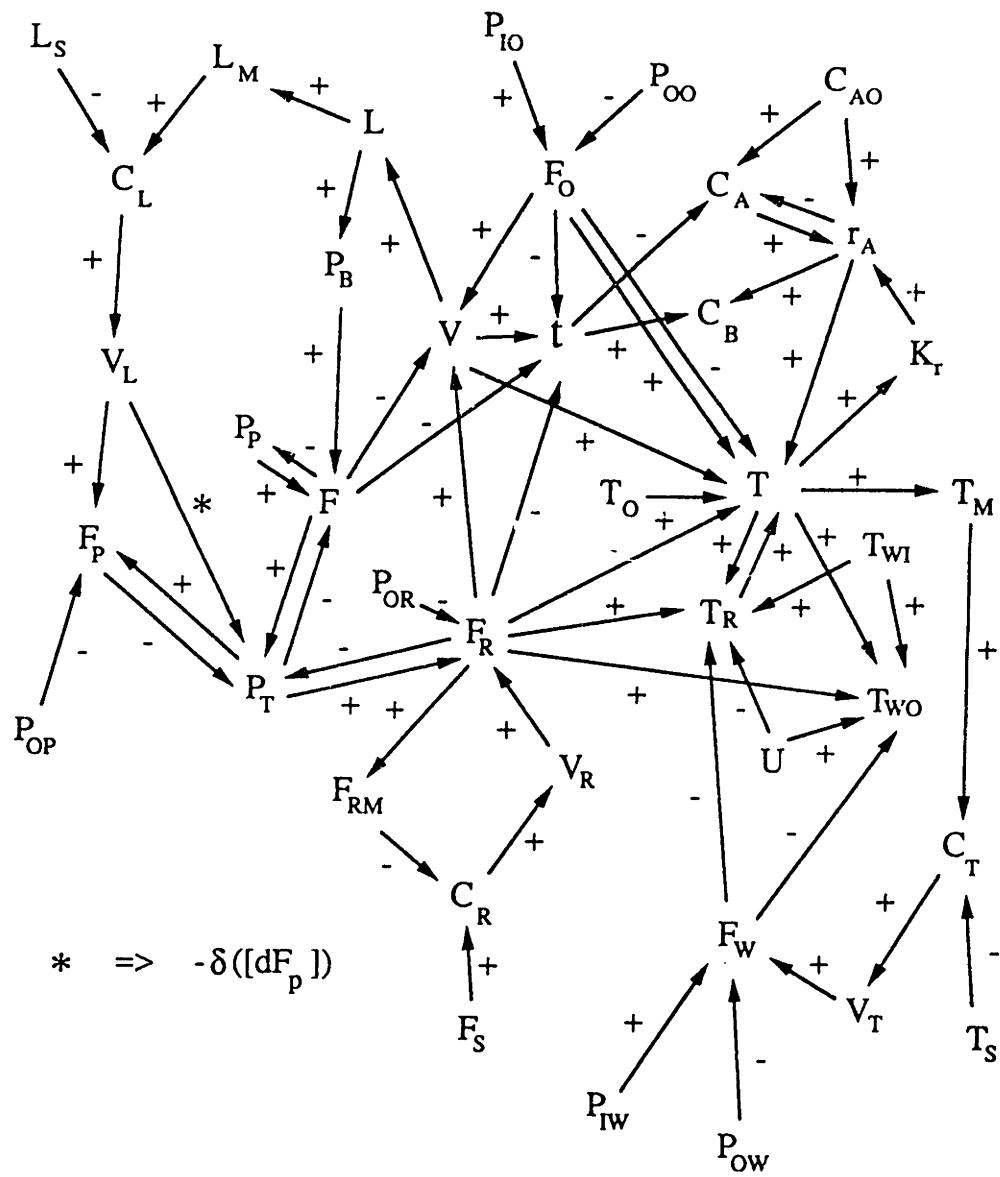


Fig. 2.9. ESDG for CSTR with Recycle

Table 2.7 Selected Faults for CSTR with Recycle

#	Fault
1	Normal operation
2	Pipe 1 partially blocked
3	Pipe 6 partially blocked
4	Feed concentration high
5	Recycle flow setpoint high
6	Heat exchanger fouling
7	Catalyst deactivation
8	Temperature control valve stuck open
9	Leak in reactor
10	Recycle flowmeter stuck high

Selected faults and disturbances for this system are listed in Table 2.7. These faults are modeled by a qualitative change in one or more of the process inputs or parameters, or by modifying the appropriate confluences.

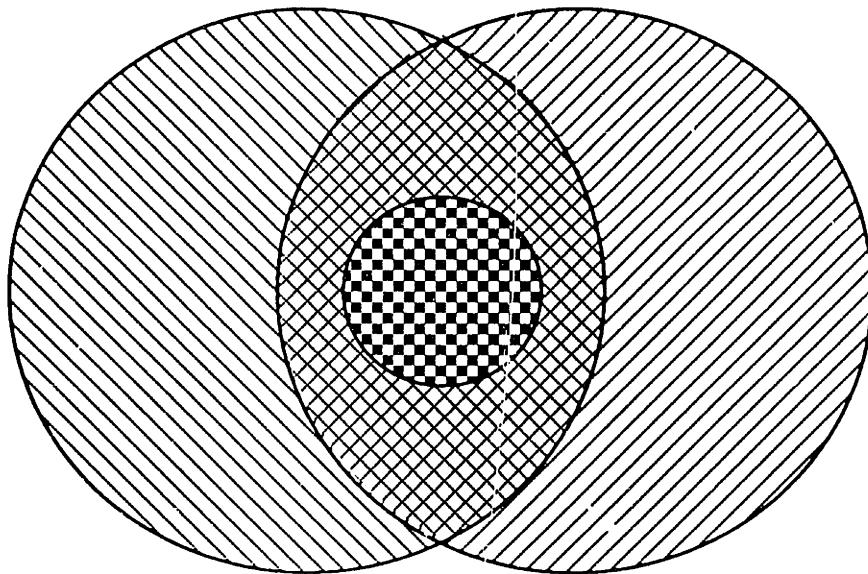
Table 2.8, shows the number of observable qualitative steady state solutions for the set of measured variables given in Table 2.9, for these faults when using:

- (i) causal confluences derived from the ESDG,
- (ii) confluences derived from a "basic" set of 27 equations in 27 unknowns,
- (iii) the combination of confluences from (i) and (ii), and
- (iv) non-causal latent confluences in addition to those in (iii).

Table 2.8 Number of Solutions Obtained Using Confluences from Various Sources

Fault	Causal (i)	Local non-causal (ii)	Causal & local non-causal (iii)	Local, latent & causal (iv)
1	1	279	1	1
2	1079	984	481	141
3	501	1359	192	13
4	12	287	10	10
5	501	4869	192	13
6	2	279	2	2
7	4	281	4	3
8	1	279	1	1
9	984	1080	178	35
10	309	1215	138	9

The relationship between the solutions (i) - (iv) is most clearly depicted in Fig. 2.10. Combining constraints from causal and non-causal sources reduces the number of solutions obtained with the ESDG by up to a factor of 2 to 3. Adding latent process constraints leads to further reductions of up to an order of magnitude. The overall reduction in the number of solutions obtained between cases (i) and (iv) is up to a factor of 35, and between cases (ii) and (iv) up to a factor of 370. Thus, combining causal and non-causal confluences significantly reduces ambiguity. However, it cannot be proven (without further analysis, possibly numerical) that all solutions obtained for case (iv) are realizable [8].



- causal confluences from ESDG (case i)
- local non-causal confluences (case ii)
- causal and local non-causal confluences (case iii)
- causal, local and latent non-causal confluences (case iv)

Fig. 2.10. Relationship Between Solutions to Different Sets of Confluences

#### 2.4.1. Using Knowledge of Control Loop Function to Reduce Ambiguity

The function of an engineered system is the purpose for which the system was designed. A priori knowledge of the function of the system can be used to reduce the ambiguity in qualitative modeling [10 & 15]. In chemical process systems, feedback control loops attempt to maintain values of important variables within acceptable ranges, thus playing an important role in determining system behavior. Control loops are designed to achieve their

function in the presence of most expected disturbances. Thus, in many situations, zero deviation of the controlled variable is maintained after all transients have died out. Exceptions occur when the disturbance magnitude is large enough to cause loop saturation, or when the fault prevents loop operation, as in the case where an element of the control loop is stuck at a fixed position (zero-gain events).

Table 2.9 Results of Qualitative Simulation (Perfect Control)

Fault	$F_0$	$T_0$	$C_{A0}$	$F_w$	$T_M$	$L_M$	$T_R$	$C_T$	$C_L$	$C_R$	$F_P$	$C_A$	$C_B$
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	-	0	0	-/0/+	0	0	+/-	-/0/+	-	-	-	-	+
3	0	0	0	0	0	0	0	0	+	0	0	0	0
4	0	0	+	+	0	0	-	+	0	0	0	+	-
5	0	0	0	-	0	0	+	-	+	+	0	0	0
6	0	0	0	+	0	0	0	+	0	0	0	0	0
7	0	0	0	-	0	0	+	-	0	0	0	+	-
8	0	0	0	+	-	0	-	-	0	0	0	+	-
9	0	0	0	0	0	0	0	0	-	-	-	0	0
10	0	0	0	+	+	0	-	+	-	-	0	-	+

The results of applying confluences (in case (iv) of Section 2.4) with the additional constraint of control loop functionality (except for zero-gain events) are shown in Table 2.9. From Table 2.9, it can be seen that confluences produce one solution for most faults. The fault that produces multiple solutions is associated with a decrease in feed rate to the reactor. The steady state balances for the system are

$$F = F_0 + F_R \quad (2.27)$$

$$FC_A = F_0 C_{A0} + F_R C_A - k_r A L C_A^n \quad (2.28)$$

$$\rho C_p F T = \rho C_p F_0 T_0 + \rho C_p F_R T_R + k_r A L C_A^n \quad (2.29)$$

Noting that the steady-state values of  $C_{A0}$ ,  $F_R$ ,  $L$ ,  $T$ , and  $T_0$  are unchanged after fault initiation, a differential analysis yields,

$$dTR = \frac{[\rho C_p(T - T_0) - k_r A L (-\Delta H_R) n C_A^{n-1} (C_{A0} - C_A)] dF_0}{\frac{F_0 + k_r A L n C_A^{n-1}}{\rho C_p F_R}} \quad (2.30)$$

The sign of the coefficient of  $dF_0$  in eq. (2.30), is ambiguous (for the stable system) and may be positive or negative depending on the values of process variables and parameters. Thus, all solutions in Table 2.9 are realizable.

Caution is required when using a priori knowledge about a system's function to deduce behavior. In certain applications, such as in predicting system behavior to unintended disturbances and changes in system structure (fault simulation), the system may not achieve its intended function. However, if the intended function is achieved, knowledge about function significantly reduces ambiguity. In our experience, spurious solutions have never been produced by causal and non-causal confluences, with the additional constraints of functioning control loops, for all examples we have tried in the chemical engineering domain.

## 2.5. Discussion

In this chapter, it was demonstrated that multiple (quantitatively redundant) sources of information are necessary to model the steady state qualitative behavior of continuous systems. These sources are summarized in Table 2.10. In one dimension, they can be classified as non-causal or causal. In the other, as local or global. Constraints derived from these sources are required to minimize the ambiguity inherent in qualitative modeling.

Non-causal latent confluences were derived through "heuristic" algebraic manipulation of basic constraints. Latent confluences correspond to a partial analytic solution of the system and therefore reduce the number of qualitative solutions. However, qualitative solutions corresponding to responses from unstable steady states are admitted by non-causal confluences. Causal confluences eliminate some of these spurious solutions (unstable steady states) and also provide constraints which may be difficult to derive through algebraic manipulation. Combining constraints from these various sources theoretically eliminates all spurious solutions in systems where analytic steady state solutions exist.

Table 2.10 Information Sources for Steady State Qualitative Modeling

	Non-causal	Causal
Local	basic confluences latent confluences	local causality (SDG)
Global	latent confluences	prior knowledge of inverse and compensatory variables global topological confluences (PFBL)

One of the major concepts resulting from this study is the requirement of global information in steady state qualitative modeling. This points out an inherent trade-off between modularity and solution multiplicity. Therefore, computer programs using steady state qualitative models need to store both local causal and non-causal models. Additionally, the ability to generate global confluences corresponding to particular system topographies from these models is required. Care is required in generating these global confluences, since they are dependent on the validity of the relevant local models.

The steady state solutions obtained by the qualitative modeling method can be used to test a fault hypothesis as part of a hypothesis/test diagnostic cycle [16]. However, there are limitations in using the steady state qualitative models. Dynamic effects were ignored, and it was assumed that the same qualitative constraints were also applicable at the new steady state. This assumption is violated when transitions across qualitative regimes occur, and could lead to the exclusion of the steady state solution. The modeling method is not applicable to predictive safety analysis, where qualitative transitions are of primary importance. In processes that exhibit inverse response, compensatory response or display

oscillatory behavior, the steady state patterns may also exclude some transient patterns. This is important in systems with large time constants where the system takes a "long" time to achieve a new steady state.

Our qualitative models have another drawback, in that they are incompatible with quantitative information. This information may be available and could be necessary when solving particular problems. One approach to this problem has led to the development of the order of magnitude formalisms [17 & 18]. These are partial solutions to this problem, in that they extend qualitative modeling by utilizing knowledge of relative magnitudes of quantities in addition to their signs in making inferences. In Chapter 5, the event modeling paradigm [19], is described. This paradigm, allows the introduction of quantitative information, and retains qualitative relationships when numerical information is not required or unavailable.

In addition, qualitative constraints do not express a system's function. Where the intended function is achieved, knowledge about the system's function can be used to reduce ambiguity. Additionally, reasoning about function can facilitate understanding of system behavior. In complex systems, human experts tend to reason about the functionality of subsystems in relation to overall system function before reasoning in more detail using knowledge about physical structure at the subsystem level [20]. The need for hierarchical representations has been recognized, and has been a subject of recent research [21 & 22]. These representations can be used within the event modeling paradigm.

Richer representations<sup>5</sup> of equipment and process streams are required to incorporate qualitative dynamic effects, necessary for on-line diagnosis. These representations should also eliminate spurious interpretations of dynamic behavior. In Chapter 3, we exploit the need for global information in qualitative modeling, when developing dynamic models.

---

<sup>5</sup>These representations should be richer than confluences.

## 2.6. Notation for Chapter 2

A	chemical species, A
$A_A C$	reactor cross sectional, heat exchange area
$A_0$	Arrhenius frequency factor
B	chemical species, B
$C_A, C_B,$	concentration of A,B in reactor
$C_{A0}$	feed concentration of A
$C_L, C_R, C_T$	controller output signals
$C_p$	specific heat capacity
$E_a$	activation energy
$f^+, f^-$	strictly monotonic increasing, decreasing functions
F	liquid flowrate
$F, F_0, F_p$	reactor outlet, feed, product, flowrates
$F_R, F_{RM}, F_S$	recycle flowrate, measurement, setpoint
$F_W$	cooling water flowrate
$H_g$	heat generated due to reaction
$H_L$	heat load in heat exchanger
$H_T$	sensible heat change from feed stream to product stream
$k, k_I, k_p$	constants
$k_1, k_2, k_r$	reaction rate constants
$L, L_M, L_S$	reactor level, level measurement and setpoint
m	stoichiometric coefficient
n	order of reaction
O	order of magnitude
p	system parameter
$P_1, P_2, P_T$	pressure at node
$P_B$	reactor base pressure
$P_p$	pump head
R	gas constant
$r_A$	reaction rate
$R_s$	flow resistance in conduit
$T, T_0, T_R$	reactor, feed, recycle stream temperatures
$T_{LM}$	logarithmic mean temperature difference
$T_M, T_S$	reactor temperature measurement, set point
$T_{WI}, T_{WO}$	cooling water inlet, outlet temperatures

$U$	heat transfer coefficient
$V$	volume of reacting mixture
$V_L, V_R, V_T$	valve stem positions
$X$	state variable

### Special Symbols

$\beta$	arc in SDG (ESDG)
$\delta$	qualitative Dirac delta function
$\Delta H_R$	heat of reaction
$\rho$	liquid density
$\tau$	residence time
[ ]	sign of expression
	absolute value of expression
•	qualitative multiplication

### Subscripts

$B$	bypass stream
$C$	cold (cooling) stream
$H$	hot stream
$I$	inlet
$i,j,k,\dots$	dummy variables
$O$	outlet
$P$	product stream
$R$	recycle stream
$S$	steady-state value
$W$	cooling water stream

## 2.7. References for Chapter 2

1. Kuipers, B.J. (1984). Common sense reasoning about causality: deriving behavior from structure. *Artificial Intelligence* 24, 169-203.
2. Fink, P.K. (1985). Control and integration of diverse knowledge in a diagnostic expert system. *Proc. 9th IJCAI* 426-431.

3. Bobrow, D.G. (Ed.) (1985). Qualitative Reasoning about Physical Systems. MIT Press, Cambridge, Massachusetts.
4. de Kleer, J., and J.S. Brown (1984) A qualitative physics based on confluences. Artificial Intelligence 24, 7-83.
5. Pan, J.Y. (1984). Qualitative reasoning with deep-level mechanism models for diagnoses of mechanism failures. Proc. IEEE 1st Conf. on AI Appl. 295-301.
6. Kuipers, B.J. (1986). Qualitative Simulation. Artificial Intelligence 29, 289-338.
7. Forbus, D.F. (1984). Qualitative process theory. Artificial Intelligence 24, 85-168.
8. Kuipers, B.J. (1985). The limits of qualitative simulation. Proc. 9th IJCAI 128-136.
9. Williams, B.C. (1984). Qualitative Analysis of MOS Circuits. Artificial Intelligence 24, 281-346.
10. de Kleer, J. (1984). How circuits work. Artificial Intelligence 24, 205-280.
11. Dormoy, J.L. (1988). Controlling qualitative resolution. Proc 7th Natl. Conf. on A.I. (AAAI-88), 319-323.
12. Iwasaki, Y. and H.A. Simon (1986). Causality in device behavior. Artificial Intelligence 29, 3-32.
13. Iri, M., K. Aoki, E. O'Shima and H. Matsuyama (1979). An algorithm for diagnosis of system failures in the chemical process. Comput. & Chem. Eng. 3, 489-493.
14. Palowitch, B.L. (1987). Fault Diagnosis of Process Plants Using Causal Models. Sc.D. Thesis, Massachusetts Institute of Technology.

15. de Kleer, J. (1979). The origin and resolution of ambiguities in causal arguments. Proc. 6th IJCAI 197-203.
16. Rasmussen, J. (1980). Models of mental strategies in process plant diagnosis. Human Detection and Diagnosis of System Failures. NATO Symp., Roskilde, Denmark, J. Rasmussen and W.B. Rouse (Eds). Plenum, New York.
17. Raiman, O. (1986). Order of magnitude reasoning. Proc. 5th Natl. Conf. on A.I. (AAAI-86), 100-104.
18. Mavrovouniotis, M.L. and G. Stephanopoulos (1987). Reasoning with orders of magnitude and approximate relations. Proc. 6th Natl. Conf. on A.I. (AAAI-87), 626-630.
19. Finch, F.E. and M.A. Kramer (1989). The handling of dynamics, multiple faults, and out-of-order alarms in the MIDAS diagnosis system. A.I.Ch.E. Spring National Meeting, Houston, TX.
20. Rasmussen, J. (1985). The role of hierarchical knowledge representation in decision making and system management. IEEE Trans. Sys. Man Cyber. SMC-15, 234.
21. Finch, F.E. and M.A. Kramer (1988). Narrowing diagnostic focus using functional decomposition. AIChE J. 34, 25-36.
22. Shum, S.K. and J.F. Davis (1986). An expert system for diagnosing process plant malfunctions. IFAC Workshop on Fault Detection and Safety in Chemical Plants, Kyoto, Japan.

### 3. Causal Modeling of Continuous Processes

Causality is the principle that effects must be propagated locally within the topology of a system. Several representations of causality have appeared previously. Of these representations, representations based on the concept of the Signed Directed Graph have proven most useful for diagnostic applications.

One important limitation<sup>1</sup> of the single-stage Signed Directed Graph (SDG) is that certain behaviors resulting from global interactions in the process are not apparent from local SDG models. An example of such behaviors is the passage of the effect of a disturbance to the manipulated variable of a control loop, without an apparent deviation of the controlled variable. In this chapter, rigorous criteria for identifying all such "non-local" behaviors are derived from a **global** analysis of the SDG. We develop a novel causal representation, the Extended Signed Directed Graph (ESDG), which accounts for these behaviors. The ESDG is suggested as a model of process behavior for the diagnosis of dynamic processes.

#### 3.1. Review of Previous Work on Causal Modeling

Within certain limits, human beings understand and can reason about the behavior of physical processes without requiring detailed numerical analysis. This skill is known to play a major role in disturbance analysis and fault diagnosis by process operators [1]. (Rasmussen, 1980). Although the mechanisms of human reasoning about physical processes are complex and not well understood, cause-and-effect is clearly a basic mechanism of human reasoning about system behavior [2].

Causal models generally consist of a representation over which disturbances are allowed to propagate under a certain set of propagation rules. These rules reflect, in one way or another, the basic causal principle that effects must be propagated locally within the topology of a system [2].

Previous research on causal modeling of continuous physical systems has been performed in the fields of artificial intelligence (A.I.) and chemical systems engineering. Modeling of

---

<sup>1</sup>This limitation is due to restricting disturbance propagation to feedforward paths in the SDG.

causality in previous works in A.I. has been problematic. Forbus [3] dismisses causality as "mainly a tool for assigning credit to hypotheses for observed or postulated behavior." de Kleer & Brown [4] use a concept of causality that describes transients during mythical (infinitesimal) time instants when non-causal confluences are not valid. Causality is not represented explicitly but is discovered by a set of heuristics during constraint propagation among confluences. Iwasaki & Simon [5] present a theory of causality in which causality is construed as output set assignment [6] of a set of simultaneous algebraic equations. Starting with assumed exogenous variables and the non-causal model of the process, they attempt to deduce local causality using precedence ordering which provides the order by which subsets of variables can be solved for by direct substitution. This procedure is incomplete at best as it does not provide causal ordering among variables in feedback loops.

In chemical systems engineering, various causal models have been proposed by previous authors [7, 8, 9, 10, 11, 12 & 13]. Most of these models are based on the concept of the directed graph. Iri et al. [8], introduce the single-stage signed directed graph (SDG). This representation provides explicit causal relationships among all variables (including those in feedback loops). Umeda et al. [9], extend this representation to the multiple stages to handle dynamic behaviors. Shiozaki et al. [11], and Tsuge et al. [12], extend the possible values of variables in the SDG to a five range pattern (+, +?, 0, -?, and -) to deal with states that are outside the range of normal process variability but still within predetermined threshold limits. The Lapp & Powers [7], and Palowitch [13] digraphs are essentially similar to Iri's.

The explicit representations of causality provided by digraphs have proven most useful for diagnostic applications and are analyzed further in this chapter. Two desirable properties of digraphs are:

- Detailed numerical information is not required in their construction, and a formal approach can be used to derive the models from domain principles governing the process.
- They provide explanations similar to causal arguments humans use in diagnosing and reasoning about process behavior

Diagnosing the failure origin using digraphs is equivalent to searching for nodes that can consistently account for deviations of all other process variables [8, 13, & 14].

An important feature that distinguishes chemical process systems from many other physical systems is the existence of complex global topologies with:

- recycle and bypass flows of material and energy,
- feedback and feedforward transmission of information.

These are reflected as feedback and feedforward paths in digraphs and result in complex process dynamics.

Previous work on causal digraphs has not adequately addressed the role of feedback and complex dynamics. In order to reduce the ambiguity inherent in qualitative modeling O'Shima [15], limits disturbance propagation to feedforward paths in the SDG. This assumption excludes process behavior in negative feedback control loops, where the effect of a disturbance is passed to the manipulated variable of a controller, without an apparent deviation of the controlled variable. Therefore, the diagnosis obtained by limiting propagation to feedforward paths excludes the correct failure origin in situations where controllers compensate for effects of disturbances on controlled variables. Recognizing that feedback control is ubiquitous in chemical process systems, Iri et al. [8], adopt a heuristic that makes exceptions for this type of apparent "global" causality. Similar exceptions have been made by Tsuge et al. [16] and Ulerich and Powers [17]. However, we show that these heuristics are incomplete as they do not adequately account for all global behavior.

In this chapter, a global qualitative analysis is presented. The analysis shows that the disturbance propagation assumptions implicit in Iri's work, predict the initial response of the system to a disturbance, but exclude the ultimate (steady state) response in other systems which exhibit types of complex dynamics, namely, inverse or compensatory responses.<sup>2</sup> We derive criteria related to the global topology of the SDG under which the SDG excludes ultimate process behavior. The key to the SDG excluding the ultimate

---

<sup>2</sup>Inverse response occurs when the initial and ultimate direction of change of a variable are opposite. Compensatory response is exhibited, if the variable returns to its initial value after all transients have died out.

response of the system is the location of positive feedback and integrating effects within negative feedback loops in the system. From these criteria an alternative causal model of the process, the Extended Signed Directed Graph (ESDG) is developed. The ESDG is similar to the SDG but includes non-physical arcs that systematically account for complex dynamics.

In the following, the SDG is introduced and the limitations of Iri's propagation assumptions are pointed out using a simple process as an example (Section 3.2). In Section 3.3, a global qualitative analysis is carried out and topological criteria for violation of these assumptions are derived in systems where transitions across qualitative regimes do not occur. In Section 3.4, the ESDG is introduced. In Section 3.5, we extend the topological criteria derived in Section 3.3 to situations where transitions across qualitative regimes<sup>3</sup> take place. Finally, the use of the ESDG as a dynamic process model is suggested in Section 3.6.

## 3.2. The Single-Stage Signed Directed Graph

The SDG is causal model of a process consisting of nodes, symbolizing process variables (parameters), and signed directed arcs representing immediate local cause and effect relationships between variables. Nodes in the SDG assume the qualitative values (0), (+), and (-), representing the nominal steady state, higher and lower than the nominal steady state, respectively. Arc signs + and - indicate whether values of the cause and effect variables tend to change in the same or opposite directions.<sup>4</sup> In this section, we analyze previous disturbance propagation assumptions for predicting process behavior from the SDG.

### 3.2.1. Derivation of the SDG

The SDG of a process can be obtained from either [8]:

- Operating data and experienced operators.

---

<sup>3</sup>A qualitative regime is a region of process behavior where all arcs of the digraph maintain their signs.

<sup>4</sup>Other attributes of the SDG are discussed in Chapter 4.

- A mathematical model of the process.

When a model is available, deriving the SDG from a mathematical model is clearly superior. Deriving the SDG from operating data will produce a model that contains only observed nodes and arcs that represent both direct and indirect causal influences. Therefore, a change in process instrumentation may require radical modification of the SDG. In addition, the operators' experience may be inadequate and as a result the SDG may not represent process behavior for previously unexperienced failures. Moreover, the approach is unstructured and may lead to an inconsistent representation of the process.

Derivation of the SDG is discussed in detail in Chapter 4, and only a summary is presented here. The SDG is derived from a dynamic process model, represented as a set of algebraic, ordinary differential and partial differential equations. For differential equations with explicit time dependence, causal arcs can be derived directly, as explained below. Algebraic equations that represent pseudo-steady state approximations of dynamic equations if possible should be restored to differential equation form. Other algebraic equations have no explicit directionality and thus the causal relationships among variables cannot be unambiguously determined based solely on the structure of the equation. Knowledge of the origin of the equation, the underlying processes, device physics and context must be utilized to determine this directionality. For example, the direction of causality between flow ( $F$ ) and valve stem position ( $V_s$ ) derived from the algebraic equation describing a (correctly working) control valve,

$$F - f^+(V_s, \Delta P) = 0 \quad (3.1)$$

(where  $f^+$  is a monotonically increasing function), is from  $V_s$  to  $F$  and not vice-versa.

The direction of causality between variables from ordinary differential equations,

$$\frac{dx}{dt} = f(x, u, p); \quad x = x_0, \quad u = u_0, \quad p = p_0 \quad (3.2)$$

is from the right to the left hand side of the equation. If  $m$  is the order of the first non-zero partial derivative,  $\partial^m f_j / \partial x_j^m$ , evaluated at the nominal steady state of eq. (3.2), the sign  $[\beta_{ij}]$  of the SDG arc  $\beta_{ij}$ , ( $i \neq j$ ), originating at  $x_j$  and ending at  $x_i$  is:

- (i)  $[\partial^m f_i / \partial x_j^m]$ , for m odd.
- (ii)  $[\partial^m f_i / \partial x_j^m] \cdot [dx_j]$ , for m even, where  $[dx_j]$  is the direction of deviation of  $x_j$  from its nominal steady state value.

Analogous rules govern arcs between  $u_i$  and  $x_i$ , and  $p_i$  and  $x_i$ . Self-cycles ( $i=j$ ) follow the same sign conventions, but are usually not depicted in the SDG.<sup>5</sup> Inequalities,  $\mathcal{L}$ , for determining the range of validity of the SDG are defined similarly to those for confluences in Section 2.2.1.

Partial differential equations can be reduced to ordinary differential equations by the method of lines or a similar method, thus the SDG can be derived for distributed systems according to eq. (3.2).

### 3.2.2. Determining System Behavior from the SDG

In simulating the effect of malfunctions on a process, it is assumed that the process is at steady state prior to fault initiation, so all nodes initially have the value 0. The immediate effect of a malfunction is to cause a deviation of a single process variable, input or parameter (the primary deviation variable) from its initial value. The deviation of the primary deviation variable is the source of all subsequent disturbances, determined by propagating the disturbance from the primary deviation variable to other nodes under a fixed set of propagation assumptions.

Ideally, propagation assumptions should be chosen so that the interpretations include the complete set of actual behaviors for all members of the system's qualitative class (completeness), and as few spurious behaviors as possible (minimality). Because of the duality between simulation and diagnosis, completeness is crucial as it guarantees the inclusion of the failure origin, while minimality is desirable because it leads to improved diagnostic resolution.

The following definitions are useful in the following:

---

<sup>5</sup>In Section 3.3.2, we show that the signs of self-cycles are required to locate variables whose ultimate response may be excluded by the SDG.

- A path from an initial to a terminal node is a directed sequence of nodes and arcs in the SDG or ESDG.
- An acyclic path is a path where all nodes (including initial and terminal) appear only once.
- A loop is a path that has the same the initial and terminal nodes, where each arc is traversed only once.
- A cycle is a loop in the SDG in which the initial and terminal nodes are the same and all other nodes in the loop are traversed only once.
- An event is any change of node sign.
- The influence of a node  $x_j$  on node  $x_i$  is the product  $[\beta_{ij}] \cdot [dx_j]$ .
- The net influence of a set of nodes  $x_j$  on a node  $x_i$  is the qualitative sum of the individual influences,  $\sum_j [\beta_{ij}] \cdot [dx_j]$ .
- The initial response is determined by the first non-zero sign assumed by each node. There may be more than one possible initial response.
- The ultimate or steady state response is any reachable state where each node with an unambiguous net influence has the same sign as the influence. There may be multiple possible ultimate responses.

The following two rules serve to generate sequences of events locally consistent with causality:

*Rule 1.* Any node can change its sign provided that the new sign equals the net influence on the node,<sup>6</sup> and

*Rule 2.* Direct transitions from (+) to (-) and (-) to (+) are forbidden.

---

<sup>6</sup>Note that the ambiguous value, ?, matches any node sign.

These rules imply that if an event occurs at  $x_i$ , the sign change will be in the direction of a non-zero input influence, or if all influences are zero,  $x_i$  can only return to normal. Expressed mathematically, when node  $x_i$  undergoes a sign change:

$$([dx_i]_k - [dx_i]_{k-1}) \cdot [\beta_{ij}] \cdot [dx_j] = (+), \text{ for some } j,$$

$$\text{or } [dx_i]_k = 0 \text{ and } [\beta_{im}] \cdot [dx_m] = 0, \text{ for all } m, \quad (3.3)$$

where  $[dx_i]_{k-1}$  is the old node sign and  $[dx_i]_k$  is the new node sign. The influences consistent with the direction of change of  $x_i$  are the possible causes of the event, that is, any node  $x_j$  input to  $x_i$  for which eq. (3.3) is satisfied.

Interpretations generated by eq. (3.3) includes all cases of inverse and compensatory response associated with feedback loops. Since inverse and compensatory responses are exhibited by relatively few process systems outside of control loops, Rules 1 and 2 can lead to many spurious interpretations.<sup>7</sup> For example, high temperature  $[dT] = (+)$  on the shell side of a heat exchanger will cause increased rate of heat flow to the tube side,  $[dQ] = (+)$ . Causally,  $[dQ] = (+)$  has a negative influence on  $T$ . According to eq. (3.3), transitions to the states  $[dT] = (0)$  and subsequently  $[dT] = (-)$  are permissible, however, these behaviors would not be observed.

To limit the production of spurious interpretations associated with feedback effects, an additional assumption regarding the behavior of feedback loops has been adopted by previous authors [15]. Put loosely, this assumption is that "an effect cannot compensate for its own cause". Thus,  $[dQ] = (+)$ , the effect, would not be allowed to override the cause,  $[dT] = (+)$ . More formally, using the definition of causality associated with eq. (3.3), a causal path can be defined as a ordered list of events beginning with the primary deviation, where each event causes the next event in the list. A **simple causal path** (SCP) can be traced from the primary deviation to any subsequent event without repeating a node. Therefore, SCPs can be mapped onto acyclic paths in the SDG. The heuristic "an event cannot compensate for its own cause" can be implemented by limitation of disturbance propagation to SCPs:

---

<sup>7</sup>The multi-stage signed directed graph, [9], gives the same result as interpreting the SDG under Rules 1 and 2.

*Rule 3.* A node is permitted to change its sign only if there is a simple causal path between the new event and the primary deviation.

In the previous example, the causal path  $[dT] = (+) \rightarrow [dQ] = (+) \rightarrow [dT] = (0)$  is not simple because the node T is repeated. A major reduction of spurious solutions is achieved through the SCP assumption, however, further measures are required to guarantee completeness. Rule 3 leads to exclusion of certain realizable responses when compensatory or inverse responses actually do occur. Compensatory responses occur by design in feedback control loops. Disturbances can appear to "jump over" the node representing the controlled variable, which remains normal. Therefore previous authors have devised special rules for interpreting the qualitative behavior of control loops.

For example, in their diagnostic algorithm, Iri et al. [8], adopt a heuristic that permits compensatory response for controlled variables in feedback loops, but otherwise require SCP. Formally, the **composite influence** of a node  $x_n$  on a node  $x_i$  is the product  $[\beta_{ij}].[\beta_{jk}]...[\beta_{mn}].x_n$ , where  $\beta_{ij}$ ,  $\beta_{jk}$ , ...  $\beta_{mn}$  are the arcs on an acyclic path between  $x_n$  and  $x_i$ . Defining the **external disturbance variable** to the controlled variable as the adjacent non-control loop variable to the controlled variable, Iri's heuristic can be implemented by the following:

*Rule 4.* A manipulated variable is allowed to change sign if the state of the controlled variable is normal (0), provided the new sign equals the composite influence from its external disturbance variable.

### 3.2.3. Limitation of Iri's Propagation Assumptions

While Rule 4 accounts for the above behavior in control loops, it leads to the exclusion of realizable inverse and compensatory responses in non-control negative feedback loops. For example, the response predicted by Rules 1 to 4 to a slow partial blockage in the outlet pipe of the gravity flow tank shown in Fig. 3.1a (increase in resistance  $R_s$  in Fig. 3.1b) is that the level increases while the outlet flow decreases. However, the outlet flow exhibits compensatory response and ultimately returns to the initial value (0) as shown in Fig. 3.1c. Because this compensatory response is not due to the action of controllers, Iri's method and its relatives [11 & 16] fail to diagnose this malfunction correctly if the negative

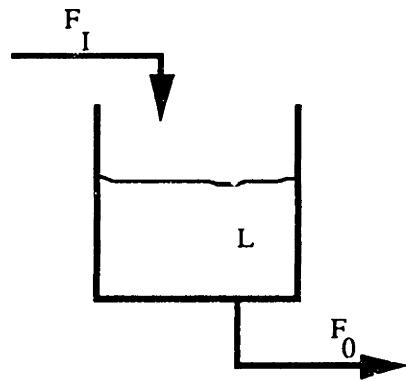


Fig 3.1a. Gravity Flow Tank

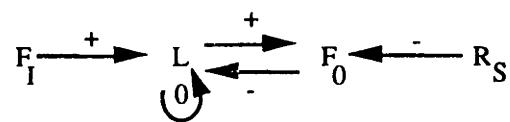


Fig 3.1b. SDG of Gravity Flow Tank

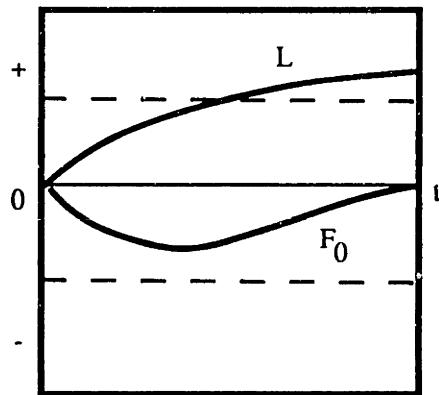


Fig 3.1c. Response to Outlet Blockage

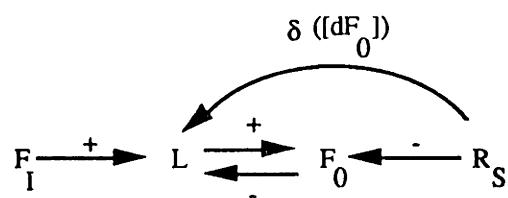


Fig 3.1d. ESDG of Gravity Flow Tank

deviation of outlet flow is sufficiently small (blockage sufficiently slow), despite the fact that latter methods use a five-range pattern.

In the following section, rigorous conditions under which the SCP and Iri's assumptions fail are presented. The analysis also shows that the behavior of controlled variables as described by Iri's heuristic is only a special case of the more general behavior of variables in negative feedback loops.

### 3.3. Determining System Behavior by Global Qualitative Analysis

In this section we determine the initial and ultimate behavior of systems by a global qualitative analysis and relate the behaviors to interpretations of the SDG obtained by using the SCP and Iri's disturbance propagation assumptions. Rigorous conditions under which the assumptions fail are derived. This analysis differs from that presented by Ishida et al. [18]. Their analysis attempts to relate global properties of the system such as stability and oscillatory behavior to the topology of the SDG. The qualitative criteria presented by Ishida et al. are weak and require numerical information for systems of order 2 or greater.

The following definitions are useful in relating system behavior to the topology of the SDG.

- **An inverse variable (IV)** is a variable that exhibits inverse response (IR) to a particular disturbance due to negative feedback effects.
- **A compensatory variable (CV)** is a variable that exhibits compensatory response (CR) to a particular disturbance due to negative feedback effects.

IVs can always display CR by varying parameter values within C. We restrict CVs to be variables that do not exhibit IR for any numerical realization of the qualitative class. We make the following additional definitions:

- The **complementary subsystem** to an acyclic path in the SDG is the subgraph that is obtained if all nodes in the acyclic path (including initial and terminal nodes) are eliminated.

- The **complementary subsystem** to a cycle in a subgraph of the SDG is the subgraph obtained if all nodes in the cycle are eliminated from the original subgraph.
- A **strongly connected component** (SCC) of the SDG, is a subgraph of the SDG for which a path exists from every node  $u$  to every node  $v$  (and from  $v$  to  $u$ ) in the SCC; and the SCC is not a proper subgraph of any other SCC [19].
- A **disturbance node** to a SCC is an adjacent node, not located in the SCC from which at least one arc initiates to a variable in the SCC.

Consider an  $n^{\text{th}}$  order system with state variables  $\underline{x}$  and input variables  $\underline{u}$  described by the general non-linear system of equations;

$$\frac{d\underline{x}}{dt} = \underline{f}(\underline{x}, \underline{u}); \quad \underline{x} = \underline{x}_0, \quad \underline{u} = \underline{u}_0 \quad (3.4)$$

Expanding each equation in eq. (3.4) in a Taylor series about the nominal steady state in terms of deviation variables:

$$\begin{aligned} \frac{d(x_i)}{dt} &= \sum_m \left( \frac{1}{m!} \cdot (dx_1 \partial / \partial x_1 + \dots + du_1 \partial / \partial u_1 + \dots)^m f_i \right. \\ &\quad \left. + R_{m+1}, i = 1 \dots n \right) \end{aligned} \quad (3.5)$$

Using the definition of SDG arcs in Section 3.2.1, eq. (3.5) is equivalent to:<sup>8</sup>

$$d(\underline{x})/dt = \mathbf{A} \cdot (\underline{x}) + \mathbf{B} \cdot (\underline{u}) \quad (3.6)$$

### 3.3.1. Initial System Behavior

In this Section, we show that the initial qualitative response of the system can be obtained by propagating through a subset of acyclic propagation paths in the SDG. Therefore, the initial response is necessarily included in the interpretations of the system as determined by propagation Rules 1 to 4.

---

<sup>8</sup>The underbar denoting matrix and vector quantities is substituted with bold characters throughout the rest of the chapter.

Considering a step change in an input variable  $u$ , the initial response must satisfy the linearized equations, eq. (3.6). Thus,

$$dx(t) = A^{-1}(e^{At} - I)b \cdot du \quad (3.7a)$$

which is equivalent to:

$$dx(t) = \sum_{n=1}^{\infty} A^{n-1} b t^n / n! \cdot du \quad (3.7b)$$

The qualitative response for each variable,  $[dx_i]$  is

$$[dx_i] = [b_i t + \sum_{j=1} a_{ij} b_j t^2 / 2! + \sum_{j=1} \sum_{k=1} a_{ik} a_{kj} b_j t^3 / 3! + \dots] \cdot [du] \quad (3.8a)$$

Equation (3.8a) can be factorized, and neglecting higher order terms, the initial qualitative response is

$$[dx_i] = [b_i t + \sum_{j \neq i} a_{ij} b_j t^2 / 2! + \sum_{k \neq j, i} \sum_{j \neq i} a_{ik} a_{kj} b_j t^3 / 3! + \dots] \cdot [du] \quad (3.8b)$$

The sign of each product  $b_i, a_{ij} b_j, \dots$  in eq. (3.8b) is the sign of an acyclic path of length  $r$  ( $r = 1, 2, \dots$ ) from  $u$  to  $x_i$  in the SDG. The summations cover all possible acyclic paths. Taking limits as  $t \rightarrow 0$ , the initial qualitative response of each variable is the qualitative sum of the effects obtained by propagation along the shortest non-zero acyclic paths in the SDG. Since the propagation paths determining the initial response are a subset of the paths that determine system behavior using Rules 1 to 3,<sup>9</sup> the initial qualitative response of the system is included in the set of interpretations produced by the SDG. In summary,

- Interpretations produced by propagating the effects of a disturbance through the SDG using the SCP (or Iri's) assumption are guaranteed to include the initial response of a system for all members of the system's qualitative class.

---

<sup>9</sup>This also includes propagation paths using Rule 4.

### 3.3.2. Ultimate System Behavior

In the previous section, it was shown that propagating the effects of disturbances using Iri's propagation assumptions is guaranteed to provide the initial response of the system. Thus, interpretations of the SDG may exclude the ultimate response of a system only in situations where a variable exhibits IR or CR. IR and CR are the net result of opposing causal effects that arise either from: (i) multiple acyclic (feedforward) paths to a node, (ii) negative feedback control loops, and (iii) other negative feedback loops. Rules 1 to 4 account for IR and CR arising from (i) and (ii). However, in cases where IR and CR arise from feedback effects in non-control loops (or when CR is caused by disturbances entering control loops at variables other than controlled variables), the propagation rules fail.

In this section, we derive necessary conditions for locating variables (IVs and CVs) whose ultimate values may be excluded by the SDG when using the SCP assumption. These conditions are derived using stability constraints and are related to the topology and signs of self-cycles of the SDG. We assume that the system is stable and thus attains an ultimate steady state. It is also assumed that there are no transitions to different qualitative regimes. The latter assumption is partially relaxed in Section 3.5.

The ultimate response to the system described by eq. (3.6) satisfies

$$\mathbf{A} \cdot (\mathbf{dx}) + \mathbf{B} \cdot (\mathbf{du}) = \mathbf{0} \quad (3.9)$$

Elements of  $\mathbf{A}$  and  $\mathbf{B}$  are mean values of the respective partial derivatives evaluated over process states traversed during the transient. If transitions to other qualitative regimes do not occur, the partial derivatives possess the same sign as SDG arcs and self-cycles at the nominal steady state. Solving eq. (3.9) we obtain

$$(\mathbf{dx}) = -(\mathbf{A})^{-1} \cdot \mathbf{B} \cdot (\mathbf{du}) \quad (3.10)$$

First we show that to determine the ultimate response of the system, it is sufficient to consider the response of variables in strongly connected components (SCCs) of the SDG with respect to perturbations in their respective disturbance variables. The SCC of a SDG is equivalent to an irreducible subsystem in the corresponding system ( $\mathbf{A}$ ) matrix [20]. By

a suitable permutation of rows and columns, a matrix with  $m$  irreducible subsystems (SCCs) can be partitioned into lower block triangular form [21]. Thus,

$$\begin{aligned} \mathbf{A} &= (A_{ij}) \quad ; i,j = 1,2,\dots,m \quad ; A_{ij} = \mathbf{0} \text{ for } j > i \\ \mathbf{B} &= (b_i) \quad ; i = 1,2,\dots,m \end{aligned} \quad (3.11)$$

Permuting  $\mathbf{x}$  appropriately, eq. (3.9) becomes

$$b_i \cdot (\mathbf{d}\mathbf{u}) + \sum_{j=1}^i A_{ij} \cdot (dx_j) = \mathbf{0} \quad ; i = 1,2,\dots,m \quad (3.12)$$

where each  $b_i$  is a row vector. Provided  $A_{ij}$  is non-zero, variables ( $x_j$ ) in SCC  $j$  are causally upstream of variables in SCC  $i$  ( $x_i$ ).

From eq. (3.12), the ultimate response of variables in each SCC is

$$(dx_i) = -(A_{ii})^{-1} \cdot (b_i \cdot (\mathbf{d}\mathbf{u}) + \sum_{j=1}^{i-1} A_{ij} \cdot (dx_j)) \quad ; i = 1,2,\dots,m \quad (3.13)$$

The ultimate response of variables in SCC  $i$  (eq. (3.13)), exists (and is bounded) provided the characteristic (eigen) values of the subsystem matrix,  $A_{ii}$  lies in the left half of the complex plane (i.e. the subsystem is stable) and values of inputs and variables in causally upstream SCCs are bounded. Since the characteristic values of the system matrix are related to those of the subsystem matrices according to;

$$\lambda = \lambda_1, \lambda_2, \dots, \lambda_m \quad (3.14)$$

stability of the system,  $\mathbf{A}$ , guarantees that solutions to eq. (3.13) exist.

Interpreting eq. (3.13), the ultimate response of variables in each SCC is the product of the subsystem matrix inverse and the sum of effects from input variables and disturbance variables from causally upstream SCCs. Equations (3.10) and (3.13) are of the same form, thus, to determine the response of variables in eq. (13):

- It is sufficient to superimpose the response to perturbations in individual disturbance variables of the SCC.

Thus, for each SCC we determine the response to perturbations in disturbance variables

$$(dx) = -(A)^{-1} \cdot b \cdot (du) \quad (3.15)$$

where  $x$ , is the vector of variables in the SCC and  $u$  a disturbance variable to the SCC.

We now state some formulae from matrix theory. The fundamental formula relating the determinant of an  $n \times n$  matrix to cycles in the matrix and its principal minors is [22]:

$$\begin{aligned} |A| &= a_{ii}|P_i| - \sum_{j \neq i} a_{ji}a_{ij}|P_{ij}| + \sum_{k \neq i,j} \sum_{j \neq i} a_{ji}a_{ik}a_{kj}|P_{ijk}| \\ &\quad + \dots + \sum_{p \neq \dots} \sum_{j \neq i} a_{ji}a_{ik}\dots a_{pj}(-1)^n|P_{ijk\dots p}| \\ &\quad + \sum_{q \neq \dots} \sum_{j \neq i} a_{ji}a_{ik}\dots a_{pq}a_{qj}(-1)^{n+1}|P_{ijk\dots pq}| \end{aligned} \quad (3.16)$$

where  $|P_{ij}|$  is the principal minor obtained by deleting elements in the  $i^{\text{th}}$  and  $j^{\text{th}}$  row and columns of  $A$ , etc. The upper summation limit is  $n$  and by definition, the minor obtained by deleting all elements of  $A$ ,  $|P_{ijk\dots pq}|$  is 1.

The determinant of a matrix can be expressed in terms of its first minors [23], according to

$$|A| = a_{ii}|P_i| + \sum_{j \neq i} (-1)^{i+j}a_{ji}|M_{ji}| \quad (3.17)$$

$|M_{ij}|$  is the minor obtained by deleting the  $j^{\text{th}}$  row and  $i^{\text{th}}$  column of  $A$ . From eqs. (3.16) and (3.17), we obtain

$$\begin{aligned} (-1)^{i+j}|M_{ji}| &= -a_{ij}|P_{ij}| + \sum_{k \neq i,j} a_{ik}a_{kj}|P_{ijk}| + \dots + \\ &\quad + \sum_{p \neq \dots} \sum_{k \neq i,j} a_{ik}\dots a_{pj}(-1)^n|P_{ijk\dots p}| \\ &\quad + \sum_{q \neq \dots} \sum_{k \neq i,j} a_{ik}\dots a_{pq}a_{qj}(-1)^{n+1}|P_{ijk\dots pq}| \end{aligned} \quad (3.18)$$

The characteristic equation of a matrix can be expressed in terms of its principal minors [23]

$$|\lambda I - A| = \lambda^n + \dots \sigma_k \lambda^k + \dots + (-1)^n |A| \quad (3.19)$$

where  $\sigma_k$ , ( $k = 1, \dots, n-1$ ) is  $(-1)^k$  multiplied by the sum of all principal minors of dimension  $k$ . From the Routh-Hurwitz criteria [24], a necessary condition for stability<sup>10</sup> is

$$(-1)^n |A| > 0 \quad (3.20)$$

From eq. (3.15), the ultimate response satisfies

$$(-1)^n |A| \cdot (dx) = -(-1)^n |A| \cdot A^{-1} b(du) = (-1)^{n-1} \cdot \text{Adj}(A) b(du) \quad (3.21)$$

where  $\text{Adj}(.)$  is the matrix adjoint.  $\text{Adj}(A)$  is the transpose of the matrix whose corresponding elements are cofactors of elements of  $A$ . Expanding eq. (3.21),

$$(-1)^n |A| \cdot (dx_i) = (-1)^{n-1} \cdot (b_i (-1)^{2i} |P_i| + \sum_{j \neq i} b_j (-1)^{i+j} |M_{ji}|) \cdot (du) \quad (3.22)$$

Applying the stability criterion (eq. (3.20)), the ultimate qualitative response of  $x_i$  is

$$[dx_i] = [(-1)^{n-1} \cdot (b_i (-1)^{2i} |P_i| + \sum_{j \neq i} b_j (-1)^{i+j} |M_{ji}|)] \cdot [du] \quad (3.23)$$

Substituting for  $(-1)^{i+j} |M_{ji}|$  using eq. (3.18),

$$\begin{aligned} [dx_i] = & [b_i (-1)^{n-1} |P_i| + \sum_{j \neq i} a_{ij} b_j (-1)^{n-2} |P_{ij}| \\ & + \sum_{k \neq i, j} \sum_{l \neq i} a_{ik} a_{kj} b_l (-1)^{n-3} |P_{ijk}| + \dots + \\ & + \sum_{p \neq \dots} \sum_{j \neq i} a_{ik} \dots a_{pj} b_j (-1)^{|P_{ijk\dots p}|} ] \cdot [du] \\ & + \sum_{q \neq \dots} \sum_{j \neq i} a_{ik} \dots a_{pq} a_{qj} b_j |P_{ijk\dots pq}| \cdot [du] \end{aligned} \quad (3.24)$$

---

<sup>10</sup>It can be shown from eq. (3.14) that  $(-1)^n |A| > 0$  for each SCC.

Each term in eq. (3.24) is the product of an acyclic path of length  $r$  ( $r = 1, \dots, n$ ) from  $u$  to  $x_i$  and its complementary principal signed minor of dimension  $n-r$ . The summations cover all possible acyclic paths. IR or CR is due to opposing effects, and a necessary condition for a variable to exhibit IR or CR due to negative feedback is:

- (1) The variable is located in a negative feedback loop or cycle (except self-cycles).

In addition, since the ultimate response is obtained by propagating through acyclic paths in the SDG, the ultimate response of a variable may be inconsistent with the SDG only if the sign of one of the signed minors in eq. (3.24) is not positive.

### *Inverse Response.*

For IR by  $x_i$ , the complementary signed principal minors (determinants) to at least one of the non-zero acyclic paths from  $u$ , above must be negative. Using eq. (3.16), a signed determinant of dimension  $r$  can be defined recursively in terms of signed determinants of its subsystems.

$$\begin{aligned}
 (-1)^r |H| = & -h_{ii}(-1)^{r-1} |Q_i| - \sum_{j \neq i} h_{ji}h_{ij}(-1)^{r-2} |Q_{ij}| \\
 & - \sum_{k \neq i,j} \sum_{l \neq i} h_{ji}h_{ik}h_{kj}(-1)^{r-3} |Q_{ijk}| \\
 & - \sum_{p \neq \dots} \sum_{l \neq i} h_{ji}h_{ik}\dots h_{pj}(-1)^{|Q_{ijk\dots p}|} \\
 & - \sum_{q \neq \dots} \sum_{l \neq i} h_{ji}h_{ik}\dots h_{pq}h_{qj} |Q_{ijk\dots pq}|
 \end{aligned} \tag{3.25}$$

Note, by definition  $|Q_{ijk\dots pq}|$  is 1. This decomposition accounts for all cycles in the system.

The necessary condition for a signed determinant,  $(-1)^r |H|$ , to be negative is:

- (i)  $H$  should contain a positive cycle (or self-cycle).
- (ii) the complementary subsystem to one of the positive cycles in  $H$  should have a non-zero (signed) determinant.

Proof.

The proof is by induction.

Step 1. Show the conditions are true for systems of dimension 0, 1, 2.

Step 2. Assume the conditions are true for systems of dimension  $k = 0, 1, \dots, r-1$ .

Step 3. If can be shown that the conditions are true for a system of dimension  $r$ , provided it is true for systems of dimension  $k = 0, 1, \dots, r-1$ : then the condition must be universally true.

For  $r = 0, 1, 2$ , eq. (3.25) becomes

$$(-1)^{0|H|} = 1 \quad (3.26a)$$

$$(-1)^{1|H|} = -h_{11} \quad (3.26b)$$

$$(-1)^{2|H|} = h_{11}h_{22} - h_{12}h_{21} \quad (3.26c)$$

Clearly, when  $r = 1$  or  $2$ , the signed determinants cannot be negative unless the system contains a positive cycle. Also, for  $r = 2$ , if  $h_{12}h_{21}$  is zero<sup>11</sup>, and in addition the determinant of the complementary subsystem to a positive cycle  $h_{11}$  or  $h_{22}$  (i.e.  $h_{22}$  or  $h_{11}$ ) is zero, the signed determinant cannot be negative. If the necessary conditions are not met in all subsystems with dimension less than  $r$ , from Step 2, we assume all signed determinants in the RHS of eq. (3.25) are non-negative. From eq. (3.25), the signed determinant of dimension  $r$  cannot be negative unless the cycle  $h_{ji}h_{ik}\dots h_{pq}h_{qj}$  is positive or one of the positive cycles (length  $k < r$ ) has a complementary subsystem with a non-zero (positive) signed determinant. Thus the conditions are universally true.

---

<sup>11</sup>The complementary subsystem to a cycle of length  $r$  is the null system,  $\phi$ . By definition, the signed determinant of the null system is 1.

### *Compensatory Response.*

For CR by  $x_i$ , the coefficient of  $[du]$  in eq. (3.24) must be zero. Since in eq. (3.24),  $|P_{ijk\dots pq}|$  is 1, if a non-zero acyclic path of length  $n$  from  $u$  to  $x_i$  exists,  $x_i$  cannot exhibit CR. In addition to no acyclic paths of length  $n$ , the complementary signed principal minors (i.e. determinants of dimension 1 to  $n-1$ ) of all other non-zero acyclic paths must be zero for  $x_i$  to exhibit CR.

The **recursive** necessary and sufficient conditions for a determinant,  $|H|$  (dimension  $r \geq 1$ ) to be zero (i.e. rank deficient matrix) are:

- (i) there are no cycles of length  $r$ .
- (ii)  $H$  contains at least one zero self-cycle.
- (iii) all complementary submatrices,  $Q_i, \dots, Q_{ijk\dots p}$  (dimensions  $1 \leq k < r$ ) of non-zero cycles (excluding self cycles) in  $H$  satisfy conditions ((i), (ii) & (iii)) recursively.

These conditions are interpreted recursively. Start with system  $H$ . If (i) or (ii) is not satisfied then  $|H|$  is non-zero. If (i) and (ii) are satisfied and no non-zero cycles exist (iii) then the determinant is zero. Otherwise apply conditions (i), (ii) and (iii) to each complementary subsystem,  $Q$ , of non-zero cycles in  $H$ .

### Proof.

The proof is similar to the inductive argument used to prove the necessary conditions for negative signed determinants. From, eqs. (3.26b) and (3.26c), the conditions are necessary and sufficient for determinants of dimension 1 and 2. (Since the conditions are applied to an arbitrary index  $i$ , and  $H$  contains at least one zero self-cycle it is not necessary to consider complementary subsystems to non-zero self cycles in (iii)). If these conditions are satisfied in all subsystems with dimensions ( $1 \leq k < r$ ), from Step 2, we assume all determinants in the RHS of Eq. (17) are zero. From eq. (3.25), the signed determinant of dimension  $r$  cannot be zero unless there is no cycle of length  $r$ ,  $h_{ji}h_{ik}\dots h_{pq}h_{qj}$ , and thus the conditions must be universally true.

Necessary conditions for locating variables (IVs and CVs) whose ultimate values may be excluded by the SDG when using the SCP assumption are **summarized** below. First, in order to identify these variables, it is sufficient to consider the response of variables in SCCs with respect to perturbations in their respective disturbance variables.

The necessary conditions for a variable in a SCC to display IR due to negative feedback to perturbations in a disturbance variable are:

- (1) The IV is located in a negative feedback loop or cycle (except self-cycles).
- (2) The complementary subsystem to one of the acyclic paths from the disturbance variable to the IV should contain a positive cycle (or self-cycle).
- (3) The complementary subsystem to the positive cycle in one of the complementary subsystems in (2) should violate either (5a, 5b or 6) below.

Similarly, the recursive necessary and sufficient conditions for a variable in a SCC to display CR due to negative feedback to perturbations in a disturbance variable are:

- (4) The CV is located in a negative feedback loop or cycle (except self-cycles).
- (5) The complementary subsystems to all acyclic paths from the disturbance variable to the CV should each
  - (a) have at least one zero self-cycle (integrator) and
  - (b) not have a cycle containing all variables in the subsystem.
- (6) The complementary subsystems to all non-zero cycles (excluding self-cycles) in each complementary subsystem in (5) should each satisfy (5a, 5b and 6).

Necessary conditions (1-3) for IR result in a potential set of IVs with respect to perturbations in a disturbance variable. For a particular system, the actual set of IVs could be any of the subsets of this set. If the sign of all acyclic paths from the disturbance variable to each variable in the SCC is unique, some of these subsets can be eliminated by considering the individual steady state equations (eq. 3.9)). A subset of possible IVs is consistent only if the sign of the net influence (using ultimate values) on each variable in the

SCC is either ambiguous or opposite to the product of the sign of the variables ultimate value and its self-cycle

$$\sum_j [\beta_{ij}] \cdot [dx_j] = - [\partial f_i / \partial x_j] \cdot [dx_j] \quad ; i = 1, 2, \dots, n \quad (\text{steady state}) \quad (3.27)$$

We shall demonstrate the applicability of these criteria by considering a series of examples.

### *Example 1*

A first order irreversible ( $A \rightarrow B$ ) exothermic reaction takes place in a continuous stirred tank reactor (CSTR) with a cooling jacket. Reactor level and coolant temperature are assumed to be constant. The linearized dynamic process equations for this system are given in Froment & Bischoff, Sec. 10.4.b [25]. The SDG for this process is shown in Fig. 3.2a and predicts unambiguously that an increase in  $C_{A0}$  results in an increase in  $C_A$  and  $T$ .

The SDG has five SCCs ( $C_A T$ ), ( $C_{A0}$ ), ( $F$ ), ( $T_C$ ) and ( $T_0$ ). Of the SCCs, only ( $C_A T$ ) contains variables in a negative feedback loop. For a highly exothermic reaction, the cycle at  $T$  could be positive. For a perturbation in  $C_{A0}$ , there is only one acyclic path from  $C_{A0}$  to both  $C_A$  and  $T$ . The complementary subsystem to the acyclic path from  $C_{A0}$  to  $C_A$ , subsystem ( $T$ ), has positive feedback effects. Thus,  $C_A$  (only) could be an IV with respect to  $C_{A0}$ .

Quantitative parameters for a numerical realization of this system are given in Table 1. For these values, the diagonal element of  $A$  corresponding to  $T$  is  $+0.231\text{hr}^{-1}$ . Figure 3.2b, shows the response of the linearized system to a  $0.1\text{M}$  increase in  $C_{A0}$ .  $C_A$  increases from  $1\text{M}$  to a maximum value of  $1.014\text{M}$  at  $1.4$  hrs before decreasing to its ultimate value of  $0.973\text{M}$ , demonstrating inverse response as predicted. Hence the SDG excludes the ultimate response of this system. By a similar analysis it is deduced that  $C_A$  is also an inverse variable (for another numerical realization) with respect to perturbations in feed flowrate,  $F_0$ .

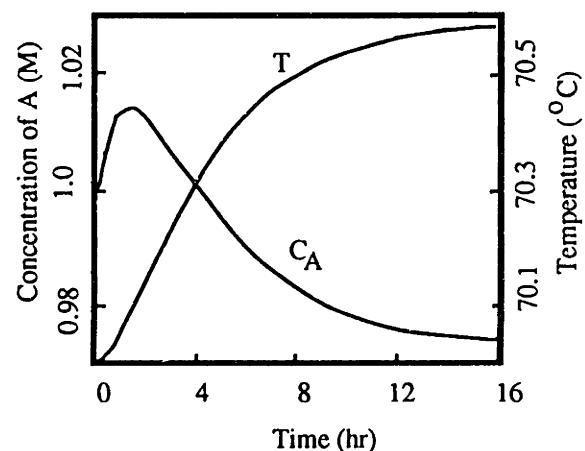
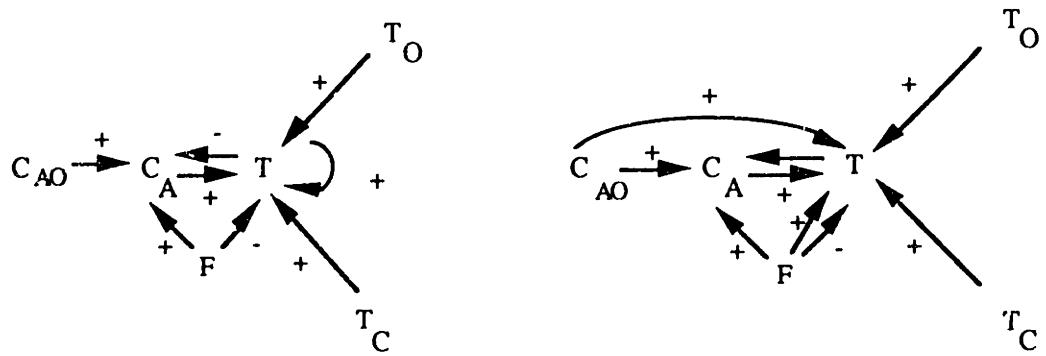


Fig. 3.2b. Response to High Feed Concentration

Table 3.1 Parameters for Continuous Stirred Tank Reactor in Example 1

$A_0 = 7.867 \times 10^{14} \text{ hr}^{-1}$	$T_0 = 55^\circ\text{C}$
$C_{A0} = 5M$	$U_A = 8.046 \times 10^2 \text{ KJ/hr}^\circ\text{C}$
$C_p = 4.184 \text{ KJ/Kg}^\circ\text{C}$	$V = 10 \text{ m}^3$
$E_a = 97.81 \text{ KJ/mol}$	$(-\Delta H_R) = 20.92 \text{ KJ/mol}$
$F = 2.5 \text{ m}^3/\text{hr}$	$\rho = 1000 \text{ Kg/m}^3$
$T_c = 5^\circ\text{C}$	

### *Example 2*

This example demonstrates the use of eq. (3.27) in reducing the number of consistent subsets of IVs. The SDG of an irreversible exothermic reaction taking place in a fluidized bed catalytic reactor (FBR) is shown in Fig. 3.3a. The SDG has five SCCs and only  $(C_{AG}, J, C_{Ap}, T_p, Q, T_G)$  contains variables in a negative feedback loop. For a highly exothermic reaction, the self-cycle at  $T_p$  can be positive. For a perturbation in  $C_{A0}$ , only one acyclic path from  $C_{A0}$  to each of these variables exists. The only positive cycle is the self-cycle at  $T_p$ . The complementary subsystems to the acyclic paths from  $C_{A0}$  to each of  $C_{AG}$ ,  $J$ ,  $C_{Ap}$ ,  $T_p$ ,  $Q$  and  $T_G$  are subsystems  $(J, C_{Ap}, T_p, Q, T_G)$ ,  $(C_{Ap}, T_p, Q, T_G)$ ,  $(T_p, Q, T_G)$ ,  $(Q, T_G)$ ,  $(T_G)$  and  $\phi$ , respectively ( $\phi$  is the null system). Only subsystems  $(J, C_{Ap}, T_p, Q, T_G)$ ,  $(C_{Ap}, T_p, Q, T_G)$  and  $(T_p, Q, T_G)$  contain positive cycles. Therefore  $C_{AG}$ ,  $J$  and  $C_{Ap}$  can be IVs with respect to perturbations in  $C_{A0}$ .

Seven proper subsets<sup>12</sup> can be formed from these IVs. Applying each of the six balances in eq. (3.27), five of these subsets violate at least one of these balances. For example, assuming  $(C_{AG}, J, C_{Ap})$  are all IVs, for a positive perturbation of  $C_{A0}$ ,  $C_{A0}(+)$ , the ultimate values of  $C_{AG}$  and  $J$  are  $C_{AG}(-)$  and  $J(-)$  respectively. These values do not satisfy eq. (3.28) below.

$$C_{AG} = C_{A0} - J \quad (3.28)$$

Thus subset  $(C_{AG}, J, C_{Ap})$  is inconsistent. The consistent subsets of IVs are  $(C_{AG}, C_{Ap})$  and  $(C_{AG})$ . Thus either  $C_{AG}$  and  $C_{Ap}$  or  $C_{AG}$  only are IVs with respect to perturbations in  $C_{A0}$ . Similarly  $T_G$  and  $Q$  can be IVs when  $T_O$  or  $T_W$  are perturbed. Applying eq. (3.27) leaves  $(Q)$  as the only consistent subset.

### *Example 3*

Applying these conditions to the SDG of the gravity flow tank, Fig. 3.1b, the SCCs of the SDG are  $(L, F_0)$ ,  $(F_I)$  and  $(R_S)$ . Of the SCCs only  $(L, F_0)$  contain variables in a negative feedback loop. The complementary subsystem to the acyclic path from disturbance variable  $R_S$  to  $F_0$  is  $(L)$ . There is a zero self-cycle on  $L$ , (Fig 3.1b) as level exhibits integrating

---

<sup>12</sup>A proper subset of a set of elements,  $S$ , is any subset of  $S$ , excluding the null set. If  $S$  has  $n$  elements, the number of proper subsets is  $2^n - 1$ .

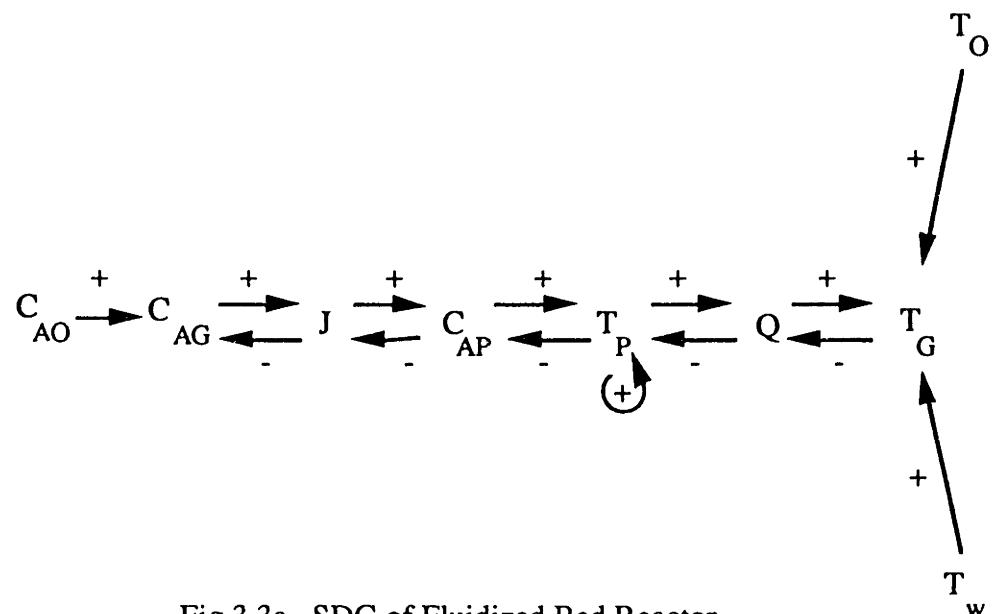


Fig 3.3a. SDG of Fluidized Bed Reactor

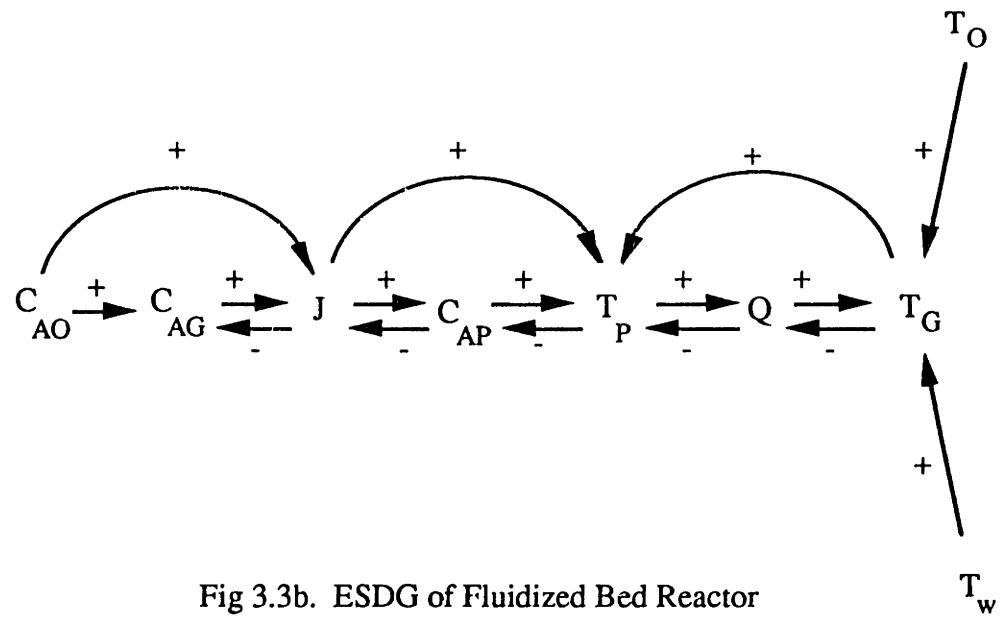


Fig 3.3b. ESDG of Fluidized Bed Reactor

effects. Within this subsystem, no non-zero cycles exist and thus  $F_0$  is a CV with respect to  $R_S$ . The criteria correctly predicts the ultimate response of the system for outlet blockage.

#### *Example 4*

One can also show that CR of controlled variables in feedback control loops is a special case of the more general behavior of CVs in negative feedback loops. Figure 3.4a, shows the SDG of a feedback control loop with PI control.  $X, X_C, X_m, X_s, X_v$ , and  $X_{sp}$  are the controlled variable, controller output, manipulated variable, controlled variable sensor, valve position, and set point, respectively.  $Y_i$  represent other nodes outside the loop. The SCC of interest is  $(X, X_s, X_C, X_v \text{ and } X_m)$  and all variables in the SCC are in a negative feedback loop, thus satisfying (4). The controller output node,  $X_C$  is associated with integral action and has a zero self-cycle. The disturbance variables to the SCC are  $Y_i$  and  $X_{sp}$ .

With respect to a perturbation in  $Y_3$ , there is only one acyclic path to each of the variables in the loop. The complementary subsystems to the acyclic paths from  $Y_3$  to each of  $X_m, X, X_s, X_C$ , and  $X_v$  are subsystems  $(X, X_s, X_C, X_v), (X_s, X_C, X_v), (X_C, X_v), (X_v)$  and  $\phi$ , respectively. Subsystems  $(X, X_s, X_C, X_v), (X_s, X_C, X_v)$  and  $(X_C, X_v)$  each contain  $X_C$  (5a). In addition, none of these subsystems contains a non-zero cycle (5b and 6). Thus  $X_m, X, X_s$  are all compensatory variables with respect to  $Y_3$ . Note that in this situation, where the disturbance enters the loop through the manipulated variable, the correct process behavior provided by the above analysis is **not accounted for** by the SCP assumption nor by Iri's "special rules" for control loops. The analysis also predicts that both  $X$  and  $X_s$  are compensatory variables with respect to perturbations in each of  $Y_1$  and  $Y_2$  and that there are no compensatory variables with respect to set point changes ( $X_{sp}$ ).

**Summarizing**, the ultimate response is excluded by the SDG, using the SCP or Iri's assumption when either of the following occur:

- A variable exhibits IR due to negative feedback.
- A variable exhibits CR in a non-control negative feedback loop.

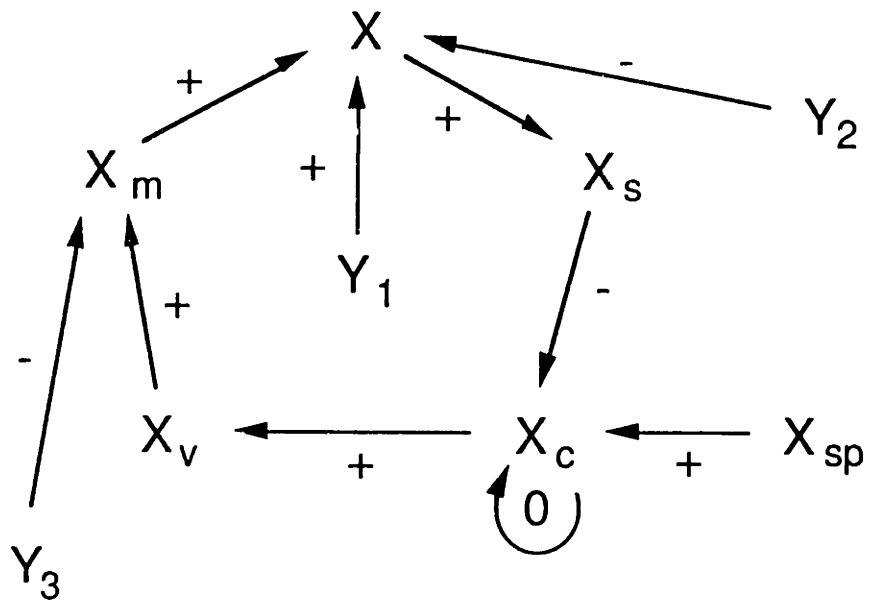


Fig. 3.4a. SDG of Control Loop

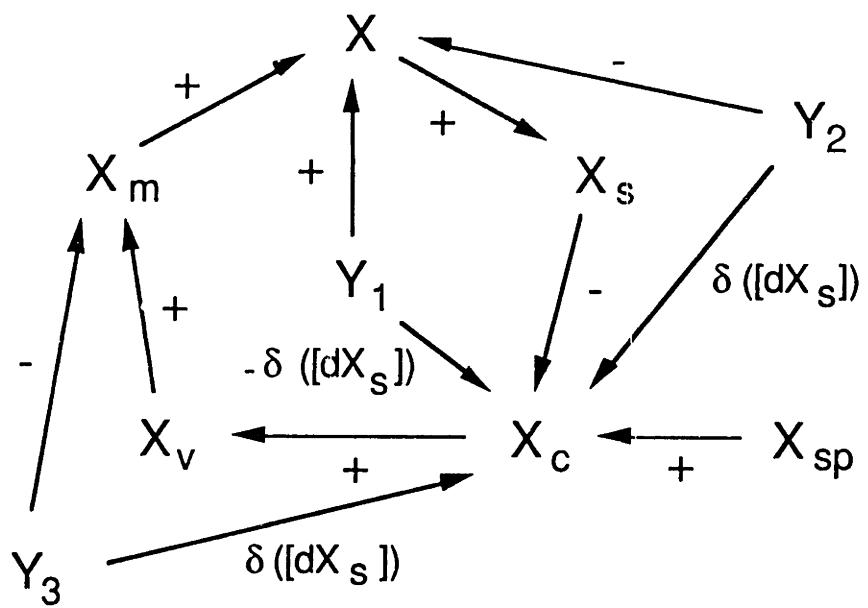


Fig. 3.4b. ESDG of Control Loop

- A disturbance enters a feedback control loop at a variable other than the controlled variable.

## 3.4. The Extended Signed Directed Graph

In the previous section, cases when interpretations obtained from the SDG (using the SCP and Iri's assumptions) may exclude the ultimate response of the system from stable steady states were identified. In this section, the SDG is modified to produce the Extended Signed Directed Graph (ESDG). The ESDG represents both local and "apparent" global causality and contains all nodes and arcs in the SDG and in addition, contains certain non-physical feedforward paths (arcs) that explain IR and CR in negative feedback loops. The ESDG is constructed so that propagation can be conducted under the SCP assumption without exceptions for control loops. The ESDG is guaranteed to produce interpretations which include the ultimate system response from stable steady states for all realizations of system parameters within the scope of its qualitative class.

### 3.4.1. ESDG Arcs for IR

Deriving IR arcs of the ESDG involves identifying IVs on each acyclic path from the disturbance variables based on the criteria of Section 3.3.2. On these acyclic paths, the IVs occur as one or more strings of adjacent variables. For each string of IVs on an acyclic path, ESDG arcs for IR originate at the node just upstream of the IVs and terminate at the non-inverse SCC variable located just downstream of the string. The sign of the arc is the product of the signs of the arcs on the forward path in the SDG between the origin and termination of the ESDG arc. In effect, the ESDG arcs "jump over" the IVs providing a SCP accounting for IR.

For the CSTR in Example 1 of Section 3.3.2.,  $C_A$  is an IV with respect to perturbations in  $C_{A0}$  and  $F$ . Two ESDG arcs are required (Fig. 3.2c), originating at  $C_{A0}$  and  $F$  and terminating at  $T$  (the non-inverse loop variable). Both arcs have positive signs since forward paths in the SDG from  $C_{A0}$  and  $F$  to  $T$  have net positive signs. These arcs enable the correct ultimate behavior of the system to be predicted using the SCP assumption.

Similarly, for the FBR in Example 2, either  $C_{AG}$  and  $C_{Ap}$  or  $C_{AG}$  only are IVs with respect to perturbations in  $C_{AO}$ . Also  $Q$  may be an IV when  $T_O$  or  $T_W$  are perturbed. Three ESDG arcs with positive signs are required (Fig. 3.3b). One originating from  $C_{AO}$  and terminating at  $J$ , another originating from  $J$  and terminating at  $T_p$  and the third originating from  $T_G$  and terminating at  $T_p$ .

### 3.4.2. ESDG Arcs for CR

The topology of ESDG arcs accounting for CR due to negative feedback is the same as for IR arcs. However, to prevent prediction of IR by the CVs, we assign to the ESDG arcs the sign  $\pm\delta([dx_j])$ , where  $\delta$  is the "qualitative Dirac delta function" (defined in Chapter 2) and  $x_j$  is the CV located just upstream of the termination of the ESDG arc. Only when  $[dx_j]$  is zero does the ESDG arc come into play, allowing the disturbance to "jump" over the CVs.

Considering the gravity flow tank,  $F_0$  is a CV with respect to  $R_S$  on the only acyclic path from  $R_S$ . Thus, one ESDG arc with sign  $+\delta([dF_0])$  from  $R_S$  to  $L$  (the non-compensatory variable) is required (Fig. 3.1d). The positive sign associated with the  $\delta$  function reflects the net sign of the path from  $R_S$  to  $L$  in the SDG.

Similarly, three ESDG arcs (Fig. 3.4b) are required for the example with the control loop, all terminating at  $X_C$ . An arc with sign  $-\delta([dX_S])$  from  $Y_1$ , one with sign  $+\delta([dX_S])$  from  $Y_2$  and another with sign  $+\delta([dX_S])$  from  $Y_3$ .

In conclusion, special treatment is not required for disturbance propagation in control loops when using the ESDG. After ESDG arcs are added to the SDG, disturbances can be assumed to propagate along SCPs only.

## 3.5. Transitions Across Qualitative Regimes

During a transient, the inequality conditions  $\mathcal{C}$ , defining arcs of the SDG may be violated, leading to a transition between qualitative regimes. These transitions can be modeled by a change in sign of one or more arcs in the SDG.<sup>13</sup> These transitions may be classified as:

### *Type A Transitions*

These occur when a **variable reaches physical bounds** at which system behavior changes. Examples are phase changes (e.g. boiling), zero gain events in control loops (such as control loop (valve) saturation) and complete blockages of conduits (zero flow). These transitions can be modeled by changes;  $+ \rightarrow 0$ ,  $- \rightarrow 0$ ,  $0 \rightarrow +$  and  $0 \rightarrow -$ .

### *Type B Transitions*

These occur when **driving forces** of phenomena such as momentum (pressure), heat (temperature) and mass (specie concentration) transfer change direction. These transitions can be represented by a combination of changes of arc signs;  $+ \rightarrow 0$ ,  $- \rightarrow 0$ ,  $0 \rightarrow +$ ,  $0 \rightarrow -$ ,  $+ \rightarrow -$  and  $- \rightarrow +$ . For example, a change in the direction of the pressure drop leads to reverse flow which is modeled by flow-pressure arcs in the SDG assuming opposite signs. Similarly, overall capacity effects on temperature (concentration) due to temperature (concentration) difference reversals such as in the mixing of two streams of unequal temperatures (concentrations) or in conduction in heat exchangers can be modeled by flow-temperature (concentration) arcs assuming opposite signs. In convection dominated flows arcs between temperature (concentration) nodes reverse direction. These can be modeled as two arcs undergoing transitions, one from  $+ \rightarrow 0$  and the other  $0 \rightarrow +$ .

In Section 3.3, it was assumed that transitions across qualitative regimes did not take place when identifying variables that exhibit IR or CR due to negative feedback. In this section, this assumption is partially relaxed and the ESDG is modified to include system response when transitions take place.

---

<sup>13</sup>Although the sign of self-cycles may change, by modeling self-cycles as cycles by the introduction of dummy variables, the sign change can be represented as a change in the sign of one of the arcs in the cycle.

### 3.5.1. Modifying the ESDG to Account for Process Behavior During Transitions

When a disturbance causes a system to undergo transitions to different qualitative regimes, the initial disturbance can be discretized for each qualitative regime the system passes through. From eq. (3.24), the ultimate response of the system from the initial conditions is given by

$$\begin{aligned}
 [dx_i] = & \sum Q.R. \left( [ b_i(-1)^{n-1} |P_i| + \sum_{j \neq i} a_{ij} b_j (-1)^{n-2} |P_{ij}| \right. \\
 & + \sum_{k \neq i,j} \sum_{j \neq i} a_{ik} a_{kj} b_j (-1)^{n-3} |P_{ijk}| + \dots + \\
 & + \sum_{p \neq \dots} \sum_{j \neq i} a_{ik} \dots a_{pj} b_j (-1) |P_{ijk\dots p}| \\
 & \left. + \sum_{q \neq \dots} \sum_{j \neq i} a_{ik} \dots a_{pq} a_{qj} b_j |P_{ijk\dots pq}| \right) Q.R. [du] Q.R. \quad (3.29)
 \end{aligned}$$

where the summation is over all qualitative regimes traversed during the transient.

From eq. (3.29), the ultimate response is the additive effect of the responses in the individual qualitative regimes. Transitions to other qualitative regimes may change the structure of the SDG, and results in a different set of ESDG arcs for each qualitative regime. Thus in principle, modifying the ESDG to account for transitions involves constructing the ESDG of the system for each qualitative regime and adding the **union** of non-physical ESDG<sup>14</sup> arcs to the SDG.

The number of qualitative regimes grows exponentially with the number of arc sign changes. However, most transitions take place only under very extreme conditions and all possible transitions do not need to be considered. The most important transitions of interest are zero gain events in control loops. These can be modeled by arc sign changes  $\rightarrow 0$  and  $- \rightarrow 0$ .

In the rest of this Section, we derive conditions for modifying the ESDG to account for zero gain events in control loops. The analysis presented can also be applied for other

---

<sup>14</sup>Conditions denoting the appropriate qualitative regimes may be represented on the ESDG arcs.

Type A transitions. We assume that arcs in the nominal qualitative regimes have signs + or -, and on transition, the signs of the relevant arcs become 0. Specifically we show that while the ESDG in the nominal regime accounts for the behavior of all CVs, some non IVs may become IVs, and additional arcs are required to explain their behavior.

### 3.5.1.1. MODIFICATION FOR TYPE A TRANSITIONS

Interpreting eq. (3.29), the ESDG of the system in its nominal qualitative regime may exclude the ultimate response of the system, only in the following situations:

- Variables in the system do not exhibit IR (CR) due to negative feedback in the nominal qualitative regime but may exhibit IR (CR)<sup>15</sup> due to negative feedback on a transition to another qualitative regime.

#### *Compensatory response*

Equation (3.29) is a summation of terms. Thus, a variable that undergoes a non-zero ultimate deviation in its nominal qualitative regime cannot undergo a (strictly) zero deviation on transition to a different qualitative regime. This leads to an important result.

- (1) A transition to a different Q.R. cannot cause a non CV in its nominal Q.R. to become a CV.<sup>16</sup>

Conversely, a transition may cause a CV to become a non CV. A common example is the behavior of controlled variables after control loop saturation. In the nominal Q.R., ESDG arcs derived in Section 3.4, initiate from a non CV on an acyclic path of the SDG, bypass a "string" of CVs and terminate at another non CV. These additional arcs will account for the behavior of all variables in the acyclic path unless a downstream variable in the "string" becomes a non CV on transition, while all its immediately upstream variables remain CVs.

We prove by contradiction, that a downstream CV in a string can become a non CV only if one of its input (upstream) variables on an acyclic path is a non CV.

---

<sup>15</sup>IR and CR are defined with the initial conditions in the nominal qualitative regimes as the reference state.

<sup>16</sup>This also applies to type B transitions.

1. Assume a situation where all inputs to a non CV are CVs.
2. At the new qualitative regime the signs of arcs terminating at the new non CV must still be + or -. Thus, if all inputs to the non CV are CVs, the ultimate values do not satisfy eq. (3.27). This results in a contradiction.

This leads to the following **important** conclusion:

- The ESDG of a system in its nominal qualitative regime accounts for the behavior of all CVs after Type A transitions.

### *Inverse response*

It can be deduced from eq. (3.29), that, an IV in its nominal qualitative regime may still be an IV after a transition. However, a non IV may become an IV, if the variable is an IV for one of the regimes traversed during the transient. Here, we develop conditions for which Type A transitions may cause a variable to become an IV.

Recalling the criteria for IVs in Section 3.3.2, it can be seen that a transition may cause a variable to become an IV only if the transition;

- (1) does not cause the sign of any of the arcs on one of the acyclic paths to the IV from the disturbance variable to change to 0,
- (2) does not cause the sign of any of the arcs in one of the positive cycles in the complementary subsystem to the acyclic path in (1) to change to 0, and
- (3) causes the value of the signed determinant of the complementary subsystem to the acyclic path in (1) and cycle in (2) to change from zero to non-zero (positive).

First, we show that in order for a Type A transition to cause the zero signed determinant of the complementary subsystem in (3), the subsystem, and thus the SCC must have fewer variables. Recalling the recursive formula relating the matrix determinant to cycles, eq. (3.16), an arc sign becoming zero cannot result in a determinant becoming non-zero, if the SCC retains the same variables. Clearly, if an arc sign becomes zero, the resulting SCC

cannot have more variables. Thus the determinant can only become non-zero if the transition causes variables to be removed from the SCC.

Next, we show that in order for a transition to cause a variable to exhibit IR, the "integrator" causing the determinant of the complementary subsystem to be zero must be removed from the SCC. For the determinant of the subsystem to be zero in its nominal qualitative regime, two possibilities exist:

- (a) none of the integrators in the subsystem are contained in cycles.
- (b) some of the integrators in the subsystem are contained in cycles.

In case (a), in order for the transitions to cause the determinant to be non-zero, it is necessary and sufficient for all integrators to be removed from the SCC. In case (b), it is necessary that all integrators not contained in cycles be removed from the SCC. Sufficient conditions involve recursive application of condition (3) in Section 3.3.2. From eq (3.19), it can be inferred that a subsystem with a zero determinant acts as an integrator. For the determinant of the complementary subsystem to be non-zero, the subsystem with the zero determinant must be removed from the SCC. Removing the zero determinant is analogous to removing an "integrator". Therefore, conditions for a non IV to exhibit IR as a result of a Type A transition can be interpreted as follows:

- An acyclic path from the disturbance to the variable still exists,
- A positive cycle in the complementary subsystem to the acyclic path still exists, and
- The "integrator" in the complementary subsystem to the acyclic path and positive cycle is removed from the SCC.

We illustrate these conditions for CVs and IVs with an example.

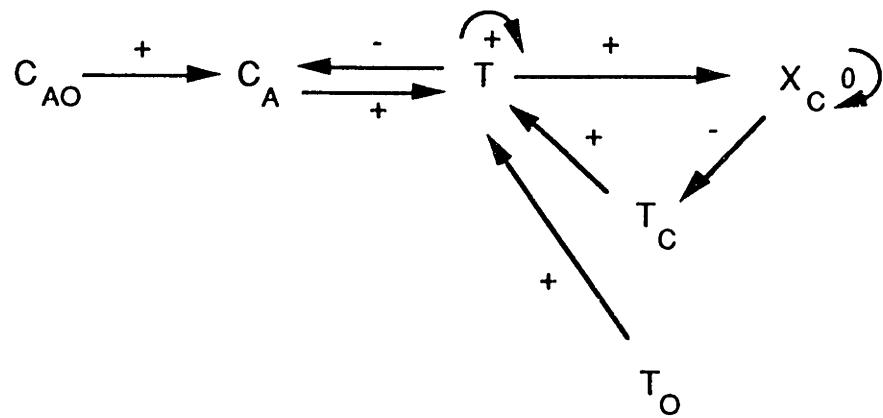


Fig. 3.5a. SDG of CSTR with Control System

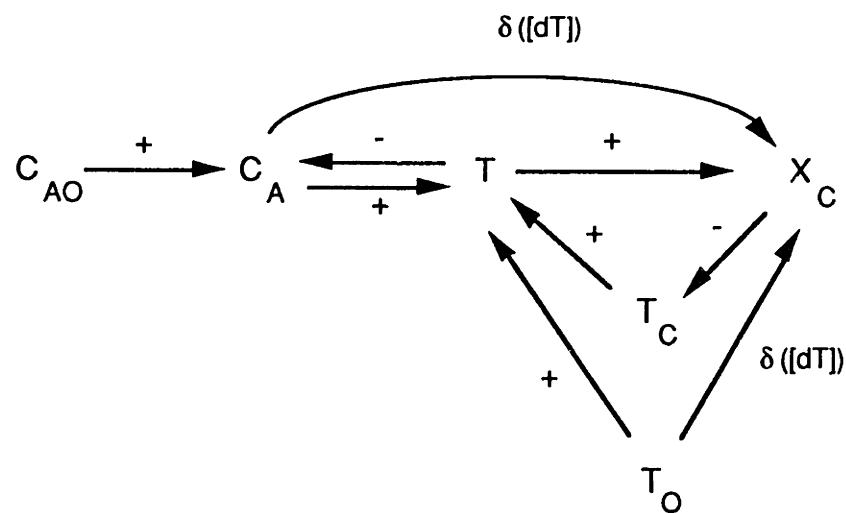


Fig. 3.5b ESDG of CSTR with Control System  
(Nominal Qualitative Regime)

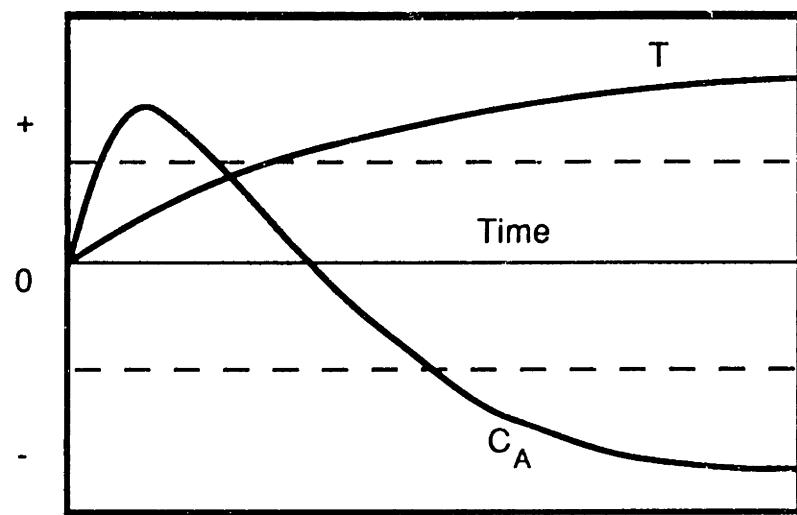


Fig. 3.5c. Response to High Feed Concentration  
(Saturated Control System)

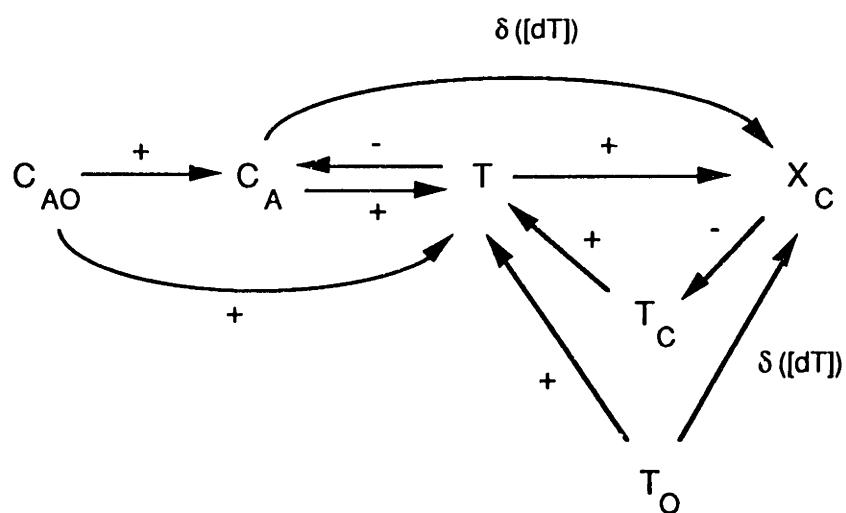


Fig. 3.5d. ESDG of CSTR with Control System  
(Saturated Control System)

### *Example*

Figure 3.5a shows the SDG for a first order irreversible ( $A \rightarrow B$ ) exothermic reaction taking place in a CSTR. The reactor temperature is controlled and the level is assumed to be constant. The ESDG for the nominal qualitative regime is shown in Fig. 3.5b. The response of the system for a large increase in  $C_{A0}$  is shown in Fig. 3.5c. During the transient the temperature control system becomes saturated and a transition to another qualitative regime takes place.

Nominally,  $T$  is a CV, but does not exhibit CR to the perturbation. On the other hand,  $C_A$ , although nominally a non IV exhibits IR. Although the ESDG accounts for the behavior of  $T$ , it cannot explain the behavior of  $C_A$ . Applying the conditions derived for Type A transitions, we observe that the removal of the integrator associated with the temperature controller from the subsystem with a positive self-cycle results in inverse response of  $C_A$ . The ESDG for the system with an additional arc that accounts for this possible transition is shown in Fig. 3.5d.

## **3.6. Using the ESDG as a Dynamic Process Model**

In previous sections, we introduced the ESDG as a process model representing both local and "apparent" global causality. On restricting disturbance propagation to SCPs, interpretations of the ESDG are guaranteed to include both the initial and ultimate behavior of the process. The ESDG may be used in conjunction with previous diagnostic algorithms [14], to locate the failure origin from abnormal process measurements.

Using the ESDG instead of the SDG as a process model for on-line diagnosis has the following potential benefits:

- Most importantly, it ensures that the correct failure origin is included in the set of malfunction candidates in situations when variables exhibit IR or CR due to negative feedback.
- Uniform search rules are used for all variables (including variables exhibiting IR or CR) resulting in more efficient real-time algorithms.

- Explicit precompiled knowledge about disturbance-inverse (compensatory) variable pairs is generated during ESDG construction. This information may be used to provide explanations.

To ensure an accurate diagnosis, the ESDG must account for all transient measurement patterns. Although interpretations of the ESDG may exclude transient patterns in certain situations, we suggest that the ESDG may be used as the basis for a dynamic process model. These situations are described below.

In order to use the ESDG for diagnosis, the numerical values of the process variables must be mapped into discrete qualitative values. For continuous processes, qualitative values are usually defined with respect to stationary reference points. Typically, a range of normal operation<sup>17</sup> is specified. Thus the qualitative value of a variable,  $x$ , is defined as

$$[dx] = '-' \text{ when } x < x_0^-$$

$$[dx] = '0' \text{ when } x_0^- \leq x \leq x_0^+$$

$$[dx] = '+' \text{ when } x_0^+ < x$$

where  $x_0^-$  and  $x_0^+$  are the lower and upper range end points respectively. Because the normal state is a range and not a point value, a process variable may appear to exhibit CR, even though the analysis in Section 3.3 indicates IR or CR are not valid behaviors. This results in the exclusion of the responses shown in Figs. 3.6a and 3.6b by the ESDG. For fixed end point values, there is a range of disturbance magnitudes for which the variable apparently exhibits CR. Above a certain magnitude, CR is not observed.

Other situations where responses are excluded are when a variable appears to exhibit IR, but the analysis in Section 3.3 indicates IR is not a valid behavior (Figs. 3.6c and 3.6d). This occurs if the width of the normal range of operation is too narrow. However it can be shown that no matter how wide the normal operating range is, a disturbance of a large enough magnitude will cause the variable to appear to exhibit IR.

---

<sup>17</sup>The analysis presented in Sections 3.3. and 3.5 is valid for a "point" range of normal operation.

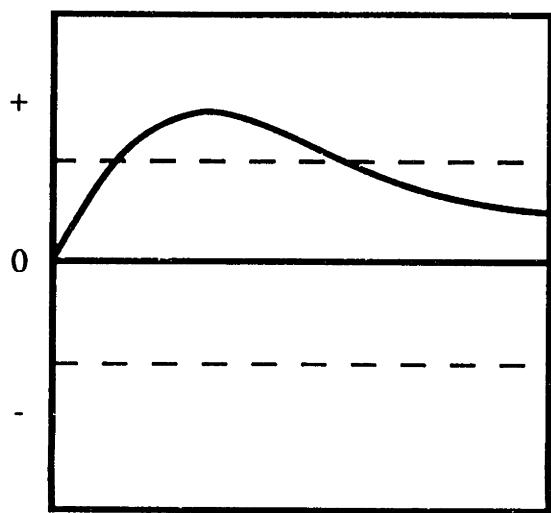


Fig 3.6a Apparent CR  
(overdamped response)

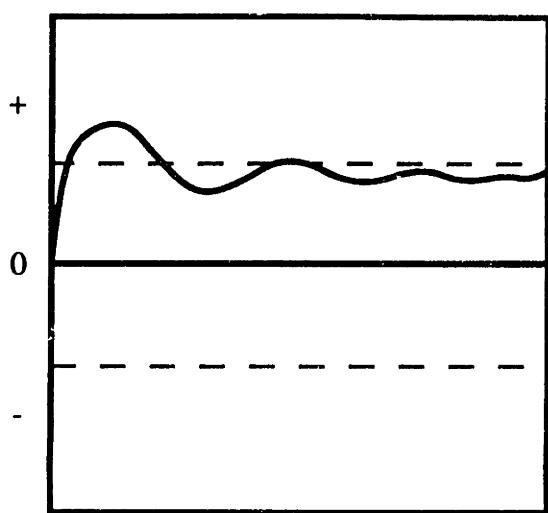


Fig 3.6b Apparent CR  
(underdamped response)

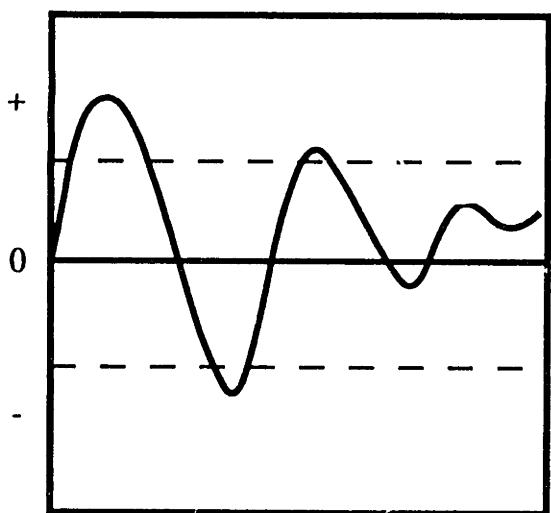


Fig. 3.6c. Apparent IR

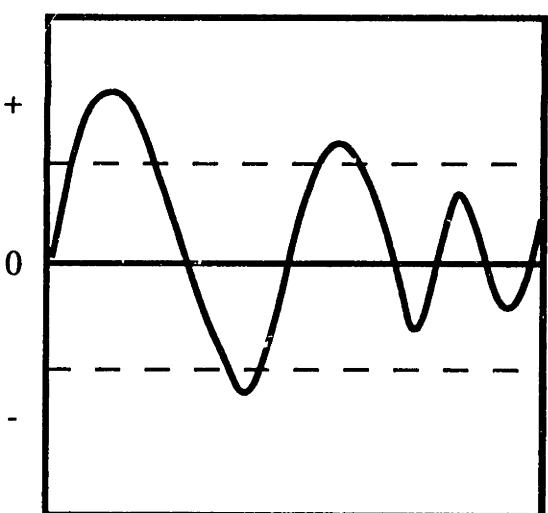


Fig. 3.6d. Apparent IR  
(CR)

One method of eliminating the apparent observation of IR shown in Figs. 3.6c and 3.6d, is to use non-stationary reference states. In stable overdamped (oscillatory) systems, the amplitude of oscillation decreases during the transient. The ratio of the amplitude between two successive peaks is defined as the decay ratio,  $D$ .<sup>18</sup> The value for the next end point,  $x_e$ , may be related to the value at the previous peak (or trough),  $x_m$ .

$$x_e = x_S - k \cdot (x_m - x_S) \cdot \sqrt{D} ; k \cdot \sqrt{D} < 1 , k > 1 \quad (3.30)$$

$x_S$  is the steady state value, and  $k$ , a constant that compensates for process non linearities, and  $D$ , an estimate of the decay ratio.

### 3.7. Conclusion

In this chapter, a causal model of continuous processes used in previous works, the SDG, was shown to exclude ultimate process response for systems that exhibit inverse or compensatory response due to negative feedback. Consequently, the SDG is an inappropriate model of qualitative behavior. To remedy the problem, necessary conditions based on the global topology of the SDG for the occurrence of the excluded behaviors were derived. Specifically, inverse response is caused by positive feedback effects in negative feedback loops, and compensatory response is caused by integral effects associated with negative feedback loops. On this basis, an alternative model of the process, the ESDG was derived. The ESDG is guaranteed to provide the full set of initial and ultimate process responses.

The use of the ESDG as a process model for use in diagnostic applications was suggested. However, certain transient responses could be excluded by the ESDG leading to inaccurate diagnosis. Determining situations where transient responses are excluded by the ESDG is a subject for future research. The use of non-stationary reference states to alleviate this problem was suggested. In Chapter 6, we use the ESDG as the basis of the Process Model for the MIDAS diagnostic system.

---

<sup>18</sup>The decay ratio has a value of about 0.25 in well designed control systems.

### 3.8. Notation for Chapter 3

A,B	chemical species, A,B
<b>A</b>	system (SCC sub) matrix
Adj	matrix adjoint
Ao	Arrhenius frequency factor
<b>b,B</b>	input vector, matrix
C <sub>A</sub>	concentration of A in reactor
C <sub>A0</sub>	feed concentration of A
C <sub>p</sub>	specific heat capacity
D	decay ratio
E <sub>a</sub>	activation energy
f <sup>+</sup>	strictly monotonic increasing, decreasing functions
F	volumetric flowrate
F <sub>0</sub>	reactor feed flowrate
<b>H</b>	subsystem matrix
<b>I</b>	Identity matrix
J	mass flux of chemical species
L	reactor, tank level
M	first minor of matrix
n	order of reaction
p	process parameter
P , Q	principal minor of system (subsystem) matrix
Q	heat flux
r <sub>A</sub>	reaction rate
R	remainder in Taylor's series expansion
R <sub>S</sub>	flow resistance in conduit
t	time
T,T <sub>C</sub> ,T <sub>0</sub>	reactor, coolant, feed temperatures
u	input variable
V	volume of reacting mixture
V <sub>s</sub>	valve stem position
x	system state variable

## Special Symbols

$\beta$	arc in SDG (ESDG)
$\delta$	qualitative Dirac delta function
$\phi$	null system
$\Delta H_R$	heat of reaction
$\Delta P$	pressure drop
$\rho$	liquid density
$\lambda$	matrix characteristic value (eigenvalue)
[ ]	sign of expression
•	qualitative multiplication

## Subscripts

c	controller output signal
I	inlet
i,j,k,...	dummy variables
G	gas stream
m	manipulated variable
p	catalyst particle
s	sensor value
S	steady-state value
sp	controlled variable set point
v	control valve
w	reactor wall

## 3.9. References for Chapter 3

1. Rasmussen, J. (1980). Models of mental strategies in process plant diagnosis. In J. Rasmussen and W.B. Rouse (Eds.), Human Detection and Diagnosis of System Failures. NATO Symposium Roskilde, Denmark. Plenum, New York. 241-258.
2. Bobrow, D.G. (1985). Qualitative Reasoning about Physical Systems: An Introduction. In D. Bobrow (Ed.), Qualitative Reasoning about Physical Systems MIT Press, Cambridge, Massachusetts.

3. Forbus, D.F. (1984). Qualitative process theory. Artificial Intelligence , 24, 85-168.
4. de Kleer, J. and J.S. Brown (1984). A qualitative physics based on confluences. Artificial Intelligence 24, 7-83.
5. Iwasaki, Y. and H.A. Simon (1986). Causality in device behavior. Artificial Intelligence 29, 3-32.
6. Steward, D.V. (1962). On an approach to techniques for the analysis of the structure of large systems of equations. Soc. Ind. Appl. Math. Rev., 4, 321-342.
7. Lapp, S.A. and G.J. Powers (1977). Computer-aided synthesis of fault trees. IEEE Trans. Reliability R-26, 2-13.
8. Iri, M., K. Aoki, E. O'Shima and H. Matsuyama (1979). An algorithm for diagnosis of system failures in the chemical process. Comput. & Chem. Eng. 3, 489-493.
9. Umeda, T., T. Kuriyama, E. O'Shima and H. Matsuyama (1980). A graphical approach to cause and effect analysis of chemical processing systems. Chem. Eng. Sci. 35, 2379-2388.
10. Kokawa, M. and S. Shingai (1982). Failure propagating simulation and nonfailure paths search in network systems. Automatica 18, 335-341.
11. Shiozaki, J., H. Matsuyama, K. Tano and E. O'Shima (1985). Fault diagnosis of chemical processes by the use of signed, directed graphs. Extension to five-range patterns of abnormality. Int. Chem. Eng. 25, 4, 651-659.
12. Tsuge, Y., J. Shiozaki, H. Matsuyama and E. O'Shima, (1985), Fault diagnosis algorithms based on the signed directed graph and its modifications. I. Chem. Eng. Symp. Ser. 92, 133-144.
13. Palowitch, B.L. (1987). Fault Diagnosis Using Causal Models Sc.D. Thesis, Massachusetts Institute of Technology.

14. Shiozaki, J., H. Matsuyama, E. O'Shima and M. Iri (1985). An improved algorithm for diagnosis of system failures in the chemical process. Comput. & Chem. Eng. 9, 285-293.
15. O'Shima, E. (1983). Computer aided plant operation. Comput. & Chem. Eng. 7, 311-329.
16. Tsuge, Y., J. Shiozaki, H. Matsuyama, E. O'Shima, Y. Iguchi, M. Fuchigami and H. Matsushita (1985). Feasibility study of a fault-diagnosis system for chemical plants. Int. Chem. Eng. 25, 4, 660-667.
17. Ulerich, N.H. and G.J. Powers (1988). On-line hazard aversion and fault diagnosis in chemical processes: The digraph + fault-tree method. IEEE Trans. on Reliab. 37, 2, 171-177.
18. Ishida, Y., N. Adachi and H. Tokumaru (1981). Some results on the qualitative theory of matrix. Trans of Soc. Instr. & Contr. Eng. 17, 1, 49.
19. Tarjan, R. (1972). Depth-first search and linear graph algorithms. SIAM J. Comput. 1, 2, 146-160.
20. Varga, J.S. (1962). Matrix Iterative Analysis, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.
21. Barnett, S., and C. Storey (1970). Matrix Methods in Stability Theory, Barnes and Noble, Inc., New York.
22. Basset, L., J. Maybee and J. Quirk (1968). Qualitative economics and the scope of the correspondence principle. Econometrica, 36, 3-4, 544-563.
23. Gantmacher, F.R. (1959). The Theory of Matrices, vols. I, II, Chelsea, New York.
24. Routh, E.J. (1955). Dynamics of a System of Rigid Bodies, Dover, New York.

25. Froment, G.F. and K.B. Bischoff (1979). Chemical Reactor Analysis and Design, Wiley, New York.

## 4. Guidelines for Developing SDG Models

In Chapter 3, a causal model of continuous process systems was introduced. The model, the Extended Signed Directed Graph (ESDG) represents both local and "apparent global" causal influences between variables in the process. The global influences were derived from a topological analysis of the process Single-Stage Signed Directed Graph (SDG).

In this chapter, guidelines for deriving the SDG of continuous process systems are presented. The SDG consists of variables, symbolizing process variables and parameters, and signed directed arcs representing immediate local causal influences between variables. Variables assume the qualitative values (0), (+), or (-), representing the nominal steady state, higher or lower than the nominal steady state, respectively. Arc signs, indicate the effect of the values of the cause variables on the direction of change of values of effect variables. The effect of malfunctions (root causes) on variables is also represented in the SDG. The process SDG is constructed bearing in mind that the ESDG will be used to produce the Process Model (PM) used for diagnosis in the MIDAS diagnostic system.

### 4.1. Introduction

Continuous chemical processes, like all physical systems, are engineered from sets of components which can be connected in a variety of ways to perform the intended overall function. The behavior of these components are largely context independent. Thus, a component can be modeled independently of the behavior of its adjacent components or the encompassing process. A **modular** approach is reasonable if the process is decomposable and several units are repeated within the topography of the process. Consequently, we adopt a modular approach when developing the SDG of a process. Therefore, the first step in developing the SDG of a process is to identify its constituent units and then to create SDG models for these units. The unit SDG models are then linked together to form the process SDG.

As was discussed in Chapter 3, the preferred approach to developing the SDG is from the engineering principles governing the domain. A lot of work has been done on conventional mathematical modeling of physical systems using these principles. Therefore, our approach is to derive the SDG from numerical models developed from engineering principles.

Different sets of rules and procedures for deriving the SDG from conventional differential and algebraic equations describing the process have been presented previously [1, 2 & 3]. These approaches are limited and do not address all SDG attributes listed in Section 4.2. In particular, self-cycles associated with SDG variables and disabling conditions associated with causal arcs are not specified. Because self-cycles are not defined, SDGs derived by previous approaches cannot be used to derive the ESDG of the process. Additionally, procedures for specifying SDG arcs derived from algebraic equations are not entirely satisfactory.

In the rest of this chapter, procedures for deriving the SDG of process units from conventional mathematical equations describing the behavior of the units are presented. Earlier procedures are augmented with rules for deriving the additional attributes. We also present a more satisfactory treatment of the derivation of SDG arcs from algebraic equations. First, we introduce the representation of the SDG in Section 4.2. In Sections 4.3 and 4.4, guidelines for deriving the SDG of process and control system units are presented. Finally, in Section 4.5, we demonstrate how these guidelines are used to derive the SDG of several types of units.

## 4.2. Representation of the SDG

The SDG consists of three basic types of objects:<sup>1</sup>

1. Variables
2. Causal arcs
3. Root causes

In the following, we list attributes of these objects.

---

<sup>1</sup>Other objects and attributes are defined in the MIDAS implementation of the SDG. The complete list of objects and attributes is presented in Chapter 6.

## 4.2.1. Variables

SDG variables (nodes) symbolize process variables or parameters in the system (unit). The variables could either be measured or unmeasured. The following is the list of attributes of variables.

1. **Initiating arcs:** The value of this attribute is the list of causal arcs that begin at the variable. These arcs terminate at other variables that are causally affected by the variable.
2. **Terminating arcs:** The attribute's value is the list of arcs that end at the variable. These arcs initiate from other variables that causally affect the variable.
3. **Self cycle:** This indicates the sign of the self cycle associated with the variable. The sign of the self cycle is required to determine "apparent global" causal influences in the process ESDG. The attribute can have one of the values +, 0 or -. The value for the self cycle for most variables is -. However, variables associated with integrators and unstable "self-excitation" effects have values of 0 and +, respectively.
4. **Sensor signal:** Indicates whether the variable is a sensor signal. Sensor signals provides the distinction between measured and unmeasured variables. The attribute is useful when deriving the PM. Its value is either t or nil.
5. **Root causes on high:** The attribute's value is a list of root causes that cause an initial deviation of the variable to be higher than its nominal value.
6. **Root causes on low:** The value of this attribute is a list of root causes that cause an initial deviation of the variable to be lower than its nominal value.

## 4.2.2. Causal Arcs

Causal arcs of the SDG represent the immediate causal influences between variables and parameters in the system. An arc initiates at a variable in the SDG and terminates at another variable. The following is the list of attributes of causal arcs.

1. **Initiating variable:** The value of this attribute is the variable from which the arc begins.
2. **Terminating variable:** The attribute's value is the variable at which the arc ends.
3. **Arc sign:** The sign of an arc indicates whether a change in the value of the initiating variable in one direction tends to cause a change in the value of the terminating variable in the same or opposite direction. The attribute's value is normally + or -. However, a value of 0 can be used to indicate the non-applicability of the arc in a given context.
4. **Time delay:** The value is a measure of the propagation time associated with the arc's causal effect. A value of 0 indicates that the effect is transmitted instantaneously. This attribute is useful for ordering deviations among sensor signal variables when deriving the PM. Practically, an effect is considered to be transmitted instantaneously if transmission of the effect takes an amount of time much less than the time scale of observation ( $t_m$ ). Conversely, a delay of 1 indicates that the transmission takes an amount of time greater than or equal to the observation time scale.
5. **Magnitude:** This indicates whether the arc's effect is negligible. The default value of this attribute is 1. However, in certain situations when an effect is judged to be negligible, the attribute's value is 0. This can also be used to indicate the non-applicability of the arc in a particular context.
6. **Disabling conditions:** The value of this attribute is a list of root causes which inactivate the causal effect of the arc. These root causes change the basic structure of the SDG. Typical examples are zero gain events in control loops, when an element of the loop is stuck at a fixed value.

### 4.2.3. Root Causes

Root causes symbolize the malfunctions that affect the unit. The primary effect of a malfunction is to cause a deviation in the value of a variable in the SDG. The following is the list of attributes of root causes.

1. **Primary deviation variable:** The attribute's value is the variable or parameter which is causally affected by the root cause. It is assumed that each malfunction has only one primary deviation variable. In situations where a root cause may have more than one primary deviation variable, an additional dummy variable is included in the SDG.
2. **Deviation direction:** The value of this attribute is the initial direction of deviation of the primary deviation variable from its nominal value. The attribute assumes one of two values, high (+) or low (-).

## 4.3. Developing SDG Models for Process Units

Chemical process units may be classified as either process units, or control system units. Units such as process controllers and decouplers are examples of control system units. These devices perform process computations. Other units, including control valves, measurement sensors are classified as process units.

Generally, control system units and process units are modeled at different levels of abstraction. Conventional mathematical models of control system units do not describe the underlying physical processes taking place in the devices, but describe the mathematical relationships between input and output variables of the device. On the other hand, consideration of the underlying physical phenomena is the usual starting point for deriving models for process units. The resulting models describe to varying degrees the physical processes occurring in the units.

In this section, the principles governing the conventional modeling of process units are reviewed. Guidelines for deriving the SDG from conventional equation models of these units are presented. Modeling of control system units is considered in Section 4.4.

### 4.3.1. Variable Selection

As in conventional modeling, the first step in deriving the SDG of a unit is to determine the set of variables describing the unit. In selecting the variables that describe a unit, two requirements must be satisfied:

- The variables should completely specify the behavior of the unit at the given level of detail.
- A consistent set of boundary variables should be defined for adjacent units.

For lumped parameter systems (units), Gibbs phase rule is a convenient starting point for guiding the selection of variables describing the system. In essence, the rule states that a closed simple system at static equilibrium is described by an extensive variable and a number of intensive variables (degrees of freedom). The number of degrees of freedom  $F$ , is given by

$$F = C + 2 - P - R \quad (4.1)$$

where  $R$  is the number of independent reactions,  $P$ , the number of phases and  $C$ , the number of components. The extensive variable normally chosen is the mass of the material in the system.

Applying the phase rule to a simple, closed, single phase, multicomponent, non-reacting system,  $C + 1$  intensive variables are required to completely specify the state of the system at equilibrium. There are some constraints as to the combinations of intensive variables that can be selected. Variables usually selected are temperature,  $T$ , pressure  $P$ , and the concentration of  $C-1$  components [4].

For open systems, with material transfer across the system's boundaries, the mass flowrate across the boundaries is added to the list of variables. A non-simple system is one with work and energy interaction at the system's boundaries. For these systems, variables representing work and energy interactions at the boundaries are added to the list.

For non-simple, open multiphase, multicomponent non-reacting systems, we begin by describing each phase by its temperature, pressure and the concentration of C-1 components, and its mass. Assuming no internal restraints to mass and energy transfer and no work interactions between phases, the mass and energy flow rate across phase boundaries are also included. Therefore, a total of  $P_{\bullet}(C + 1)$  intensive variables describe the phases. Applying the phase rule at equilibrium, a total of

$$P_{\bullet}(C + 1) - (C + 2 - P) = (C + 2)_{\bullet}(P - 1) \quad (4.2)$$

of these intensive variables are redundant. At equilibrium these redundant variables are related to the set of independent variables through equilibrium relationships. However, some of the equilibrium relationships may not hold during the transient. Instead, a combination of rate transport and equilibrium equations often apply. Parameters (variables) representing the transport rates are also included in the list of system variables.<sup>2</sup>

For reacting systems, the degrees of freedom in each phase is reduced by the number of reactions occurring in the phase. We introduce an equation and one additional process parameter (variable) to represent the reaction rate for each independent chemical reaction.

Summarizing, the following variables are typically used to model the **dynamic** behavior of lumped parameter systems.

### **Extensive**

1. Mass of material in each phase.
2. Mass, energy and work transfer across unit boundaries.

### **Intensive**

1. Phase temperature.
2. Phase pressure.

---

<sup>2</sup>It will be shown that the SDG is best derived from a dynamic model, rather than an equilibrium (static) model.

3. C - 1 phase compositions.

### Variables expressing transfer rates

1. R parameters, describing each independent reaction rate in each phase.
2. C parameters, describing species transport across phase boundaries.
3. Variables describing energy (and if applicable work) transport across phase boundaries.

For distributed parameter systems, the variables are further discretized according to Section 4.3.3.3.

A consistent set of interface variables for adjacent units should be specified. By convention we specify phase pressures, temperatures and compositions at a unit's boundaries as the interface variables.

## 4.3.2. Selecting the Malfunctions

The next step in deriving the SDG of a process unit is to determine the set of malfunctions that may affect the unit.<sup>3</sup> In modeling the effects of disturbances, a distinction is made between those at the boundary of the process system and other disturbances. For units at the boundaries of the process system, these disturbances are also modeled as malfunctions. For other units, these disturbances are not included in the set of malfunctions.

During this step, engineering judgement is used in selecting a reasonable set of malfunctions. If a malfunction is not modeled a priori, the diagnostic system can never diagnose that malfunction as a root cause for observed process deviations. On the other hand, considering all possible malfunctions, including improbable ones, potentially reduces the resolution and the speed of the diagnostic system.

---

<sup>3</sup>In this context, a malfunction is the basic cause of all process deviations from the nominal steady state. Disturbances, on the otherhand lead to further propagation of the effects of the malfunction via variables in adjacent units.

Associated with malfunctions are their primary deviation variables. Sometimes the effects of a malfunction is best modeled as directly affecting a process parameter (such as flow resistance, heat transfer coefficient). These parameters are added to the list of SDG variables selected in the previous section.

### 4.3.3. Determining the Causal Structure

Conventional dynamic equation models describing process behavior may be represented as sets of algebraic, ordinary and partial differential equation equations. The final step in deriving the SDG of a unit, is to represent the causal influences among variables in the unit. Other workers [1, 2 & 3] have provided procedures for specifying SDG arcs from equations describing the unit. All these procedures incorporate in one way or another, the basic idea of assigning causal influences among variables and parameters related by a single equation.

In this section, we present an alternative set of procedures for specifying causal influences from a general set of conventional equations. Given a system of equations, a variable is assigned as the "output" variable for each equation. Output variables are selected subject to the constraint that no variable is the output for more than one equation. For each equation, causal arcs initiate from other variables contained in the equation and terminate at the output variable. Thus, specifying output variables for a set of equations is equivalent to specifying the direction of causal influences in the SDG. In the following, rules for specifying output variables of the equations, the resulting SDG arcs and other attributes of the SDG are detailed.

#### 4.3.3.1. INDEPENDENT VARIABLES AND PARAMETERS

Independent variables and parameters are variables whose values are externally specified with respect to the equations set, and assumed to be independent of values of other variables in the unit (process). These variables are not assigned as outputs of any of the equations describing the unit. Consequently, SDG arcs do not terminate at these

variables. Since these variables cannot be associated with integrating or self-excitation effects,<sup>4</sup> the sign of the **self cycle** associated with the variable is -.

Values of all additional parameters introduced in Section 4.3.2 to model the effects of malfunctions are independent of values other variables in the unit.<sup>5</sup> These malfunctions are included as members of the **root causes on high** and **root causes on low** attributes of the parameters. Similarly, the parameters are the **primary deviation variable** and **deviation direction** of these root causes.

#### 4.3.3.2. ORDINARY DIFFERENTIAL EQUATIONS

Causality implies the passage of time between the cause and effect variables. Differential equations have an explicit time dependence and causal influences can be derived from ordinary differential equations as shown below.

The first step in specifying causal influences for a unit described solely by ordinary differential equations is to transform the equations into a set of first order differential equations

$$\frac{dx}{dt} = f(x, u, p); \quad x = x_0, \quad u = u_0, \quad p = p_0 \quad (4.3)$$

Here, x, u, and p, are vectors of state variables, inputs variables and unit parameters, respectively. Linearizing about initial conditions, the equations become

$$\frac{d(dx)}{dt} = \underline{A} \cdot (dx) + \underline{B} \cdot (du) + \underline{D} \cdot (dp) \quad (4.4)$$

where dx, du and dp are deviations from initial steady state values. Elements  $a_{ij}$ ,  $b_{ij}$ , and  $d_{ij}$  of the matrices A, B and D are related to partial derivatives of eq. (4.3). Expressed mathematically, if m is the order of the first non-zero partial derivative evaluated at the initial conditions,

---

<sup>4</sup>If the variables are associated with either of these effects, the system is unstable.

<sup>5</sup>In diagnostic applications, independent variables that do not have associated root causes may be omitted from the SDG.

$$a_{ij} = \partial^m f_i / \partial x_j^m \cdot dx_j^{m-1} \quad (4.5a)$$

$$b_{ij} = \partial^m f_i / \partial u_j^m \cdot du_j^{m-1} \quad (4.5b)$$

$$d_{ij} = \partial^m f_i / \partial p_j^m \cdot dp_j^{m-1} \quad (4.5c)$$

For each of these equations in the set,  $x$ , is assigned as the output variable. The direction of causality of variables in eqs. (4.3) and (4.4) is from the right to left hand side of the equation. A causal arc initiates from  $x_j$ ,  $u_j$  or  $p_j$  and terminates at  $x_i$  ( $x_j \neq x_i$ ), if the corresponding partial derivative ( $a_{ij}$ ,  $b_{ij}$ , or  $d_{ij}$ ) is not strictly zero.<sup>6</sup>

The sign of the arc is related to the qualitative sign of the corresponding partial derivative. For example, the sign of the arc from  $x_j$  to  $x_i$  is:

(i)  $[\partial^m f_i / \partial x_j^m]$ , for  $m$  odd

(ii)  $[\partial^m f_i / \partial x_j^m] \cdot [dx_j]$ , for  $m$  even

where  $[.]$  is qualitative sign of its argument, .. Signs of arcs from  $u_j$  and  $p_j$  to  $x_i$  are defined similarly. Typically, these partial derivatives are expressions relating unit parameters and variables. In most situations, detailed numerical information is not required to unambiguously determine their signs. Often knowledge of ordinal relationships such as relative stream temperatures is sufficient to determine these signs.

Inequalities  $\underline{C}$ , for determining the range of validity of arc signs, and thus the SDG, are directly related to these ordinal relationships. A region of parameter and variable values where arc signs are valid is defined as a qualitative regime. Typically, these transitions across qualitative regimes occur when:

- The driving forces of phenomena such as momentum (pressure), heat (temperature) and mass (specie concentration) transfer change direction.

---

<sup>6</sup>An expression is strictly zero if its value is zero, independent of the particular numerical values of variables and parameters.

Transitions could also occur in situations when the original equation from which the SDG arc was derived is no longer applicable. Examples of such situations is when a variable reaches physical bounds such as in control loop saturation or phase changes. Modeling these situations requires a separate SDG model for each qualitative regime.

The sign of the **self cycle** of a variable is the sign of the corresponding diagonal element of the A matrix:

- (i)  $[\partial^m f_i / \partial x_i^m]$ , for m odd.
- (ii)  $[\partial^m f_i / \partial x_i^m] \cdot [dx_i]$ , for m even.

Integrators are associated with strictly zero self cycles, while unstable tendencies are represented by positive self cycles.

In situations where signs of any element in the above matrices cannot be determined unambiguously, additional dummy variables could be added to the SDG to create dual influences. Equation (4.19a) for the CSTR in Section 4.5.3, is an example of a case where an SDG variable is retained to avoid ambiguity.

If the output variable of the differential equation is a sensor signal the value of the **sensor signal** attribute of the variable is t. Otherwise the value of the attribute is nil.

The numerical value of the time associated with transmission of effects through arcs terminating at  $x_i$  is of the same order of magnitude as the time constant of the equation for which the variable is the output variable ( $-1/a_{ij}$ ,  $a_{ij} \neq 0$ ).<sup>7</sup> Unless the time constant is known to be much less than the time scale of observation (i.e.  $| -1/a_{ij} | \ll t_{\text{max}}$ ), the value for the **time delay** of an arc is assumed to be 1. When  $a_{ij}$  is 0, the delay is assumed to have the value, 1.

---

<sup>7</sup>A scaled single input, single output (SISO) first order system is one in which the ratio of the thresholds of input,  $x_j$ , and output,  $x_i$ , is  $-a_{ij}/a_{ii}$ . For an input disturbance, the time for  $x_i$  to reach its threshold,  $t_l$  is  $(-1/a_{ij}) \cdot \ln(k/(k-1))$ , where k is the ratio of the magnitude of the input disturbance to its threshold. If the input variable crosses its threshold,  $k > 1$ . For most reasonable values of k; ( $1.000045 \leq k \leq 10.5083$ ),  $t_l$  is of the same order of magnitude as the quantitative value of the time delay of the arc.

In certain situations, the primary effect of a malfunction is to cause a deviation in one of the variables that has been assigned as an output to an equation. This usually occurs for zero gain events where the value of the primary deviation variable remains at a fixed value. These malfunctions included as members of the **root causes on high** and **root causes on low** attributes of the output variables. Similarly, the **primary deviation variable** and **deviation direction** of these root causes (malfunctions) are the output variables. In these situations, an additional effect of the malfunction is to inactivate the effects of all causal arcs terminating at the variable. This is modeled by including the malfunction as a member of the **disabling conditions** attribute of all arcs terminating at the arc. The arc between  $S_i$  and  $S_p$  in the SDG of the control valve (Fig. 4.5) includes malfunctions in its list of disabling conditions.

#### 4.3.3.3. PARTIAL DIFFERENTIAL EQUATIONS

Partial differential equations are used to describe distributed parameter systems. In these systems, values of certain process variables may vary spatially. Unlike in lumped parameter systems, these units, cannot be described by a finite number of state variables. However in practice, partial differential equations can be reduced to ordinary differential equations by the method of lines or a similar method, and then the SDG can be derived for these units according to the procedure outlined in the previous section.

In numerical simulation, many spatial discretization points may be required to provide acceptable accuracy. However in qualitative simulation, relatively few discretization points are required to determine all qualitative system behaviors. Further discretization does not yield additional real behaviors but may lead to generation of spurious solutions. The optimum discretization must be determined by trial and error [3]. In our experience, we have found that we usually do not require more than 3 nodes per variable in units with one independent spatial direction. Examples of the SDG of distributed parameter systems are the pipe and heat exchanger in Sections 4.5.1 and 4.5.4, respectively.

#### 4.3.3.4. ALGEBRAIC EQUATIONS

Algebraic equations are not causal process models as they indicate an instantaneous transmission of effects. For a unit described by a set of differential and algebraic

equations, the problem then is to specify output variables for the remaining algebraic equations. In general, it may not be possible to uniquely assign output variables for the remaining algebraic equations. Thus, the direction of the causal influences among variables cannot be derived based solely on the structure of each equation. This is further complicated by the possibility of algebraic manipulation of a set of independent equations to obtain other sets of independent equations.

Knowledge of the origin of the equation, the underlying processes, device physics and context must be utilized to determine the direction of causal influences. To help specify output variables, algebraic equations can be further classified as:

- Conservation equations.
- Rate transport equations.
- Reaction rate expressions.
- Algebraic equalities.

In order to specify the direction of causality Palowitch [3] classifies algebraic equations as rate transport equations, functional relationships (e.g. kinetic equations) and algebraic equations. The classification presented here includes conservation equations as a separate type.

### *Conservation Equations*

These algebraic equations result from approximations to differential equation models describing the conservation of fundamental properties such as mass and energy. These equations should always be restored to their original differential equation form if possible. These algebraic equations derive from two sources.

- (i) **Pseudo steady state approximations:** In certain systems, the magnitudes of time constants associated with some variables are much larger than time constants for other variables in the system. In conventional modeling, the values of the derivatives of these variables is sometimes approximated as 0. This results in an

algebraic equation. Converting these equations to the original differential equations is usually trivial. SDG arcs can then be derived for the resulting differential equations according to the procedures in Section 4.3.3.2.

- (ii) **Equations with fast dynamics:** These equations arise from conservation relations which if modeled in greater detail are described by differential equations. Transients associated with these relations die out quickly. Examples are the flow-pressure relationships for incompressible fluid flow in a pipe. In Section 4.5.1, we show that by modeling these relationships at a greater level of detail, the algebraic equations can be converted to differential equations. In converting these equations, only the qualitative form of the relationships and not the actual differential equations need to be specified. The procedure in Section 4.3.3.2 for deriving the SDG from the resulting differential equations apply. However, the delay associated with the equation is usually less than the observation time scale. Therefore, arcs from these equations have a **time delay** of 0.

### *Rate Transport Equations*

Rate transport equations result from driving force relationships for heat and mass transfer. Typically, these equations relate the rate of transport of energy (mass) with a temperature (specie concentration) difference. Since temperature and species concentrations are specified as output variables of conservation equations, variables representing the transport rates are specified as output variables for these equations. The sign of the **self cycle** associated with these variables is -.

Causal arcs derived from these equations **initiate** from other variables present in the equation and **terminate** at the variable denoting the transport rate. These arcs have a **time delay** of 0. The **sign** of the arcs is defined similarly as for ordinary differential equations and is related to the sign of the corresponding partial derivative. Other attributes of the arcs are determined as for ordinary differential equations.

In cases of energy (mass) transport across interphase boundaries, the variable denoting the transport rate may be eliminated from conservation equations for energy (specie mass). Eliminating these variables generally results in a positive cycles in the SDG. Positive cycles in the SDG are necessary but not sufficient conditions for variables exhibiting

inverse response to negative feedback (Section 3.3.2). This potentially results in ambiguity in determining inverse variables when deriving the process ESDG. It is shown in Appendix D, that inverse response due to negative feedback can never occur as a result of positive cycles associated with transport across interphase boundaries. However retaining the variable for the transport rate eliminates the positive cycle, thus resolving the ambiguity.

### *Reaction Rate Expressions*

Reaction rate expressions are empirical relationships relating the rate of reaction to the concentrations of species involved in a chemical reaction. Since the species concentrations are specified as output variables of conservation equations, variables representing reaction rates are specified as output variables for these equations. The sign of the **self cycle** associated with these variables is -.

Causal arcs derived from these equations **initiate** from other variables present in the equation and **terminate** at the variable denoting the reaction rate. These arcs have a **time delay** of 0. The **sign** of the arcs is defined similarly as for ordinary differential equations and is related to the sign of the corresponding partial derivative. Other attributes of the arcs are determined as for ordinary differential equations.

For reversible reactions, the variable denoting the reaction rate may be eliminated from conservation equations for chemical species. Eliminating these variables results in positive cycles in the SDG. It is shown in Appendix D, that inverse response due to negative feedback can never occur as a result of positive cycles associated with reversible reactions. Retaining the variable for the reaction rate eliminates the positive cycle, thus resolving the ambiguity. The reaction rate variable is also retained for exothermic reactions. Otherwise an ambiguity results for the sign of the self-cycle associated with temperature.

### *Algebraic Equalities*

Algebraic equalities are relationships that are exactly satisfied during the transient. Many of these equations may be derived from fundamental definitions and from definitions for material properties and process equipment parameters. For example, the relationship

$$L = V/A \quad (4.6)$$

between liquid volume and level in a vessel defines the cross sectional area of the vessel. Similarly, the relationship between the pressure at the base of a vessel and the fluid level

$$P_b = L\rho g \quad (4.7)$$

is obtained from combining the definitions of pressure, weight and liquid density.

In many situations it may be possible to uniquely assign outputs to the remaining variables such that no variable in the system becomes the output of more than one equation. When this is not possible, specifying output variables requires knowledge of the origin of these equations. For example level,  $L$  is usually assigned as the output variable of eq. (4.6). Saying that changing the volume of liquid in a vessel causes a change in level rather than level changes cause volume changes is more intuitive. Similarly, it is intuitive to state that the (weight of the) liquid in a tank exerts a pressure on the base of the tank. Thus, we can justify selecting base pressure as the output variable of eq. (4.7). The sign of the self cycle associated with these output variables is -.

Causal arcs derived from these equalities **initiate** from other variables in the equation and **terminate** at the output variable. These arcs have a **time delay** of 0. The **sign** of the arcs is defined similarly as for ordinary differential equations and is related to the sign of the corresponding partial derivative. Other attributes of the arcs are determined as for ordinary differential equations.

#### **4.3.3.5. SUMMARY OF ARC SPECIFICATION PROCEDURES**

In summary, the procedure for deriving the SDG of a system described by a mixed set of ordinary differential, partial differential and algebraic equations is as follows.

1. Identify the set of independent variables and parameters.
2. Convert all ordinary differential equations to a system of first order equations.
3. Discretize partial differential equations to ordinary differential equations.
4. Revert algebraic equations deriving from conservation relationships to first order ordinary differential equations.
5. Assign outputs to the equations for transport and reaction rates.
6. Assign output variables to the remaining algebraic equations ensuring no variable is the output variable of more than one equation.
7. Determine the attributes of variables, causal arcs and root causes according to the procedures in the previous sections.

#### **4.4. Developing SDG Models for Control System Units**

Control system units perform process computations. These units are usually modeled at a relatively high level of abstraction. The models do not consider the underlying physical phenomena but provide explicit relationships between input and output variables. The procedure for developing SDG models for these units is usually more straightforward than the procedure for process units.

Starting with vectors of input ( $\underline{u}$ ) and output ( $\underline{y}$ ) variables for the unit, we ensure that a consistent set of boundary variables are defined for each unit. Then, engineering judgement is used to select the set of malfunctions affecting the unit. The most common types of failures are:

- Failure of a variable to a fixed value.
- An additive bias of a variable.
- A change in the value of one the units inputs.<sup>8</sup>

For additive biases, additional variables representing the bias is added to the list of the units variables.

Next, causal influences are inferred from the conventional equation models describing the unit. The first step in determining the causal influences, is to transform the equations into state-space canonical form

$$\underline{dx}/dt = \underline{A} \cdot \underline{x} + \underline{B} \cdot \underline{u} \quad (4.8a)$$

$$y = \underline{C} \cdot \underline{x} + \underline{D} \cdot \underline{u} \quad (4.8b)$$

Additional variables ( $x_i \neq y_j$ ), introduced by this transformation are added to the list of the units variables. The overall direction of causality is from the units inputs to the outputs. The output variables for eqs. (4.8a) and (4.8b) are  $\underline{x}$  and  $y$ , respectively.

Once the direction of causality has been specified, attributes of SDG arcs and variables are defined similarly to the procedures in Section 4.3. Specifically, malfunctions associated with the failure of a variable to a fixed value disable all causal arcs terminating at the primary deviation variable of the malfunction. An example of the derivation of the SDG of a control system unit is presented in Section 4.5.6.

---

<sup>8</sup>By convention, failures of outputs of other units are represented in the SDG of those units.

## 4.5. Examples of SDG Models of Process and Control System Units

In this Section, we demonstrate how the guidelines outlined in Sections 4.3 and 4.4 are used to derive the SDG of several types of units. Emphasis is placed on the procedure for determining causal influences from equations describing the units.

### 4.5.1. Pressure-flow Phenomena in a Pipe

A pipe is a distributed parameter system with no capacity for mass storage. Temperature effects are not modeled, and we assume an incompressible fluid with constant physical properties. Since density is constant, the volumetric flowrate instead of the mass flowrate can be used to describe the pipe.

Variables describing this unit are  $P_{in}$ ,  $P_{out}$ , and  $F$ . The only malfunctions assumed to affect the pipe are a blockage and a leak at the pipe's outlet. Additional parameters (variables) required to model these malfunctions are pipe resistance  $R_s$  and the leak flowrate  $Q_l$ . Leak flowrate and resistance are assumed independent of flowrate,  $F$  and pressures  $P_{in}$  and  $P_{out}$ .

We can also consider an internal leak in the pipe or a leak at its inlet as possible malfunctions. The effects of these leaks are often indistinguishable given the location of sensors in most processes. We use engineering judgement to eliminate these leaks as possibilities. This does not eliminate the ability to diagnose these leaks. A leak at a pipe's inlet can be considered to be a leak at the outlet of the pipes upstream unit. An internal leak can be modeled by joining two pipe units together.

The pipe has very fast dynamics and algebraic equations describing the pipe are

$$F = f^-(R_s) \cdot f^+(P_{in} - P_{out}) \quad (4.9)$$

$$F = F_{in} \quad (4.10)$$

$$F = F_{out} + Q_l \quad (4.11)$$

where  $F_{in}$ ,  $F_{out}$  are flowrates in adjacent units.

We shall now show that these result from differential equations when the phenomena are modeled in greater detail.

Considering a momentum balance for a liquid slug of length  $L$  in the pipe (Fig. 4.1a), we have

$$\frac{d^2(\rho VL)/dt^2}{\text{mass x acc}} = \rho L dF/dt = A_c(P_{in} - P_{out}) - 2\pi r L f^+(F) \quad (4.9a)$$

Net Force      friction losses

Equation (4.9a) reduces to eq. (4.9) at steady state.  $F$  is its output variable.

Now we derive the dynamic equations with boundary pressures as their output variables. We model the boundary pressures as open vessels with a finite amount of storage. This is analogous to considering the fluid to be partially compressible. The mass balances for these vessels are

$$(g/A) \cdot dM_{in}/dt = dP_{in}/dt = (\rho g/A) \cdot (F_{in} - F) \quad (4.10a)$$

$$(g/A) \cdot dM_{out}/dt = dP_{out}/dt = (\rho g/A) \cdot (F - F_{out} - Q_l) \quad (4.11a)$$

Once again, these equations reduce to their respective steady state forms above.

On linearizing the ordinary differential equations, arcs in the SDG are derived according to the rules in Section 4.3. The SDG is shown in Fig. 4.1b. The ordered pair associated with SDG arcs denote the arc sign and delay. Since the branches are derived from equations with fast dynamics, the time delays on all branches are zero. The boundary pressures are associated with integrators (self cycles with value 0). The value of the sign of the self cycles of other variables is the default value, -. While positive and zero self-

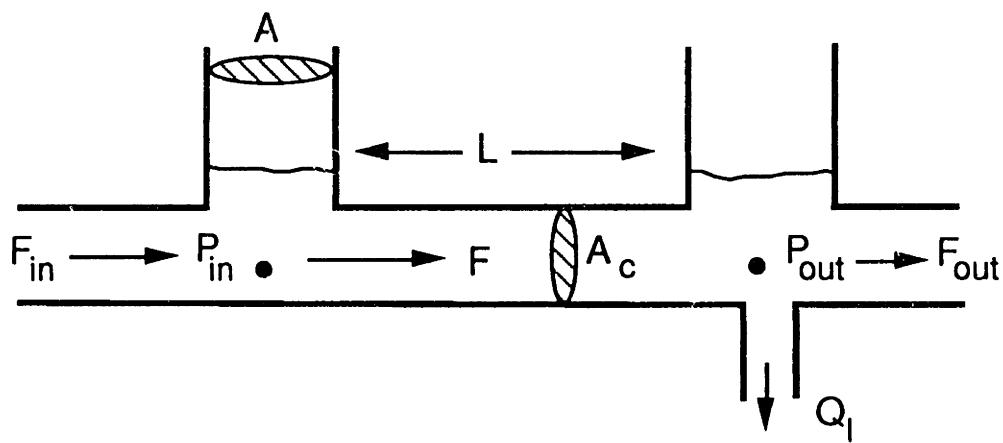


Fig. 4.1a. Model of Flow-Pressure Phenomena in Pipes

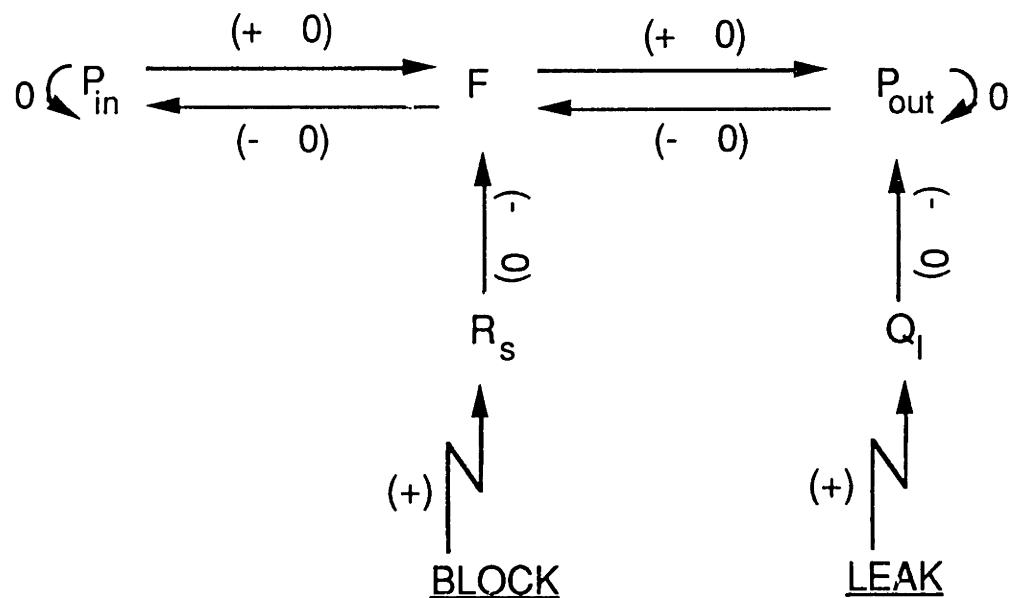


Fig. 4.1b. SDG of Pipe

cycles associated with variables are depicted on the SDG, negative self-cycles are not shown to reduce clutter. The effect of root causes on their primary deviation variables are also shown.

## 4.5.2. Mixing Tank

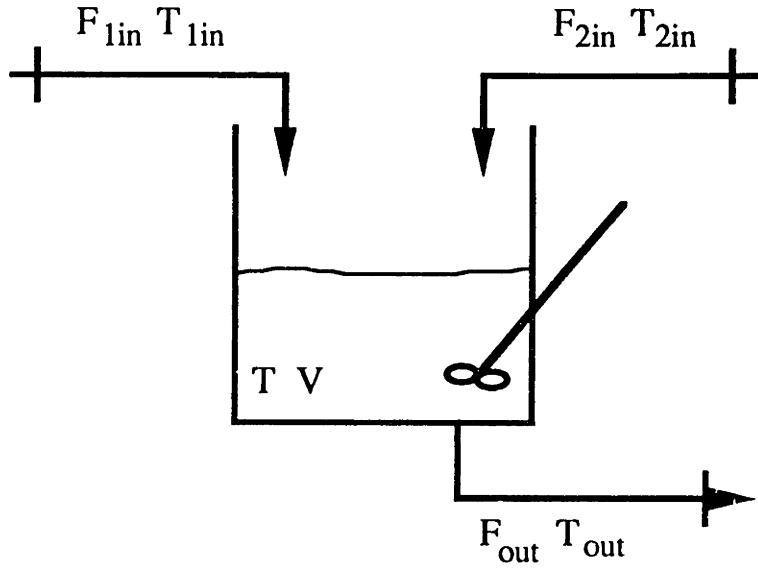


Fig. 4.2a. Mixing Tank

Consider a mixing tank with two inlet streams of unequal temperature (Fig. 4.2a). This is a lumped parameter system with a capacity for material (energy) storage. The densities and compositions of both streams are equal and we assume that physical properties are independent of temperature.

The externally specified variables are  $T_{\text{in}}$  and  $F_{\text{in}}$  of both inlet streams and  $P_{\text{out}}$  at the tank outlet. The variables describing the tank are  $V$ ,  $L$ ,  $P_b$  and  $T$ , and  $T_{\text{out}}$  and  $F_{\text{out}}$  at the outlet. Malfunctions affecting the tank are a tank leak, tank outlet blockage and loss of insulation which affect the parameters  $Q_l$ ,  $R_x$  and  $H_v$ .

The equations relating the tank variables are

$$\frac{dV}{dt} = F_{1\text{in}} + F_{2\text{in}} - F_{\text{out}} - Q_l \quad (4.12)$$

$$\frac{d(T \cdot V)}{dt} = F_{1\text{in}} \cdot T_{1\text{in}} + F_{2\text{in}} \cdot T_{2\text{in}} - F_{\text{out}} \cdot T - Q_l \cdot T + H_v \quad (4.13)$$

from material and energy balances and

$$T_{out} = T(t - t_d) \quad ; \quad t_d \ll t_m \quad (4.14)$$

$$F_{out} = f^-(R_x) \cdot f^+(P_b - P_{out}) \quad (4.15)$$

$$L = V/A_c \quad (4.16)$$

$$P_b = \rho g L \quad (4.17)$$

Equation (4.13) can be transformed into standard form by combining it with eq. (4.12) to obtain,

$$dT/dt = (1/V) \cdot (F_{1in} \cdot (T_{1in} - T) + F_{2in} \cdot (T_{2in} - T) + H_V) \quad (4.13a)$$

Standard procedures for differential equations are used to determine the signs and delays of arcs derived from eqs. (4.12) and (4.13a). An integrator is associated with V. The signs of arcs from  $F_{1in}$  and  $F_{2in}$  terminating at T are dependent on the relative stream temperatures  $(T_{1in} - T)$  and  $(T_{2in} - T)$ , respectively. The sign of each arc is equal to the sign of the temperature difference.

Equation (4.14) can be transformed to a partial differential equation, which on discretization yields a positive arc with 0 delay from T to  $T_{out}$  ( $t_d \ll t_m$ ). Equation (4.15) describes the flow-pressure relationship at the tank outlet and an arc is derived as in the previous example.

Equations (4.16) and (4.17) are strict algebraic equalities. Only one consistent set assignment of output variables is possible. L is assigned as an output variable of eq. (4.16) and  $P_b$  of eq. (4.17). The values of the sign of the self cycles associated with both variables is the default value -. Assuming  $T_{1in} > T > T_{2in}$  and  $T > T_{amb}$ , the SDG of the tank is shown in Fig 4.2b. Additionally, we include interface variables of adjacent units and their associated causal effects.

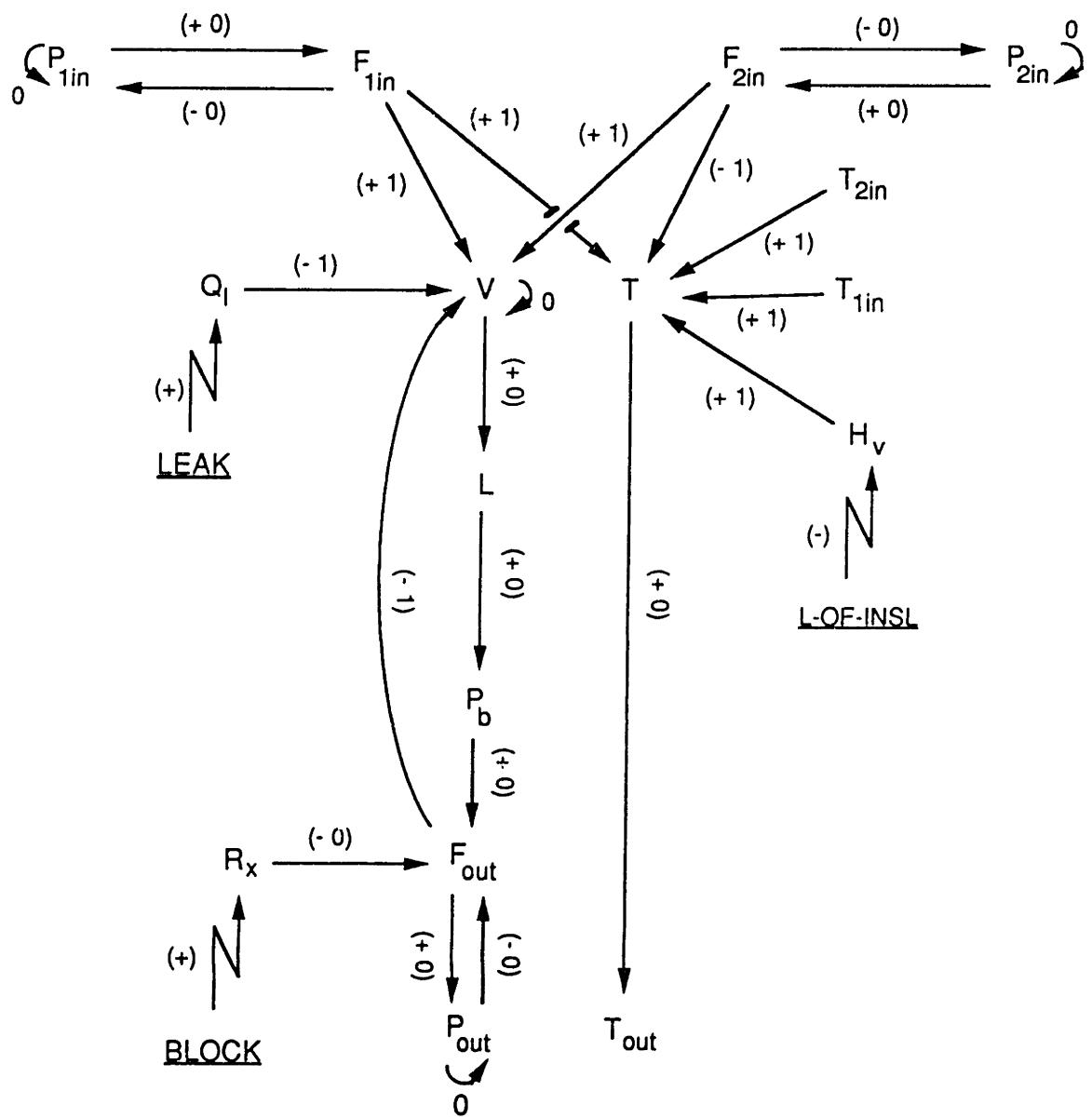


Fig. 4.2b. SDG of Mixing Tank

### 4.5.3. Continuous Stirred Tank Reactor

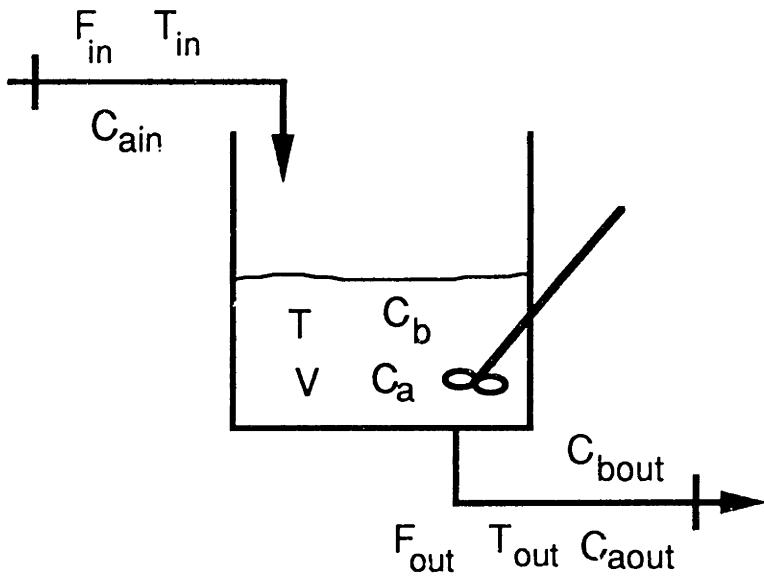


Fig. 4.3a. Continuous Stirred Tank Reactor

Consider the continuous stirred tank reactor (CSTR) shown in Fig. 4.3a. An irreversible, exothermic ( $\Delta H_{rxn} < 0$ ), catalytic reaction ( $A \rightarrow B$ ) with rate expression,  $r = k_r C_a^n$ , takes place in the CSTR. The inlet stream variables  $C_{ain}$ ,  $T_{in}$  and  $F_{in}$  and tank outlet pressure,  $P_{out}$  are externally specified. We assume constant physical properties for the liquid in the tank. The variables describing the tank are  $V$ ,  $L$ ,  $P_b$ ,  $T$ ,  $C_a$  and  $C_b$ , and  $T_{out}$ ,  $F_{out}$ ,  $C_{aout}$  and  $C_{bout}$  at the outlet. An additional independent variable  $r$ , describes the rate of reaction. Malfunctions associated with the reactor are a leak, outlet blockage, catalyst deactivation and loss of insulation which affect parameters  $Q_l$ ,  $R_X$ ,  $k_r$  and  $H_V$ , respectively.

The equations describing the reactor are

$$\frac{dV}{dt} = F_{in} - F_{out} - Q_l \quad (4.18)$$

$$\frac{d(T \cdot V)}{dt} = F_{in} \cdot T_{in} - F_{out} \cdot T - Q_l \cdot T + H_V + (rV/\rho C_p) \cdot (-\Delta H_{rxn}) \quad (4.19)$$

$$\frac{d(C_a \cdot V)}{dt} = F_{in} \cdot C_{ain} - F_{out} \cdot C_a - Q_l \cdot C_a - r \cdot V \quad (4.20)$$

$$d(C_b \cdot V)/dt = -F_{out} \cdot C_b - Q_l \cdot C_b + r \cdot V \quad (4.21)$$

from overall material, energy and specie material balances. The reaction rate expression is given by

$$r = k_r \cdot C_a^n \quad (4.22)$$

$$k_r = f^+(T) \quad (4.23)$$

Other equations describing the system are;

$$T_{out} = T(t - t_d) ; t_d \ll t_m \quad (4.24)$$

$$C_{aout} = C_a(t - t_d) ; t_d \ll t_m \quad (4.25)$$

$$C_{bout} = C_b(t - t_d) ; t_d \ll t_m \quad (4.26)$$

$$F_{out} = f^-(R_X) \cdot f^+(P_b - P_{out}) \quad (4.27)$$

$$L = V/A_c \quad (4.28)$$

$$P_b = \rho g L \quad (4.29)$$

SDG arcs for eqs. (4.18) and (4.24) through (4.29) are derived as in the previous example. Substituting eq. (4.18) into eqs. (4.19) through (4.21) we obtain,

$$dT/dt = (1/V) \cdot (F_{in} \cdot (T_{in} - T) + H_v) + (r/\rho C_p) \cdot (-\Delta H_{rxn}) \quad (4.19a)$$

$$dC_a/dt = (1/V) \cdot (F_{in} \cdot (C_{ain} - C_a)) - r \quad (4.20a)$$

$$dC_b/dt = -(1/V) F_{in} \cdot C_b + r \quad (4.21a)$$

We could also substitute eqs. (4.22) and (4.23) into eqs. (4.19a) through (4.21a). For example, eq. (4.19a) would yield

$$dT/dt = (1/V) \cdot (F_{in} \cdot (T_{in} - T) + H_V) + (f^+(T) \cdot C_a^n / \rho C_p) \cdot (-\Delta H_{rxn}) \quad (4.19b)$$

The partial derivative of the right hand side of eq. (4.19b) with respect to  $T$

$$f^{+'}(T)(-\Delta H_{rxn})/\rho C_p - F_{in}/V \quad (4.30)$$

cannot be unambiguously determined without numerical information specific to a particular CSTR (Section 4.3.3.2). Therefore, we do not perform this substitution and retain  $r$  to avoid ambiguity. In addition  $r$  is not eliminated as it is one of the variables outlined in Section 4.3.1 describing the unit.

Equations (4.19a) through (4.21a) are in standard form, therefore, they can be linearized and SDG arcs derived accordingly. For the remaining equations, eqs. (4.22) and (4.23),  $r$  and  $k_r$  are selected as their output variables. The sign of the self cycles associated with these variables is -. The SDG of the reactor is shown in Fig. 4.3b.

#### 4.5.4. Shell and Tube Heat Exchanger

We demonstrate the derivation of causal influences from rate transport equations by considering a shell and tube heat exchanger (Fig. 4.4a). Heat is transferred from the hot shell side liquid to cold tube side liquid. The heat exchanger is also a distributed parameter system with no capacity for mass storage. Inlet stream temperatures  $T_{sin}$ ,  $T_{tin}$ , are externally specified. Variables describing the unit are  $P_{sin}$ ,  $P_{sout}$ ,  $F_s$ ,  $P_{tin}$ ,  $P_{tout}$ ,  $F_t$ ,  $T_s$ ,  $T_{sout}$ ,  $T_t$  and  $T_{tout}$ . An additional variable,  $Q$  describes the rate of heat transfer from the shell liquid to tube side liquid. We assume incompressible liquids with constant physical properties and that the heat exchanger wall has negligible thermal capacity.

For clarity, we do not consider blockages and leaks in both sides of the heat exchanger. The only malfunctions considered are heat exchanger fouling and a loss of insulation which affect parameters  $U$  and  $H_V$  respectively.

Flow-pressure equations for both sides of the heat exchanger are similar to equations describing the same phenomena in the pipe of Fig. 4.1a. These result in an identical structure in the SDG.

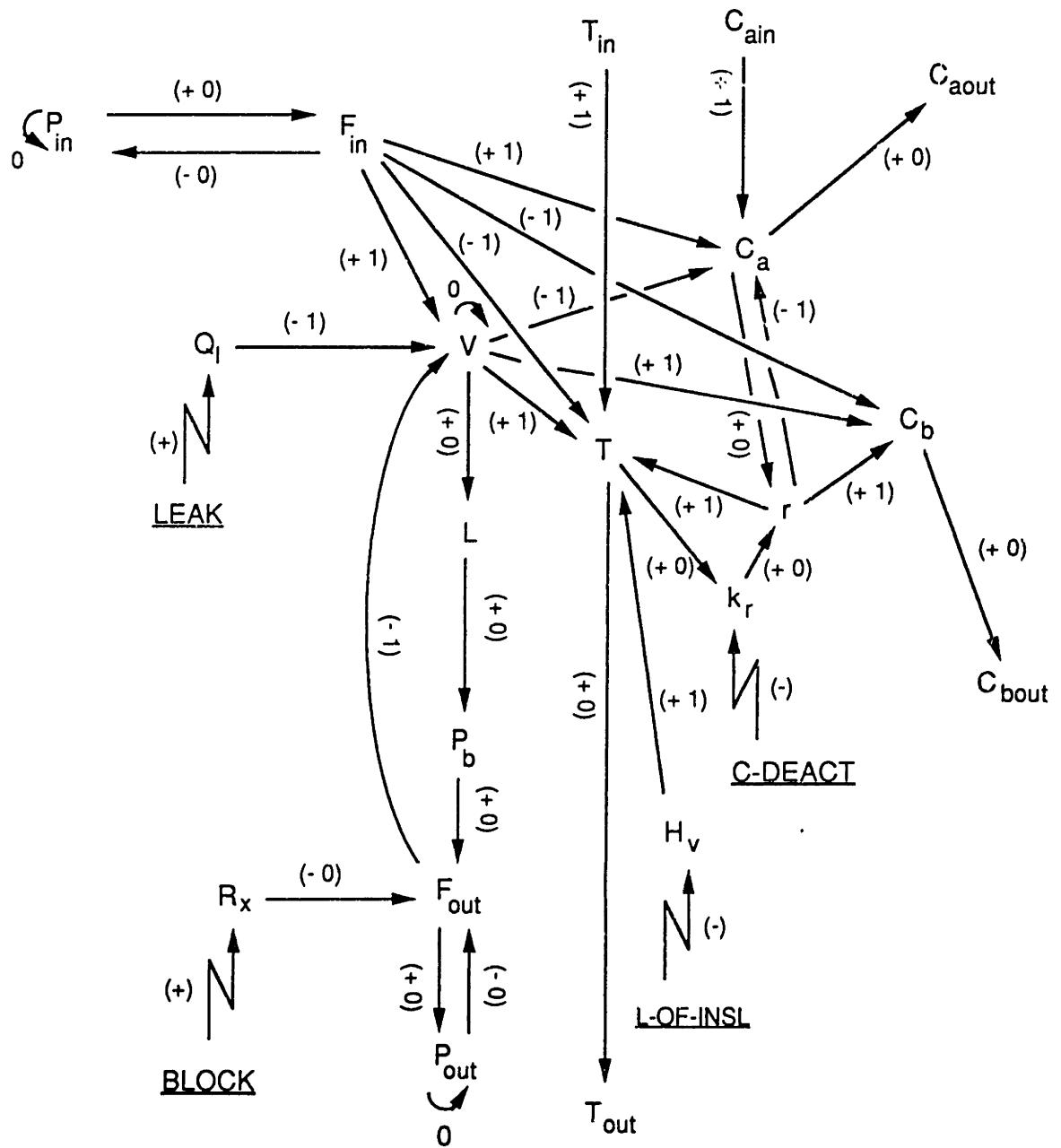


Fig. 4.3b. SDG of CSTR

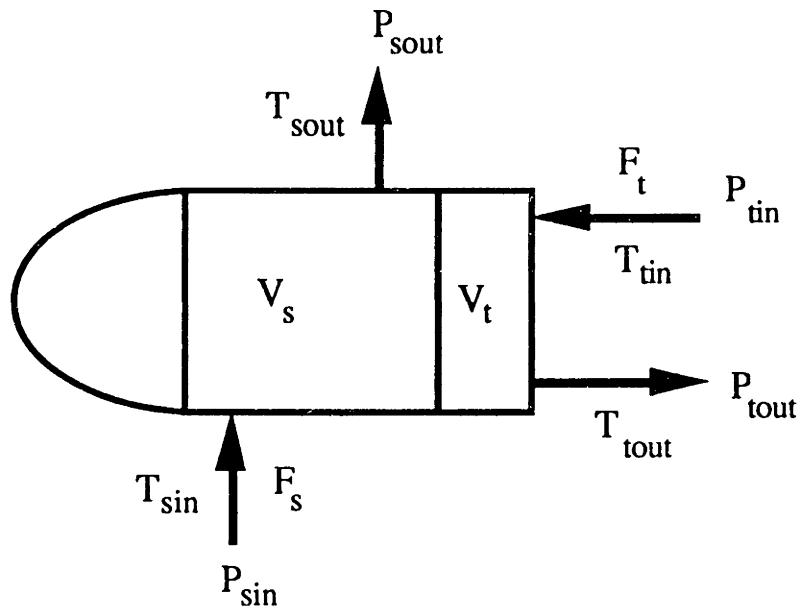


Fig. 4.4a. Shell and Tube Heat Exchanger

By approximating each side of the heat exchanger as a lumped parameter system with constant temperature, energy balances results in

$$(V_s)dT_s/dt = F_s(T_{sin} - T_s) + H_v - Q/(\rho C_p) \quad (4.31)$$

$$(V_t)dT_t/dt = F_t(T_{tin} - T_t) + Q/(\rho C_p) \quad (4.32)$$

$$T_{sout} = T_s(t - t_d) \quad ; \quad t_d \approx t_m \quad (4.33)$$

$$T_{tout} = T_t(t - t_d) \quad ; \quad t_d \approx t_m \quad (4.34)$$

where  $V_s$  and  $V_t$  denote the volume of liquid in the shell and tubes, respectively and are constant. The rate of heat transfer is given by

$$Q = UA_c(T_s - T_t) \quad (4.35)$$

SDG arcs for eqs. (4.31) through (4.34) are derived as in the previous examples.  $Q$  is assigned as an output to eq. (4.35) and SDG arcs are derived accordingly. The sign of the self-cycle associated with  $Q$  is -. The SDG for the heat exchanger is shown in Fig. 4.4b.

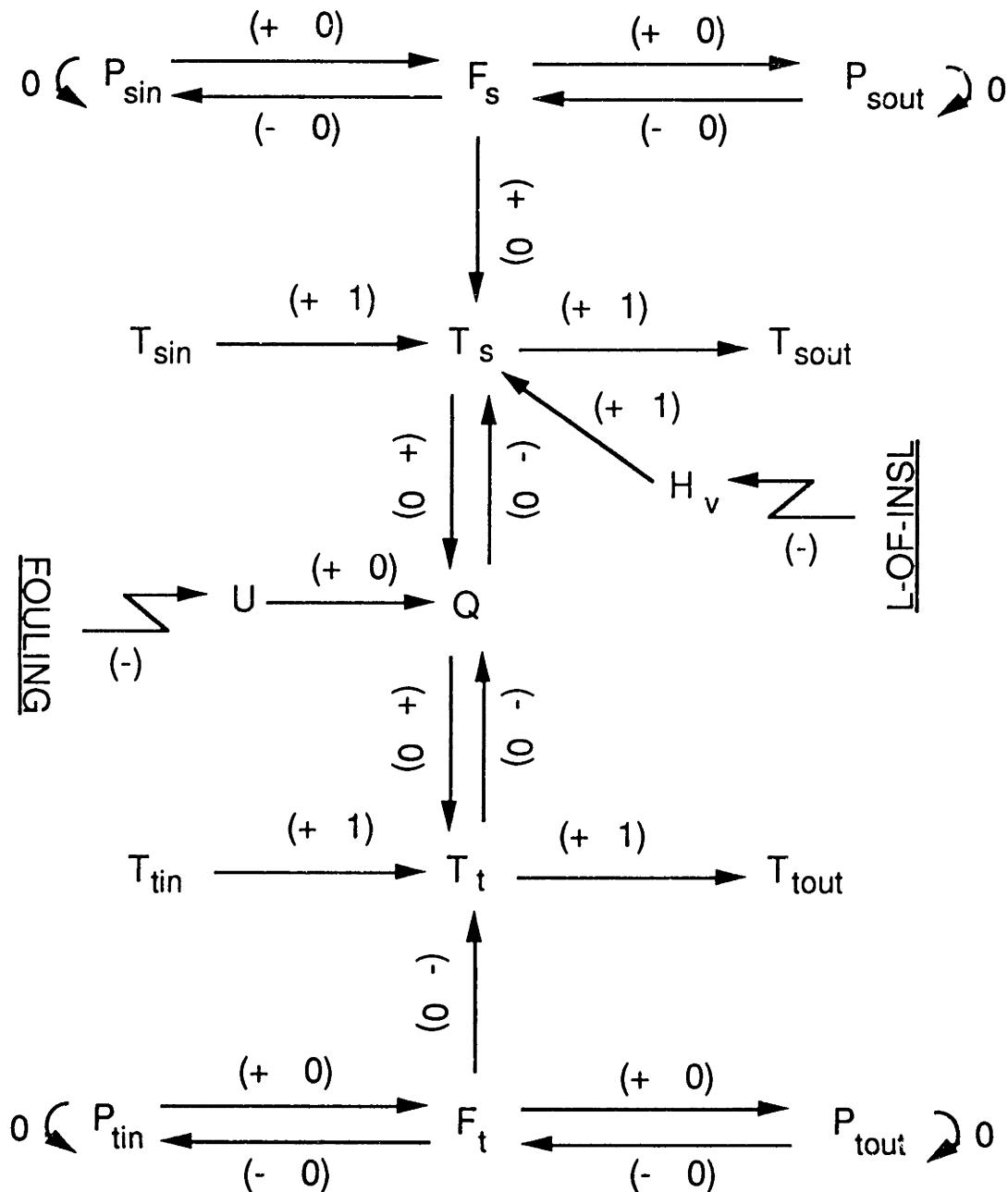


Fig. 4.4b. SDG of Shell and Tube Heat Exchanger

#### 4.5.5. Control Valve

We demonstrate the representation of disabling conditions associated with causal arcs by considering the SDG for a control valve.

A control valve can be modeled as a pipe with a variable resistance. The resistance is varied by the input signal,  $S_i$ , from the associated process controller. The input signal is externally specified with respect to the control valve. Variables describing the control valve include the valve stem position,  $S_p$  and total resistance to flow,  $R_t$ . Other variables are  $P_{in}$ ,  $P_{out}$ , and  $F$ .

The malfunctions considered are valve blockage, valve outlet leak and valve stem stuck at a fixed position (open or closed). The blockage and leak, require additional "independent" variables: fluid resistance,  $R_s$ , and leak flowrate,  $Q_l$ , respectively to model their effects. On the other hand, no additional variables are introduced to model the effect of the valve stem stuck open or stuck closed.

The algebraic equations describing flow-pressure phenomena of the liquid in the valve are

$$F = f(R_t) \cdot f^+(P_{in} - P_{out}) \quad (4.36)$$

$$F = F_{in} \quad (4.37)$$

$$F = F_{out} + Q_l \quad (4.38)$$

where  $F_{in}$ ,  $F_{out}$  are flowrates in adjacent units. By converting eqs. (4.36) through (4.38) to their original differential equations, SDG arcs may be derived as for the pipe.

Assuming the valve dynamics to be first order, the equations required to model the variable resistance in the pipe and the motion of the valve stem are

$$R_t = R_s + f^-(S_p) \quad (4.39)$$

$$dS_p/dt = f^+(S_i) \cdot f^-(S_p) \quad (4.40)$$

After specifying  $R_t$  as the output variable for eq. (4.39), SDG arcs are derived accordingly.

The SDG of the control valve is shown in Fig. 4.5. No additional parameters are introduced to model the effects of malfunctions resulting in a fixed valve stem position.<sup>9</sup> The primary deviation variable of these malfunctions is the valve stem position. Thus the SDG arc initiating from  $S_i$  and terminating at  $S_p$  includes these malfunctions as members of its **disabling conditions**.

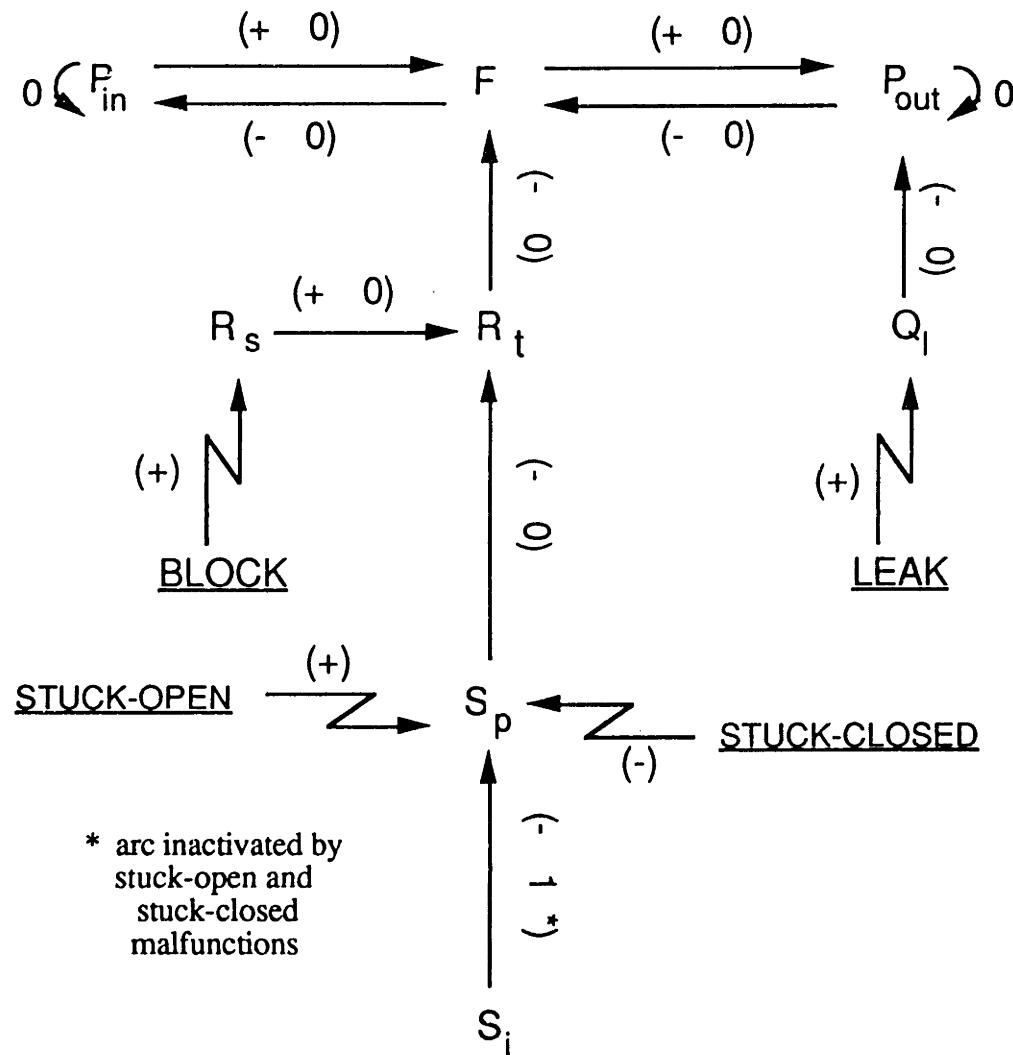


Fig 4.5. SDG of Control Valve

---

<sup>9</sup>Valve stuck open and valve stuck closed.

## 4.5.6. PI Controller

Controllers are examples of units performing computations in control loops. For a proportional-integral (PI) controller controlling variable  $X$ , the equations describing the operation of the controller are

$$E = X - X_{sp} \quad (4.41)$$

$$C = k_p \cdot E + k_i \int E \cdot dt \quad (4.42)$$

where  $X_{sp}$  is the set point,  $E$  and  $C$  the controller error and output signal respectively.  $X$  and  $X_{sp}$  are specified externally to the controller and  $E$  and  $C$  the output variables of the equations. The only failure we are concerned with is the controller output signal being stuck at a fixed high or low value.

Equation (4.42) contains an integral term and the first step is to transform the equations to standard state space form. This is done by augmenting the system of equations with an additional state variable. We define

$$\frac{dA}{dt} = E \quad (4.43)$$

Therefore

$$C = k_p \cdot E + k_i \cdot A \quad (4.42a)$$

The SDG for the augmented system is shown in Fig. 4.6. An integrator is associated with  $A$ . The controller output,  $C$ , is a **sensor signal** and this is depicted in the SDG by representing  $C$  in bold characters. The malfunctions affecting  $C$  result in  $C$  being stuck at a fixed value and has the effect of inactivating all SDG arcs terminating at  $C$ . This effect is modeled by including these malfunctions in the list of **disabling conditions** for these arcs.

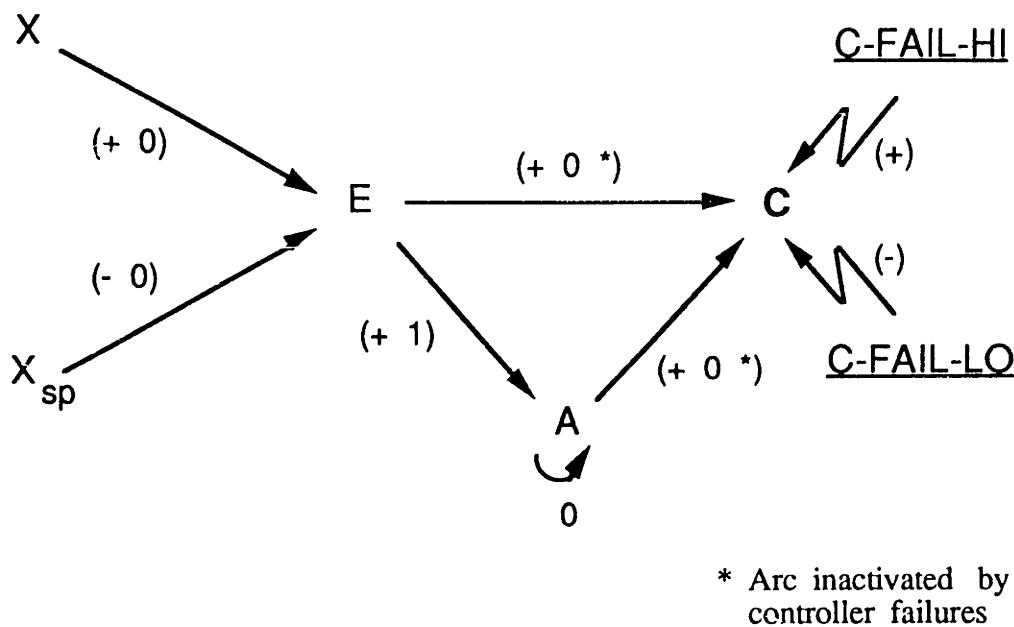


Fig. 4.6. SDG of Controller

## 4.6. Conclusion

In this chapter, a procedure for constructing the SDG of a process has been outlined. The approach taken is to decompose the process into its constituent units and then to develop SDG models for these units. Guidelines for selecting variables specifying unit behavior were presented. Rules for deriving the SDG from sets of differential and algebraic equations relating these variables were then developed. In developing unit SDG models, it is ensured that a consistent set of interface variables are defined, so models of adjacent units can be linked together.

These guidelines form the basis of unit SDG models developed for preparing the PM used in the MIDAS system. In order to make the MIDAS system more robust to variations in measurement thresholds and event orders,<sup>10</sup> values for the time delay attribute of some the causal arcs may be modified. In subsequent chapters, situations where modifications have been made to the delays are indicated.

---

<sup>10</sup>An event occurs when the value of a variable crosses a threshold.

## 4.7. Notation for Chapter 4

A,A <sub>C</sub>	area, cross sectional area
$\mathbf{A}$	system matrix
C	number of chemical species
C	controller output signal
C <sub>p</sub>	specific heat capacity
E	error
F	number of degrees of freedom
F	volumetric flowrate
f <sup>+</sup> f <sup>-</sup>	monotonically increasing, decreasing functions
g	gravitational acceleration constant
H <sub>V</sub>	insulation parameter
k <sub>i</sub> ,k <sub>p</sub>	controller constants
k <sub>r</sub>	reaction rate constant
L	liquid level
L	pipe length
M	mass
n	order of reaction
p	process parameter
P	number of phases
P,P <sub>b</sub>	pressure, vessel base pressure
Q	heat flux
Q <sub>l</sub>	leak flowrate
r	chemical reaction rate
R	number of independent chemical reactions
R <sub>S</sub> ,R <sub>t</sub> ,R <sub>X</sub>	resistance to flow, total resistance, outlet resistance
S <sub>i</sub>	input signal
S <sub>p</sub>	valve stem position
t,t <sub>d</sub> ,t <sub>m</sub>	time, time delay, measurement time scale
T	temperature
u	process input
U	heat transfer coefficient
V	liquid volume
x	state variable

$X, X_{sp}$	controlled variable, setpoint
y	unit output

### Special Symbols

$\Delta H_{rxn}$	heat of reaction
$\rho$	fluid density

### Subscripts

a,b	chemical species A, B
amb	ambient condition
in	inlet stream
out	outlet stream
s,t	shell, tubes

## 4.8. References for Chapter 4

1. Iri, M., K. Aoki, E. O'Shima and H. Matsuyama (1979). An algorithm for the diagnosis of system failure in the chemical process. Comput. & Chem. Eng. 3, 489-493.
2. Umeda, T., T. Kuriyama, E. O'Shima and H. Matsuyama (1980). A graphical approach to cause an effect analysis of chemical processing systems. Chem. Eng. Sci. 35, 2379-2388.
3. Palowitch, B.L. (1987). Fault Diagnosis of Process Plants using Causal Models. Chapter 3, Sc. D. Thesis, Massachusetts Institute of Technology.
4. Modell, M. and R.C. Reid (1983). Thermodynamics and Its Applications. Prentice-Hall Inc. Englewood Cliffs, N.J.

## **5. The Model Integrated Diagnostic Analysis System: An Overview**

The Model Integrated Diagnostic Analysis System (MIDAS) [1], is an operator aid for malfunction diagnosis in continuous chemical and refinery processes. The system acts as an operator associate rather than as a consultant, performing diagnostic analysis independent of operator input, utilizing only measurements from existing process sensors. Using artificial intelligence techniques, MIDAS specifically addresses problems not treated in previous diagnostic systems, including: non-monotonic process dynamics such as inverse responses and compensatory responses in control and other feedback loops, multiple faults and induced sensor failures, and out of order and false alarms.

MIDAS utilizes qualitative and quantitative process models in interpreting real-time sequences of abnormal events. By maintaining a separation between diagnostic inference and the process model, a plant independent inference strategy is used for diagnosis. Thus MIDAS is applicable to a wide range of industrial processes.

In Chapter 6, procedures for developing process models used in MIDAS are described in greater detail. The objective of the present chapter, is to enable the material in Chapter 6 to be presented in the proper context. In this chapter, the overall structure and basic features of MIDAS are described. The following presentation closely parallels those by Finch and Kramer [2 & 3].

### **5.1. Introduction**

Fault diagnosis entails all the steps involved in the identification of the root cause(s) of abnormal process conditions. The complexity of this task has led to increased interest in developing computer aids.

The automation of fault diagnosis has been thwarted in the past by:

- Practical limitations in preparation of accurate mathematical models and excessive computational requirements of conventional numerical methods.

- The inability in representing the symbolic reasoning of human diagnosticians.

One significant benefit of the artificial intelligence (AI) approach to fault diagnosis is the elimination of the symbolic bottleneck [4 & 5]. AI overcomes the symbolic bottleneck, using either "deep knowledge" or "shallow knowledge" techniques. In the shallow knowledge technique, detailed mathematical models and rigorous mathematical solution techniques are disposed of in favor of process specific diagnostic rules of thumb. In the deep knowledge technique, use is made of qualitative or approximate mathematical models and flexible logical inference techniques.

In spite of these advantages, there are several outstanding problems in the diagnosis of continuous process systems using AI techniques. These include:

- The incorporation of non-monotonic dynamic process behaviors in a process model and the utilization of these behaviors as a source of diagnostic information.
- Robustness of diagnostic conclusions to symptom variations. These variations are further magnified by the discretization of process information.
- The diagnosis of multiple malfunctions, particularly simultaneous independent malfunctions and induced sensor failures.

The Model Integrated Diagnostic Analysis System (MIDAS) is designed specifically to overcome these problems. MIDAS is deep-knowledge based system, that uses a qualitative process model derived from the Extended Signed Directed Graph (ESDG) of the process. The procedures used to derive this model are process independent and generally do not require process specific information. Use of the ESDG as a basis for the process model enables MIDAS to interpret non-monotonic dynamic behaviors and abnormal process transients. The qualitative model may be enhanced by incorporating process specific quantitative constraints, thus, improving diagnostic resolution. Features of the inference algorithm developed by Finch [6 & 7], reduce the sensitivity of MIDAS to symptom variations and give MIDAS the capability to diagnose multiple malfunctions.

In the following, we describe the basic features and the overall structure of MIDAS. In Section 5.2, certain aspects of the knowledge representation issues considered in the design

of MIDAS are introduced. A brief description of inference in MIDAS follows in Section 5.3. Finally, the current implementation of MIDAS is discussed in Section 5.4.

## 5.2. Knowledge Representation in MIDAS

Perhaps the most critical factor in the design of a successful computer program is the choice of representation of domain knowledge [8]. In a very abstract sense, all computer-based representation schemes are equivalent because they are embedded ultimately into patterns of bits in computer memory. Practically, however, some representation schemes are more powerful and offer both the developer and user of the program more convenience. In process engineering applications, both knowledge about process behavior and the problem solving strategy must be represented **explicitly** in the computer [9]. The explicit representation of both types of knowledge is one of the basic philosophies in artificial intelligence programming.

In MIDAS, the **physical** and **conceptual**<sup>1</sup> objects relevant to diagnostic problem solving are represented explicitly, using **frames**. Frames represent classes of objects and are implemented as generalized property lists. Property lists provide the facility to give a single entity different values, each associated with an explicitly named attribute. Individual members of a class are represented as frame instances. Mechanisms that allow frames and instances to inherit properties from one another make frames generalized property lists. These mechanisms provide convenient facilities to represent general "commonsense" knowledge about objects of the same type.

In the rest of this section, we introduce the frame object types and the programming methodology used in MIDAS. We begin by describing the programming methodology.

### 5.2.1. Programming Methodology

The programming methodology used in MIDAS is a combination of action-oriented (AOP) and object-oriented programming (OOP). Action-oriented programming refers to the conventional programming approach, where algorithmic procedures operate on the data structures (frames). Object-oriented programming derives from the ability to include

---

<sup>1</sup>For example, causal influences and diagnostic hypothesis.

algorithmic procedures as values for attributes of frame objects [10]. These procedures, referred to as **demons** are activated automatically when a value is needed for a property, or if the value of the property is changed. The power of the OOP paradigm is further enhanced by frames having message passing facility.

The frame-based representation and the combination of AOP and OOP is ideally suited for the diagnosis of dynamic physical processes for the following reasons:

- A correspondence between the structural and connectivity information of physical objects in the flowsheet and their representation in the computer is provided by the frame structure.
- Process independent behavioral knowledge, which largely consists of causal relationships among variables in individual units, is conveniently represented using frames.
- Diagnosis based on a causal understanding of process behavior is largely procedural. The diagnostic strategy is explicitly represented by the manipulation of frame attributes by procedural algorithms.
- The process of formulating and revising hypothesis in a dynamically changing environment is facilitated by message passing and demons associated with conceptual objects.

In the rest of this section, we describe the major categories of objects used in MIDAS: monitors, the process model and the hypothesis model. The relationship of these objects with MIDAS's inference algorithm (the event interpreter) is shown in Fig. 5.1.

### 5.2.2. Monitors

The purpose of monitors is to translate numerical process data into discrete events (alarms).<sup>2</sup> Two types of monitors are used in MIDAS: **Sensor Monitor(s)** and

---

<sup>2</sup>In this context, alarms are used to indicate deviations from abnormality and do not necessarily refer to potentially unsafe situations.

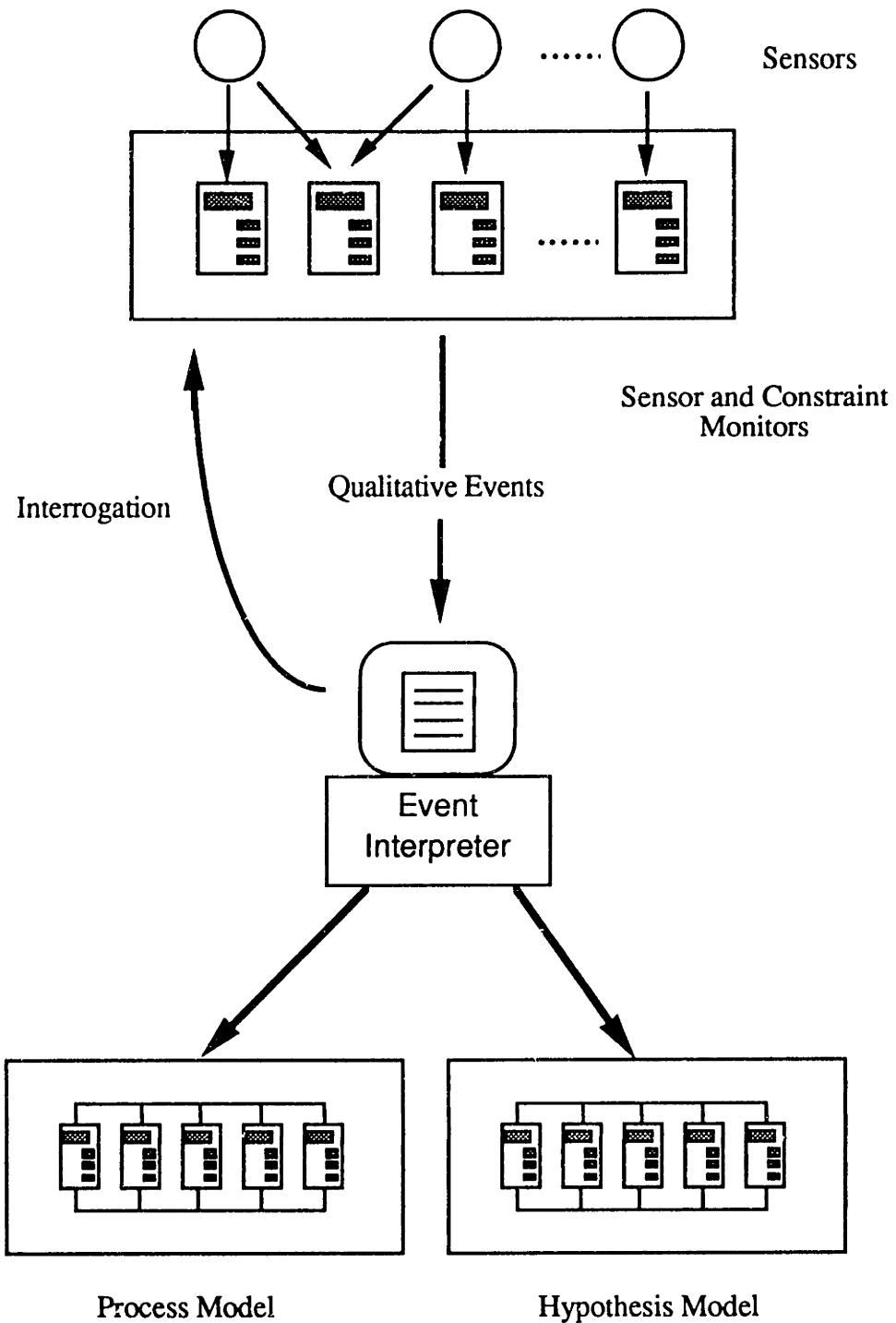


Fig. 5.1. Relationship of MIDAS Objects to Event Interpreter

**Constraint Monitor(s).** Sensor monitors collect and analyze data from on-line process sensors, while constraint monitors collect and analyze constraint residuals computed from sensor data. Constraint monitors are the means by which MIDAS exploits information about quantitative constraints. Monitors exist as objects in the MIDAS knowledge base, and there is one monitor for every sensor or constraint relation.

To detect events from process data, nominal values and trajectories for the measured variable or constraint are used as bases for comparison with on-line sensor data. Statistical criteria which take into account normal process variability produced by measurement noise and ordinary transients are used in the above analysis. Events related to gross sensor failures are also produced using range and noise limit checks.

Two types of analysis are performed by the monitors: **evaluation** and **prediction**. Evaluation applies tests similar to statistical quality control tests [11] to identify abnormal shifts in values or trends from vectors of measurement data. Prediction forecasts future values to determine if events can be expected to occur. Prediction by the monitors allows MIDAS to be robust to variations in the order of events.

MIDAS monitors are implemented as frames with demons invoking procedural code when new data is received. When the monitor determines that a new event has occurred, the event interpreter is activated as discussed in Section 5.3. A detailed description of the implementation of monitors is given by Finch [6].

### 5.2.3. Process Model

The process model (PM) is the means by which knowledge about process behavior is represented in MIDAS. Two types of models are used: **qualitative** and **quantitative**. The qualitative model describes causal relationships among events associated with measured variables and a set of enumerated root causes. General purpose algorithms, described in detail in Chapter 6, are used to produce the qualitative model from component causal models and the process flowsheet. The quantitative model relates events associated with process specific constraints to the root causes. A manual procedure, outlined in Chapter 6, is used to derive quantitative constraint information. The process model is constructed off-line and undergoes no structural modification during diagnosis.

Objects of the process model are also implemented as frames. The following objects comprise the PM:

**Measured Variable(s) and Measured Constraint(s):** Information contained in these objects include the current status (e.g. FAILED) and history of current and past states (e.g. HIGH) and trends (e.g. INCREASING).

**Potential Event(s):** Potential event objects store information pertaining to possible changes in status, state or trends of measured objects.

**Potential Root Causes(s):** Potential root cause objects enumerate the potential malfunctions in the process. Prior probabilities associated with the malfunctions are included in the list of attributes of these objects.

**Local Cause Link(s):**<sup>3</sup> Local cause links define the potential events expected first when a malfunction occurs.

**Compiled Causal Link(s):** These define directional links between potential event objects. Diagnostic conditions that modify the diagnosis if the link is invoked during inference are included attributes list of compiled causal links. These conditions represent a set of situation-action inferences and are analogous to rules in a rule based expert system.

#### 5.2.4. Hypothesis Model

The hypothesis model (HM) is the means by which the current diagnostic hypothesis is recorded. Objects of the HM are constructed on-line by the MIDAS inference algorithm during diagnosis.

Objects of the HM include events that have been observed and root causes that have been hypothesized. These objects mirror objects in the PM. This separation improves the clarity of MIDAS. In addition to improving clarity, the separation ensures that problems associated with implementing complicated truth maintenance mechanisms are avoided.

---

<sup>3</sup>Local cause links are implemented as attributes of potential root causes, measured variables and constraints.

The HM consists of the following types of objects:

**Recorded Event(s):** A new recorded event is created on detection of an event by its associated monitor. Recorded events are classified as primary (source)<sup>4</sup> or secondary events of inferred malfunction clusters.

**Hypothesized Root Cause(s):** The attributes of these objects include recorded events that support and oppose the hypothesized root cause and the relative likelihood of the object to other hypothesized root causes.

**Inferred Malfunction(s):** These clusters represent all recorded events and associated malfunctions that may be causally connected. Maintaining separate clusters allows MIDAS to diagnose multiple malfunctions. Attributes of inferred malfunctions include recorded events and hypothesized root causes associated with the cluster.

### 5.3. Inference in MIDAS: The Event Interpreter

The diagnostic inference methodology implemented in MIDAS is based on an idealized model of the deductive reasoning process of an operator performing diagnosis in a dynamic environment. The Event Interpreter (EI) consists of a set of procedural algorithms that construct and dynamically update hypotheses for the cause(s) of events. When a new event is detected, the EI attempts to relate the event to existing hypotheses, utilizing knowledge of relationships defined in the PM. At any point in time, the state of diagnosis is:

- A hypothesis concerning the number of independent malfunctions, and
- A list of hypothesized root cause candidates with their relative rankings, for each malfunction.

A detailed description of the EI appears in [6]. Figure 5.2, is a condensed flowchart of its inference cycle. The interpreter is activated by the detection of a new event by the monitors. The first action following detection, is the creation of a new recorded event object for the event.

---

<sup>4</sup>Source events are events that explain other recorded events in the cluster. They are used to determine if new hypothesized root causes should be created.

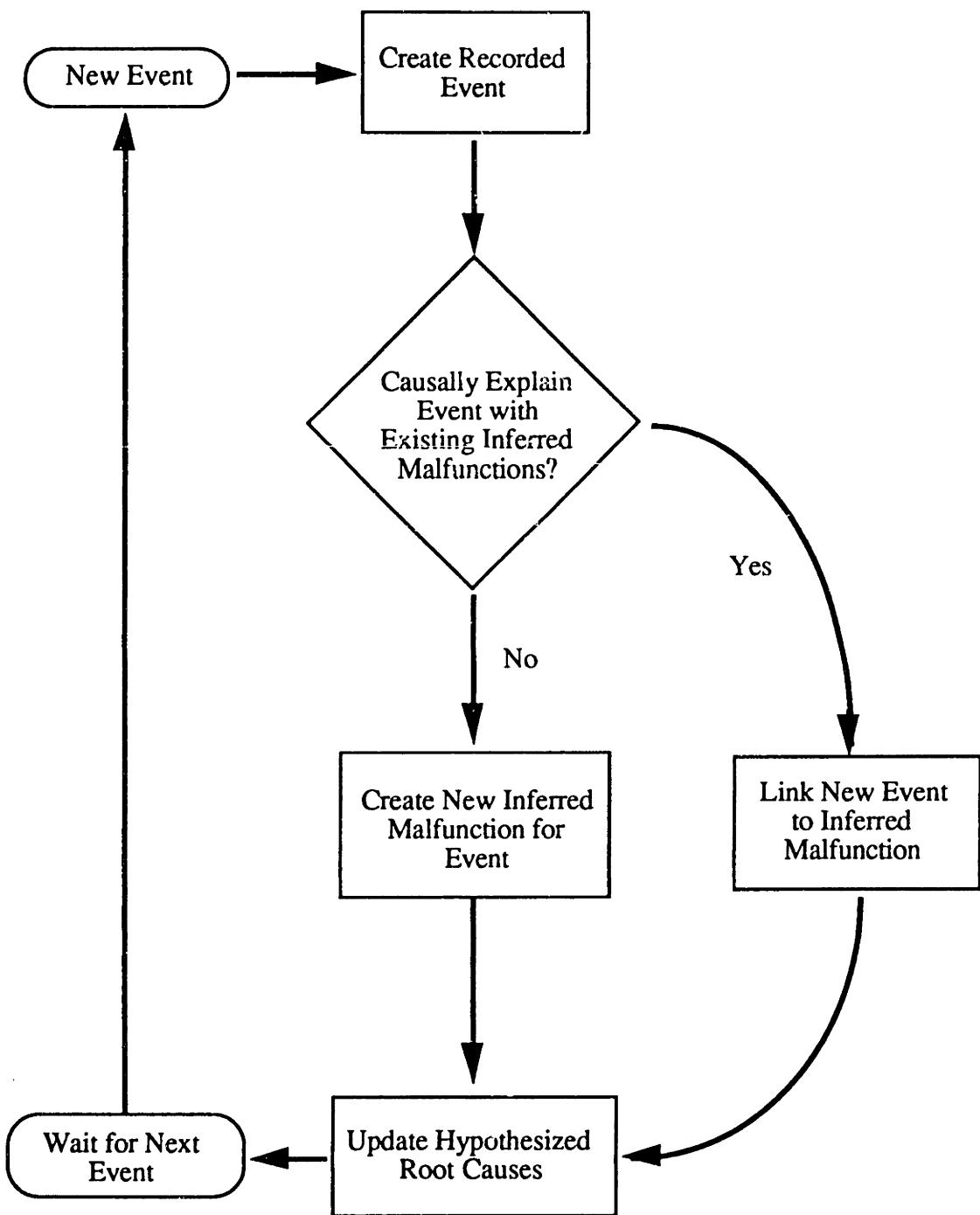


Fig. 5.2. Flowsheet of MIDAS Inference Cycle

Next, the interpreter searches the PM to determine whether the event is a result of an existing malfunction, or is indicative of a new malfunction. Searches are made for compiled causal links relating the existing event to existing recorded events and local cause links between the new event and active hypothesized root causes. If the event can be linked to an existing event or hypothesized root cause, the event is assumed to be a consequence of an existing malfunction,<sup>5</sup> and the new event joins an existing malfunction cluster. Conditions associated with the link are used to update the state of the diagnosis. If the search is unsuccessful, a new malfunction is assumed present and a new inferred malfunction is created to explain the event.

Following this step, the interpreter determines if new hypothesized root causes should be created for the current malfunction cluster. If the new event is determined to be a source event, hypothesized root causes are created for all previously unhypothesized local causes of the event, and added to the cluster.

Finally, the likelihood rating of all hypothesized root causes in the affected cluster are re-evaluated. This involves applying one of several formulae using the prior probability and the supporting and opposing evidence of each hypothesized root cause. The supporting and opposing evidence are events in the cluster that support and oppose a hypothesis, respectively, based on the existence, or lack of, a causal pathway between the root cause and the event. Ranking the hypothesis, completes the cycle and the EI waits for the next event.

## 5.4. Implementation and Program Architecture

MIDAS is implemented on a 386-class PC, using frame structures in Goldworks<sup>6</sup> to represent process knowledge. LISP routines are used to derive the PM by linking component unit models according to the flowsheet topology. Inference is accomplished using a combination of custom LISP routines, demons and message passing. The structure of the MIDAS software package is shown in Fig. 5.3.

---

<sup>5</sup>MIDAS assumes that a minimal set of malfunctions is the cause of all observed events.

<sup>6</sup>Trademark, Gold Hill Computers, Cambridge, MA.

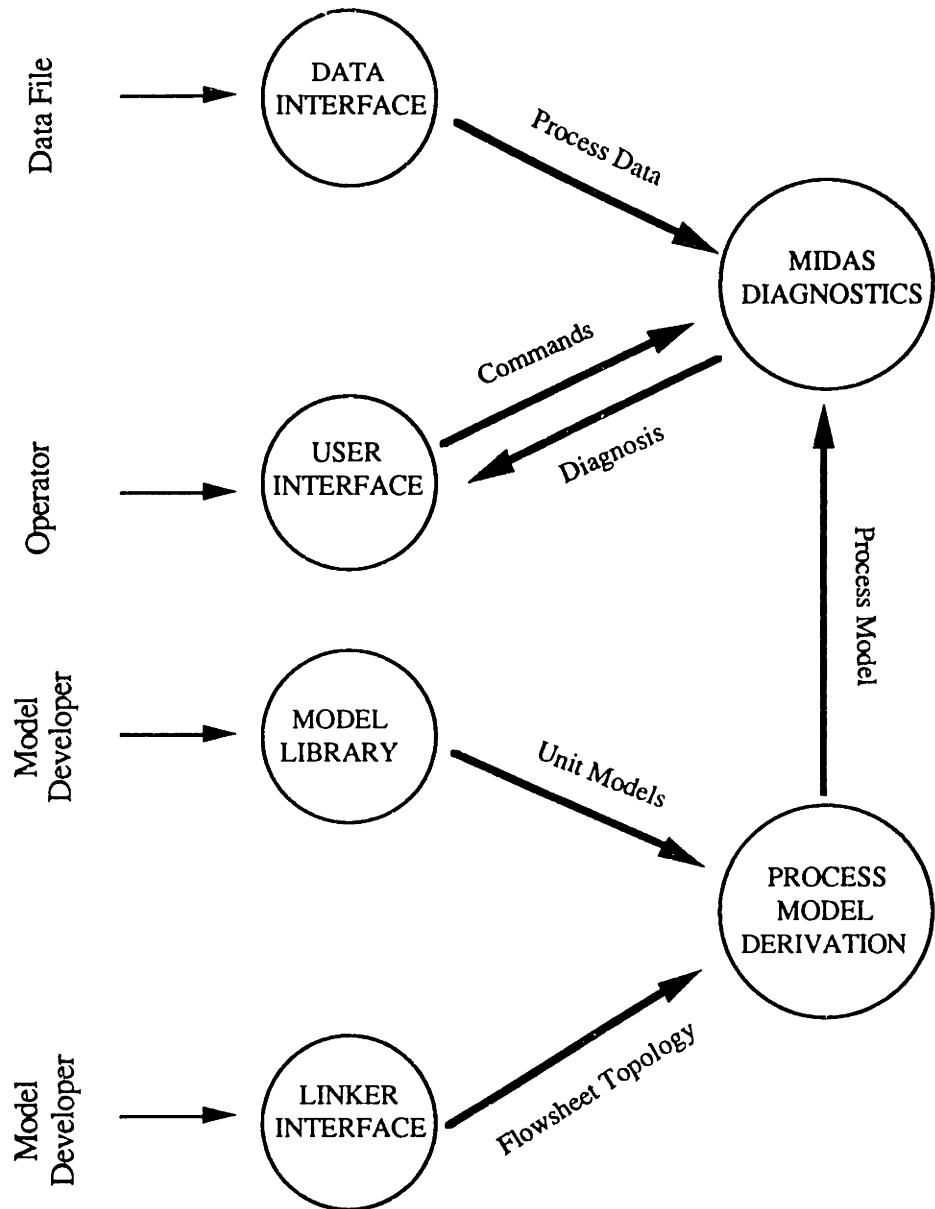


Fig. 5.3. Structure of MIDAS Software Package

A data acquisition interface allows MIDAS to access DBASE III<sup>7</sup> records. A menu-driven user interface is used to present diagnostic conclusions and to inspect information contained in PM, HM and monitor objects. Adjustable inference parameters can also be modified from the user interface.

---

<sup>7</sup>Trademark, Ashton-Tate, Torrance, CA.

## 5.5. Conclusion

In this chapter, the basic structure and features of MIDAS were described. MIDAS was specifically designed to handle some of the more complicated aspects of diagnosis in dynamic environments. Namely, complex non-monotonic process dynamic behavior, multiple simultaneously occurring malfunctions and robustness to variability in the malfunctions' symptoms. The aspects of the process model and inference algorithm that give MIDAS this capability were discussed, briefly. More detailed discussions of the process modeling and inference techniques appear in Chapter 6 and [6], respectively.

## 5.6. References for Chapter 5

1. Kramer, M.A., F.E. Finch and O.O. Oyeleye (1989). Operating Manual and User's Guide for MIDAS (version 1.0a), Laboratory for Intelligent Systems in Process Engineering, Massachusetts Institute of Technology.
2. Finch, F.E. and M.A. Kramer (1989). The handling of dynamics, multiple faults, and out-of-order alarms in the MIDAS diagnosis system. A.I.Ch.E. Spring National Meeting, Houston, TX.
3. Kramer, M.A. (1988). Automated diagnosis of malfunctions based on object oriented programming. J. Loss Prev. Process Ind., 1, 10, 226-232.
4. Banares-Alcantara, D. Siriam, V. Venkatsubramanian, A. Westerberg and M. Rychener (1985). Knowledge-based expert systems for CAD. Chem. Eng. Prog., 81, 9, 25-30.
5. Waterman, D.A. (1986). A Guide to Expert System, Addison-Wesley, Reading, MA.
6. Finch, F.E. (1989). Automated Fault Diagnosis of Chemical Process Plants using Model-Based Reasoning. Sc.D. Thesis, Massachusetts Institute of Technology.

7. Finch, F.E. and M.A. Kramer (1989). A new diagnostic algorithm for dynamic processes. Comput. & Chem. Eng., in preparation.
8. Winston, P.H. (1984). Artificial Intelligence, Addison-Wesley, Reading, MA.
9. Stephanopoulos, G. (1987). The scope of artificial intelligence in plant wide operations. In G.V. Reklaitis and H.D. Spriggs (Eds.), Computer-Aided Process Operations Elsevier, N.Y.
10. Winston, P.H. and B.K.P. Horn (1984). Lisp, Addison-Wesley, Reading, MA.
11. Shewhart, W.A. (1931). Economic Control of Quality in Manufactured Product, Van Nostrand, New York.

## 6. Process Modeling in MIDAS

In Chapter 5, a description of the overall structure of the Model Integrated Diagnostic Analysis System (MIDAS) was presented. MIDAS is a "deep knowledge" system using artificial intelligence techniques and is designed to treat specific problems not addressed in previous diagnostic systems: the representation and utilization of non-monotonic process dynamic information, diagnosis of multiple malfunctions and robustness to symptom variability.

In this chapter, we describe a structured approach for expressing process behavioral knowledge in the modeling formalism of MIDAS. In this approach, the "core" of the knowledge base consists of **qualitative** causal process relationships, and is derived directly from the process flowsheet. The qualitative knowledge base may be supplemented by specifying **quantitative** constraint relationships.

The procedures used in deriving the core of the knowledge base are algorithmic, process independent and can be implemented without using process specific knowledge. These procedures significantly increase the reliability and reduce the time and engineering effort required in developing diagnostic knowledge bases. The task of developing the core of the knowledge base is reduced to specifying the flowsheet structure and the signs of certain variable and parameter relationships, such as relative temperatures in adjacent process units. Working diagnostic systems for large flowsheets may be produced in hours or days, rather than weeks or months as is usually the case for conventional rule based expert systems (RBES).

### 6.1. Introduction

Malfunction diagnosis entails all the steps from the detection to the identification of the root cause(s) of abnormal process conditions. Human limitations in diagnosing failures in physical systems has led to increased interest in developing computer aids. In chemical process systems, the goals of assuring process safety, improving profitability, availability and product quality provide additional incentives for automated malfunction diagnosis.

One class of diagnostic aids is based on artificial intelligence (AI) programming techniques. The knowledge used by humans in diagnostic problem solving is often based on a causal

understanding of the domain [1]. A significant advantage of the AI approach is the ability to represent this type of symbolic information. AI overcomes the symbolic bottleneck, using either "deep knowledge" or "shallow knowledge" techniques. In the shallow knowledge approach, the knowledge base usually consists of a set of heuristic rules that explicitly relate a malfunction to its symptoms. Typically, these rules are derived by interviewing process experts. In the deep knowledge approach, the knowledge base consists of approximate quantitative models and "formal" qualitative<sup>1</sup> models of process behavior, developed from the underlying domain principles.

The Model Integrated Diagnostic Analysis System (MIDAS) [2] is a "deep knowledge" system for malfunction diagnosis in continuous chemical and refinery processes. The system is designed to address problems not treated in previous diagnostic systems, specifically:

- Representation of non-monotonic dynamic process behaviors in the knowledge base and utilization of these behaviors as a source of diagnostic information.
- Robustness of diagnostic conclusions to symptom variations, out of order events and false alarms.
- Diagnosis of multiple malfunctions, particularly simultaneous independent malfunctions and induced sensor failures.

One of the basic paradigms of artificial intelligence programming, is the explicit representation of both the domain knowledge required for problem solving, and the problem solving strategy. By maintaining a separation between diagnostic inference and knowledge about process behavior incorporated in a diagnostic model of the process, a plant independent inference strategy is used. Therefore, once behavioral knowledge about a process is expressed in the diagnostic process model (PM), inference in MIDAS is applicable to the process. A discussion of MIDAS's inference algorithm appears in [3].

The objective of this chapter is to present the approach and procedures used to derive the PM from the process flowsheet. The chapter is organized into several sections. In Section 6.2 an overview of the approach is presented. Desirable attributes of the approach, and

---

<sup>1</sup>Some of these modeling formalisms are described in detail in Chapters 2 and 3.

characteristics of the PM are highlighted. In Section 6.3, we introduce the PM as a representation that supports diagnostic reasoning in a dynamic environment. The following four sections, Sections 6.4 through 6.7 describe in detail, each major step in the derivation of the PM. Finally in Section 6.8, we conclude with a discussion about possible improvements to the procedures used in deriving the PM.

## 6.2. Deriving Process Models for MIDAS: An Overview

Process models used in MIDAS incorporate both **qualitative** and **quantitative** process information. Qualitative information is expressed as causal relationships<sup>2</sup> between a set of enumerated root causes and measured variables. Quantitative process information is represented as the relationships between process specific quantitative constraints and root causes.

We begin by summarizing the basic stages in constructing the PM. An overall flowchart is shown in Fig. 6.1. The first three steps are implemented using interactive algorithms, while input to the last step entails specifying relationships that are derived manually. The algorithms are implemented in Common LISP using the Goldworks Expert System Development Environment.<sup>3</sup> In subsequent sections, the algorithms and procedures used in each of these stages is described in further detail.

The first step in deriving the PM is to construct the single-stage Signed Directed Graph (SDG) of the process. The SDG [4], represents the immediate local causal influences between process variables. The process SDG is derived by linking unit SDG models from a model library according to the flowsheet topology.

---

<sup>2</sup>These can be more accurately described as relationships derived from causal considerations that reflect the temporal order of events associated with measured variables.

<sup>3</sup>Trademark, Gold Hill Computers, Cambridge, MA.

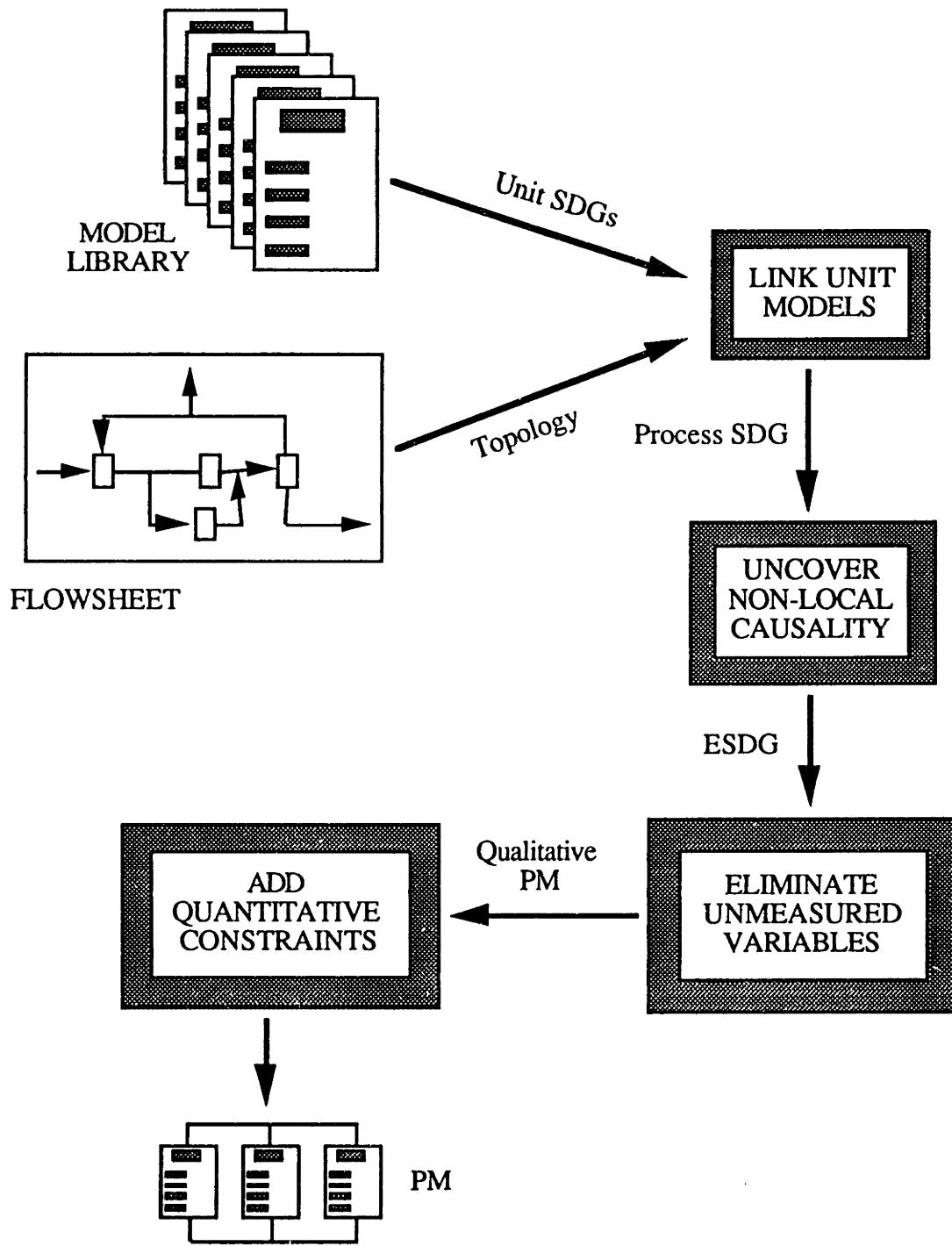


Fig. 6.1. Steps in Constructing the PM

Next, an algorithm for uncovering all "apparent non-local" causalities stemming from **global** interactions in the flowsheet is applied. These non-local causalities result in process variables exhibiting non-monotonic dynamic behavior, specifically, inverse, control system and other compensatory responses.<sup>4</sup> Non-local causality is represented by the addition of causal arcs to the SDG, resulting in the process Extended Signed Directed Graph (ESDG). The algorithm for deriving the ESDG from the SDG is based on the global topological analysis of the SDG presented in Section 3.3 of Chapter 3.

Following this step, another algorithm converts the ESDG to qualitative relationships in the PM, by eliminating unmeasured variables and coalescing causal pathways. During this stage, interactive input may be used to resolve ambiguities resulting from merging pathways of opposing sign. These qualitative relationships serve as the **core** of the process knowledge base for MIDAS.

Finally, the PM may be enhanced by incorporating knowledge of relationships between root causes and process specific quantitative constraints. Typical constraints include, conservation relationships and process equipment performance specifications. The constraints are entered manually by the developer of the PM and usually require knowledge of process specific numerical parameters. At this stage, relationships between the results of visual inspections performed by process operators, and the existence or inexistence of malfunctions may be expressed. These relationships supplement the qualitative and quantitative relationships in the PM.

### **6.2.1. Desirable Attributes of the Modeling Approach**

In general, the process modeling approach used in deep knowledge systems has several advantages when compared to the shallow knowledge RBES approach. Furthermore, the PM used in MIDAS has additional desirable attributes when compared to several other deep knowledge systems. What follows is a brief discussion of these characteristics.

---

<sup>4</sup>Inverse response occurs when the initial and ultimate direction of change of a variable are opposite. Compensatory response is exhibited, if the variable returns to its initial value after all transients have died out.

### *Reliability*

In RBES, the procedures used in developing a set of rules for a particular application is unstructured and often ad hoc. Additionally, the rules are sometimes derived without consideration to the principles governing the domain. As a result, the rule set may be incomplete or unreliable, and cannot be easily verified [5]. On the other hand, the PM is derived from models based on underlying domain principles. Moreover, the approach used is structured. In particular, procedures for deriving the qualitative core of the PM are algorithmic. Thus knowledge bases (PMs), derived using this approach are more easily verified for completeness and consistency when compared to rule sets in conventional RBES.

### *Engineering Effort*

Obtaining diagnostic knowledge by interviewing domain experts could easily take months or a few years for large process flowsheets [6]. Once units from the model library have been connected according to the flowsheet structure, the time taken to develop the PM may be reduced to several hours or days, using the above approach. Additionally, the time and expense required for process parameter identification when using conventional mathematical modeling approaches is avoided.

### *Process Generality*

Shallow knowledge systems tend to suffer from a lack of generality. Each process typically requires the development of a unique rule base, even for minor changes in plant configuration. This is a critical problem, as, upwards of 10,000 rules may be required in a large plant if generic rules are not used [7]. The approach described above accommodates changes in process configuration. Changes in configuration are addressed by linking unit models according to the new flowsheet structure and then simply applying subsequent procedures.

## *Graceful Degradation*

Humans have the important quality that when faced with unfamiliar circumstances, they offer reasonable guesses or partial solutions rather than failing abruptly. Diagnostic systems should have a similar property. One limitation of RBES is that they fail when faced with novel situations, as the rules in the system incorporates only knowledge derived from past experiences [8]. Once the causal interactions in the process and the primary effects of the actual malfunction have been modeled correctly, MIDAS will always include the malfunction in its hypothesis set.

## *Process-Specific Information*

The core of the PM consists of qualitative relationships. The necessary information needed to derive these relationships using the model library consists of:

- The structural connectivity of units in the flowsheet.
- Knowledge of the signs of parameter relationships, such as relative stream temperatures in adjacent units.

This information is obtained without much effort for most processes. Unlike conventional linearized models, the qualitative model is usually applicable to the same process over a variety of operating conditions and to several processes with the same structural connectivity.

One benefit of the approach used in deriving the PM is that the final PM can be tailored to the amount of available process-specific quantitative information. The value of quantitative information in diagnosis is well established [9, 10, 11]. Two principal means exist for incorporating quantitative information in the PM. First, qualitative ambiguities which occur as a result of merging causal pathways may be resolved by the developer of the PM. The model may be further enhanced by the addition of quantitative constraints. The integration of qualitative and quantitative constraint models is facilitated by the event-oriented representation of the PM [3]. Thus, a family of models is produced by utilizing varying amounts of quantitative information.

## *Sensor Placement*

Derivation of qualitative relationships in the PM starts with models containing variables that completely describe the qualitative state of the system. Unmeasured variables are subsequently removed during the procedure. This ensures that qualitative relationships may be derived independently of the number and location of sensors in the process.<sup>5</sup>

## *Unmeasured Variable Values*

Another desirable attribute of removing unmeasured variables during model development, is that diagnosis may be performed without knowledge of their values. This potentially improves the efficiency of on-line diagnosis, as assumptions about the values of these variables do not have to be postulated and retracted during diagnosis.<sup>6</sup>

## *Malfunction Diversity*

Several diagnostic systems are limited in the variety of malfunctions they diagnose. Some assume all sensors are functioning correctly, and diagnose malfunctions in process units [12]. Others are limited to diagnosing sensor malfunctions, only [13]. In MIDAS, the set of malfunctions that may be diagnosed include malfunctions associated with process equipment degradation and failure, and external process disturbances. Specific examples of these types of malfunctions are heat exchanger fouling and leaks and feedstock and utility variations, respectively. MIDAS also diagnoses failures and biases in process sensors and their associated control system units. All these malfunction modes are associated with the local unit models and are subsequently represented in the PM.

## *Dynamic Information*

Most diagnostic systems perform diagnosis based on "snapshots" of process states at a single time point. Since processes can exhibit non-monotonic dynamic behaviors such as

---

<sup>5</sup>The ability to derive quantitative constraints is limited by sensor location.

<sup>6</sup>However, results of visual inspections may be used as a source of diagnostic information.

inverse and compensatory responses, performing diagnosis by interpreting snapshots is potentially misleading. In order for MIDAS to interpret non-monotonic behaviors correctly, this information must be represented in the PM. These behaviors occur as a result of local and **global** causal interactions in the process. The topological analysis performed during the second stage of model development correctly identifies situations where such non-monotonic behaviors may occur. This information is subsequently represented in the PM and is utilized as a source of diagnostic information.

### *Accuracy and Resolution*

By far the **two most important characteristics** of any diagnostic system are that the system:

- Provide an accurate diagnosis, including the true malfunction origin.
- Produce a diagnosis with a high degree of resolution, presenting a "minimal" set of malfunction candidates.

The ability of the MIDAS to diagnose accurately with sufficient resolution is inherent in the PM, as the model incorporates all possible process behaviors. Previous disturbance propagation assumptions used in determining possible process behaviors from the SDG:

- Limit the production of spurious non-monotonic dynamic behaviors that may result from global process interactions.
- Exclude certain situations where non-monotonic behavior is exhibited as a result of global causal interactions.

The topological analysis performed in MIDAS identifies all situations where non-monotonic behaviors may arise due to global interactions. Furthermore, only situations where such behaviors may arise are indicated. Thus, the PM facilitates an accurate diagnosis without unnecessarily degrading resolution. Resolution may be further enhanced by the utilization of quantitative constraint information. Performance of MIDAS with respect to these criteria is quantified in the context of a case study of an example process in Chapter 7.

## 6.3. The MIDAS Process Model

The PM is the means by which knowledge about process behavior is represented in MIDAS. It is constructed off-line by the procedures described in the previous section, and undergoes no structural modification during diagnosis.

In this section, we discuss the structure of the PM. In Section 6.3.1., features of the PM used in representing process dynamic behavior, and for supporting diagnostic inference are introduced using a simple process as an example. In the Section 6.3.2., the frame-based representation of the PM is described in detail.

### 6.3.1. Representing Dynamic Process Behavior using the PM

The PM supports the deductive procedures of the MIDAS inference algorithms. Two significant features of diagnostic inference in MIDAS are [3]:

- Values of unmeasured variable states are not used in inference.
- The diagnostic hypothesis is continuously updated in response to incoming process information.

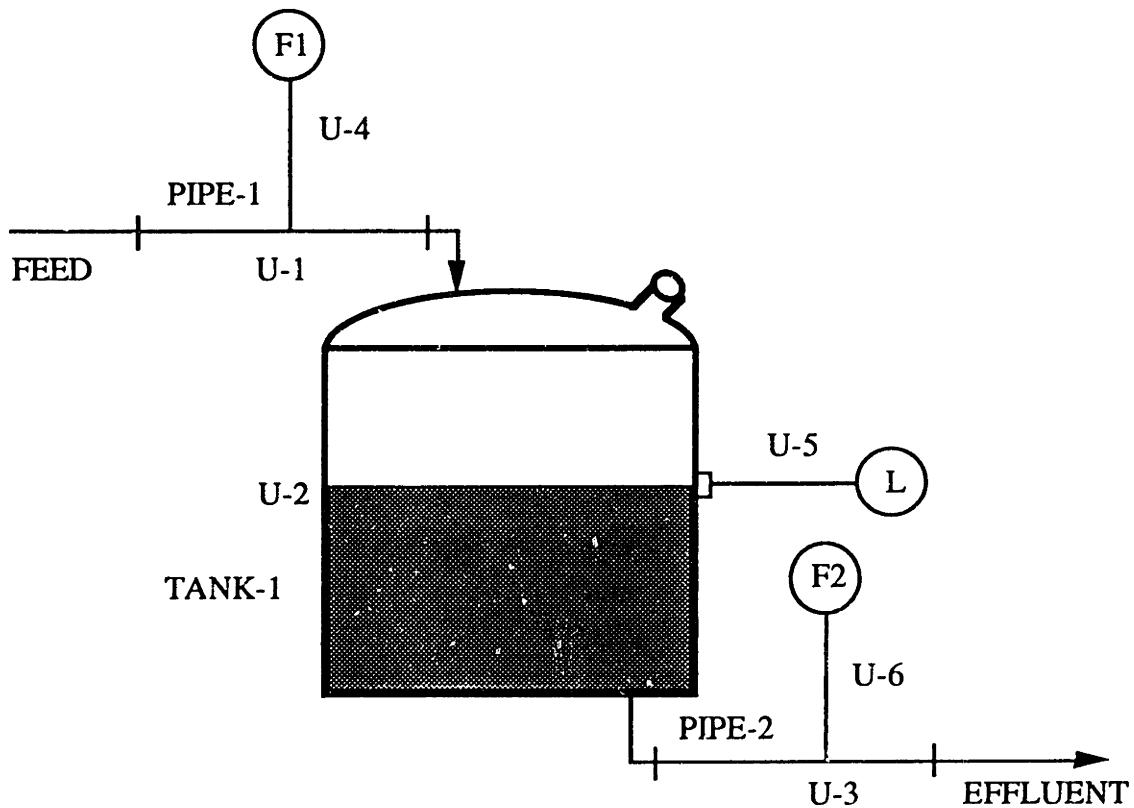
These features place the following requirements on the PM:

- The PM relates malfunctions to measured states (and status) only.
- **Directional relationships** from which the current state of the hypothesis can be updated, must exist between the measured variables states (and status).

We introduce the PM by considering how process dynamic information may be utilized in malfunction diagnosis of a simple process. Figure 6.2 shows the process schematic of a gravity flow tank process. The following sequence of events<sup>7</sup> are expected to occur as a result of a partial blockage in the outlet pipe.

---

<sup>7</sup>An event is defined as a transition between the qualitative states or status of a measured entity.



**Fig. 6.2. Gravity Flow Tank**

- (1) f2: normal → low
- (2) l: normal → high
- (3) f2: low → normal

F2 displays compensatory response and its value returns to normal after all process transients have died out.

After the first event, the current hypothesis consists of the following four malfunctions:<sup>8</sup> **tank-1-outlet-leak, tank-1-blockage, pipe-2-blockage and f2-sensor-failed-**

---

<sup>8</sup>Other malfunctions that result in the same event have been omitted for the sake of clarity.

low. On observing the next event, the additional information is used to eliminate **tank-1-outlet-leak** and **f2-sensor-failed-low** as possible root causes, as either of the two malfunctions cannot lead to the observation of **I high**. When **f2** returns to normal, the new information does not refine the current diagnosis. This is because both remaining malfunctions can lead to compensatory response of **f2**. Therefore, the final diagnosis is: **tank-1-blockage or pipe-2-blockage**. However, should another event occur, for example,

(4)    **I:**    **high**     $\rightarrow$     **normal**

the diagnosis is classified as being the result of a transient, as **I** does not exhibit compensatory response to any malfunction in the process.

This information may be represented in the state transition graph of Fig. 6.3a. An important point to note about this graph is the explicit representation of dynamic information. This is accomplished using links connecting state transitions associated with the same variable or links connecting state transitions of different variables. The diagnostic actions attached to the links provide a convenient means for updating the diagnostic hypothesis during process transients. The links may be interpreted as "situation-action" inferences, whose actions are applied if the situation associated with the link matches the current situation in the process. Each of these links is **analogous** to a rule in a conventional RBES.

The qualitative core of the PM derived using the algorithms described in Section 6.2. closely parallels the state-transition graph, and is shown in Fig. 6.3b.<sup>9</sup> The major difference between the graphs is that (i) the links between state-transitions are replaced by links between the consequent state resulting from the transition, and (ii) conditions under which the link may be activated/deactivated<sup>10</sup> are represented.

---

<sup>9</sup>The PM produced by the algorithms contains other links and associated conditions. A minimum set of links and conditions that produce the diagnosis for this scenario is shown for clarity.

<sup>10</sup>The condition at which the link between L: (H) and L: (N) may be deactivated are not shown for clarity. This condition is F2: (N) or F2: (H)

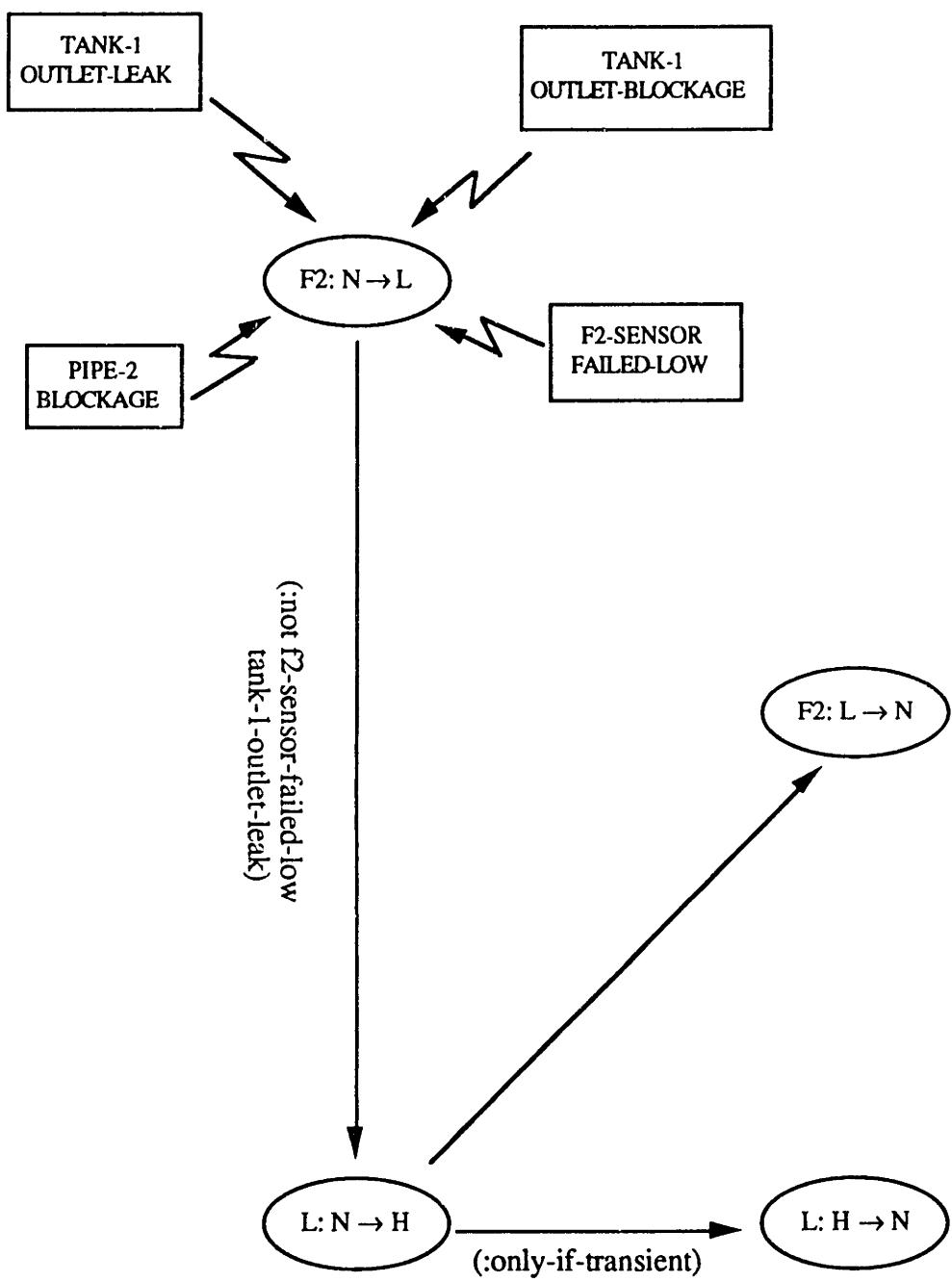


Fig. 6.3a. State Transition Graph for Gravity Flow Tank

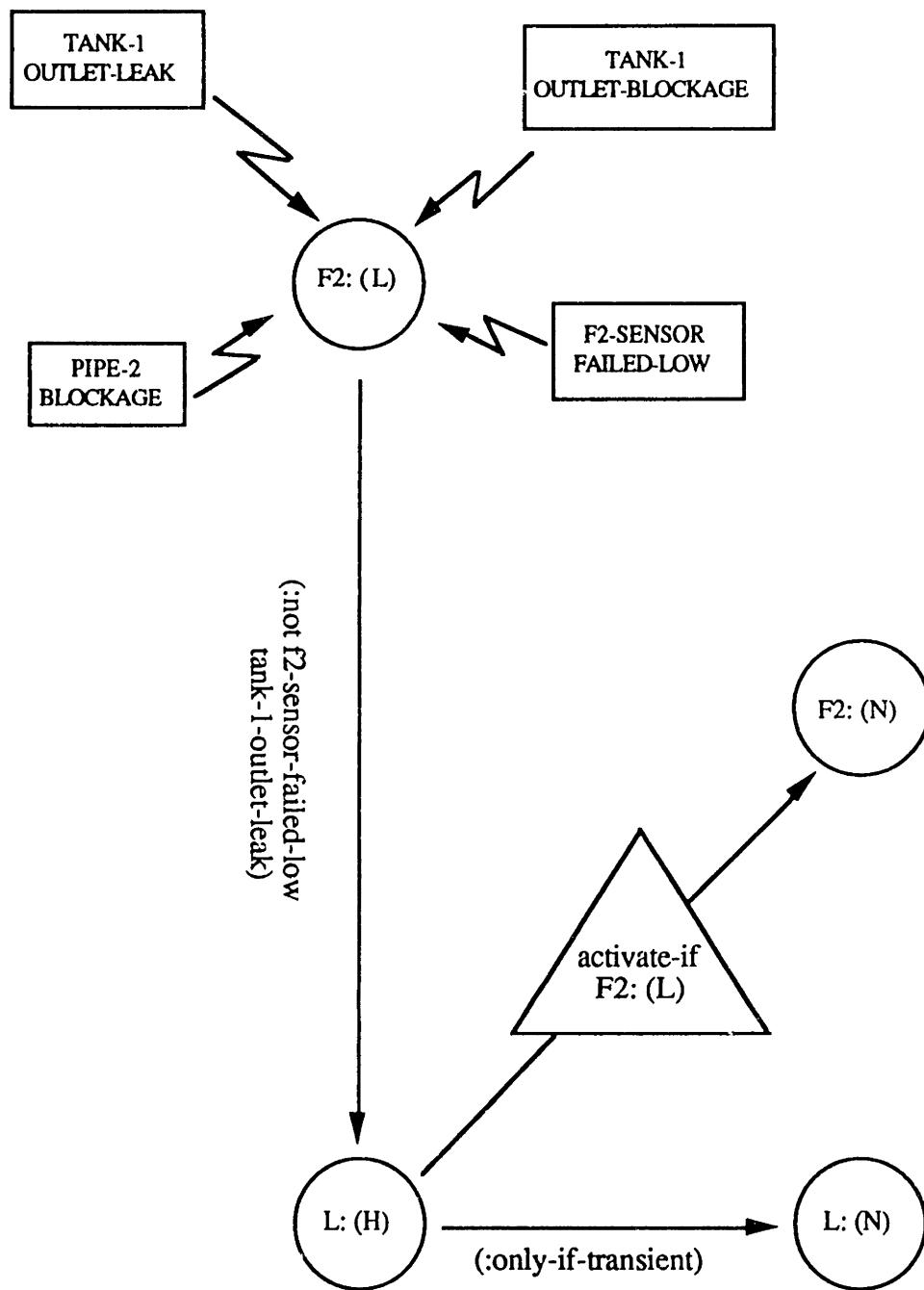


Fig. 6.3b. PM for Gravity Flow Tank

## 6.3.2. Description of Process Model Objects

In the previous section, the PM, depicted as a graph, was introduced as a means for representing process behavioral knowledge. In addition to **qualitative** causal relationships between a set of enumerated malfunctions and measured variables, the PM also represents knowledge about relationships between **quantitative** constraint residuals and the malfunctions.

In MIDAS, the graph is implemented using the Goldworks frame-based knowledge representation scheme. Frames are used to represent both **physical** objects, e.g. malfunctions; and **conceptual** objects, e.g. links; in the PM.

Frames represent classes of objects, and are implemented as generalized property lists. Individual members of a class are represented as frame instances. Property lists provide the facility to give a single entity different values, each associated with an explicitly named attribute. Among the attributes of a frame instance are procedures, known as **demons** which are activated automatically when a value is needed for an attribute, or if the value of the attribute is changed. Mechanisms that allow frames and instances to inherit properties from one another, make frames generalized property lists. These mechanisms provide convenient facilities to represent general knowledge about objects of the same type.

In addition to PM objects, algorithms that derive the PM also create other objects used for diagnosis in MIDAS. These objects include:

**Monitor object(s):** receive process data input from the MIDAS data interface.

**Screen interface popup-choose objects(s):** are part of the MIDAS menu-driven user interface. They are used to inspect values contained in PM and monitor objects during diagnosis. Popup-choose objects are described in [14].

In the following, we describe the objects comprising the PM. Attributes of these objects are defined in Tables 6.1 through 6.6. Attributes of screen interface and monitor objects specified by the algorithms are presented in Appendix E.

## *Potential Root Causes*

Potential root causes symbolize the set of enumerated malfunctions that potentially affect process units. Potential root causes are represented by rectangles in the PM of Fig. 6.3b. A description of the set of attributes of potential root cause objects is shown in Table 6.1.

## *Potential Events*

Potential events provide a uniform representation for representing qualitative and quantitative information about the state of the process. These objects indicate potential state or status changes associated with a measured variable or a measured constraint. The consequent state (status) of potential event objects are depicted by circles in the PM of Fig. 6.3b. Attributes of potential event objects are described in Table 6.2.

## *Measured Variables*

Measured variables represent observations of variables in the process. A list of attributes of measured variable objects appears in Table 6.3.

## *Measured Constraints*

Measured constraints are associated with observable constraint residuals in the process. Current values for measured constraints are computed from values of the corresponding measured variables. Attributes of measured constraint objects are shown in Table 6.4.

Table 6.1 Potential Root Cause Object

Slot Name	Data Type	Comments
APPLICABLE-TESTS	list	TESTS containing diagnostic information about the root cause.
ID	symbol	Name of POTENTIAL-ROOT-CAUSE instance.
PRIMARY-DEVIATION	list	EVENTS observed first as result of root cause. Events denoted by pair of MEASURED-OBJECT and consequent state/status.
PRIOR-PROBABILITY	real	Relative probability of root cause to other POTENTIAL ROOT CAUSES.

Table 6.2 Potential Event Object

Slot Name	Data Type	Comments
ACTIVE	symbol	YES or NO. Initially NO. When YES activates demon that initiates inference.
CONSEQUENT-STATE	symbol	State of MEASURED-OBJECT after EVENT. HIGH, LOW or NORMAL. NIL if EVENT does not involve status change.
CONSEQUENT-STATUS	symbol	Status of MEASURED-OBJECT after EVENT. OK or NORMAL. NIL if EVENT does not involve status change.
EXCLUSIVE-EVENTS	list	Other EVENTS associated with MEASURED-OBJECT.
ID	symbol	Name of POTENTIAL-EVENT instance.
OBJECT	symbol	Associated MEASURED-OBJECT instance.
PRIOR-STATE	symbol	State of MEASURED-OBJECT before EVENT. HIGH, LOW or NORMAL. NIL if EVENT does not involve status change.
PRIOR-STATUS	symbol	Status of MEASURED-OBJECT before EVENT. OK or NORMAL. NIL if EVENT does not involve status change.
TYPE	symbol	Type of MEASURED-OBJECT. VARIABLE or CONSTRAINT.

Table 6.3 Measured Variable Object

Slot Name	Data Type	Comments
CURRENT-STATE	symbol	Current state of MEASURED-VARIABLE.
EVENT-SET	list	EVENTS associated with measured variable.
ID	symbol	Name of MEASURED-VARIABLE instance.
LOCAL-CAUSES	list	POTENTIAL-ROOT-CAUSE instances listing MEASURED-VARIABLE as primary deviation.
PRECURSOR-LINKS	list	COMPILED-CAUSAL-LINK instances listing MEASURED-VARIABLE as result node.
SUCCESSOR-LINKS	list	COMPILED-CAUSAL-LINK instances listing MEASURED-VARIABLE as source node.
TYPE	symbol	Type of measured object. VARIABLE

Table 6.4 Measured Constraint Object

Slot Name	Data Type	Comments
CURRENT-STATE	symbol	Current state of MEASURED-CONSTRAINT.
EVENT-SET	list	EVENTS associated with measured constraint.
ID	symbol	Name of MEASURED-CONSTRAINT instance.
LOCAL-CAUSES	list	POTENTIAL-ROOT-CAUSE instances listing MEASURED-CONSTRAINT as primary deviation.
PRECURSOR-LINKS	list	COMPILED-CAUSAL-LINK instances listing MEASURED-CONSTRAINT as result node.
SUCCESSOR-LINKS	list	COMPILED-CAUSAL-LINK instances listing MEASURED-CONSTRAINT as source node.
TYPE	symbol	Type of measured object. CONSTRAINT.

Table 6.5 Compiled Causal Link Object

Slot Name	Data Type	Comments
ACTIVATE-IF	list	Conditions that activate the link if previously inactive. If NIL, link always active.
ACTIVE	symbol	Indicates link is currently active. YES or NO.
DEACTIVATE-IF	list	Conditions that deactivate link if previously active. If NIL, link always inactive.
ID	symbol	Name of COMPILED-CAUSAL-LINK instance.
NOT-CONDITIONS	list	POTENTIAL-ROOT-CAUSES contraindicated by link.
ONLY-IF-CONDITIONS	list	POTENTIAL-ROOT-CAUSES indicated by link.
RESULT-NODE	symbol	MEASURED-OBJECT affected by event.
RESULT-STATE	symbol	Consequent state of result-node. HIGH, LOW or NORMAL.
RESULT-STATUS	symbol	Consequent status of result-node. FAILED or OK.
SOURCE-NODE	symbol	MEASURED-OBJECT initiating the event.
SOURCE-STATE	symbol	Consequent state of source-node. HIGH, LOW or NORMAL.
SOURCE-STATUS	symbol	Consequent status of source-node. FAILED or OK.

Table 6.6 Test Object

Slot Name	Data Type	Comments
ACTIVE	symbol	YES or NO. Nominally NO. If YES activates inference cycle.
EXCLUSIVE-TESTS	list	Instances of mutually exclusive TESTS.
ID	symbol	Name of TEST instance.
NOT-CONDITIONS	list	POTENTIAL-ROOT-CAUSES contraindicated by link.
ONLY-IF-CONDITIONS	list	POTENTIAL-ROOT-CAUSES indicated by link.

## *Compiled Causal Links*

Compiled causal links represent directional links between potential event objects. During diagnosis, events indicating the change in state or status of a measured object are attributable to compiled causal links. Associated with compiled causal links are a set of actions used to modify the current diagnostic hypothesis if the link is invoked during diagnosis. Also included in the list of attributes of compiled causal links are conditions under which the link may be activated/deactivated. These are depicted by triangles in the PM of Fig. 6.3b. Attributes of compiled causal link objects are shown in Table 6.5.

## *Tests*

Tests are the results of visual inspections performed by the operator to confirm or deny the existence of a set of potential root causes. During diagnosis, inspection results may be entered manually via the operator interface. The attributes of test objects appear in Table 6.6.

In the following sections, we describe in more detail, the procedures and algorithms involved in deriving relationships represented in the instances of PM objects. Some attributes of PM objects are used solely to relate PM objects to one another and contain no information about causal or quantitative relationships. The derivation of these attributes is trivial, and we do not address them any further. In particular we omit the derivation of relationships between PM objects and monitor and screen interface objects. The focus is on deriving the relationships among root causes and process observations.

## **6.4. Creating the Process SDG**

The first step in developing the PM is to create the process SDG. In this section, the algorithm used to create the process SDG is described. Detailed instructions for using this algorithm are provided in Chapter 9. First, we discuss the representation of the SDG as frame object instances.

## 6.4.1. Description of Process SDG Objects

The process SDG represents the immediate **local** causal influences between malfunctions and variables in the process. Information in the SDG is represented using the following frame objects:

### *Units*

Units in the SDG symbolize separate physical entities in the process. Examples of process units are chemical reactors, pipes, heat exchangers and evaporators. Associated with units in the SDG are root causes and variables.

### *Root Causes*

Root causes represent the set of malfunctions that affect process units. The primary effect of a root cause is to cause a deviation of the value of a variable in the SDG. A root cause object must be defined in the SDG, for every potential root cause in the PM.

### *Variables*

Variables in the SDG symbolize process variables or parameters in the process. Some of the variables represent process observations available for diagnosis. For each variable representing a process observation, there is a corresponding measured variable in the PM.

### *Causal Arcs*

Causal arcs represent causal influences between variables and parameters in the process. An arc initiates at a variable in the SDG and terminates at another variable. In the SDG, causal arcs symbolize immediate local causal influences.<sup>11</sup> SDG arcs may be further classified as temporary arcs and standard (permanent) arcs, as described in Section 6.4.2.

---

<sup>11</sup>Additional arcs created to produce the ESDG represent non-local influences.

The list of attributes of objects comprising the SDG are defined in Appendix F.

## 6.4.2. Algorithm for Creating the SDG

A modular approach is adopted when creating the process SDG. A flowchart describing the algorithm for creating the SDG is shown in Fig. 6.4. The algorithm derives the SDG in two stages.

- Instantiating SDG objects for individual units.
- Connecting and linking unit SDG models according to the flowsheet structure.

Instances of SDG objects for process units are created, using algorithms in the unit model library. A top level driver routine allows the developer select a unit type and create SDG objects for an instance of the units type. Associated with each unit type, are interactive Common LISP algorithms which create instances of SDG objects for the unit in response to specified input. Instances of SDG objects for process units may be created in any order, provided instances particular to a unit are created before creating object instances for sensors measuring variables of the unit.

The model library algorithms implement the rules developed in Chapter 4, for deriving the SDG from differential and algebraic equation sets. An input specification language is provided to enter unit SDG models for unit types not included in the library. SDG models for these units are specified in batch files according to the input language.

The algorithms potentially create SDG models that represent generalizations of different realizations of the unit type. Further specialization for the particular unit instance is achieved when the model developer responds to questions for the unit. Different responses to questions for a unit type, results in different values for attributes of SDG objects, and possibly a different structure of the SDG.

For each unit the developer answers the following questions:

- A unique name describing the unit

- A unique identification number for the unit
- Other questions describing the phenomena occurring in the unit. For example, in chemical reactors qualitative information describing details of the reaction network should be specified.

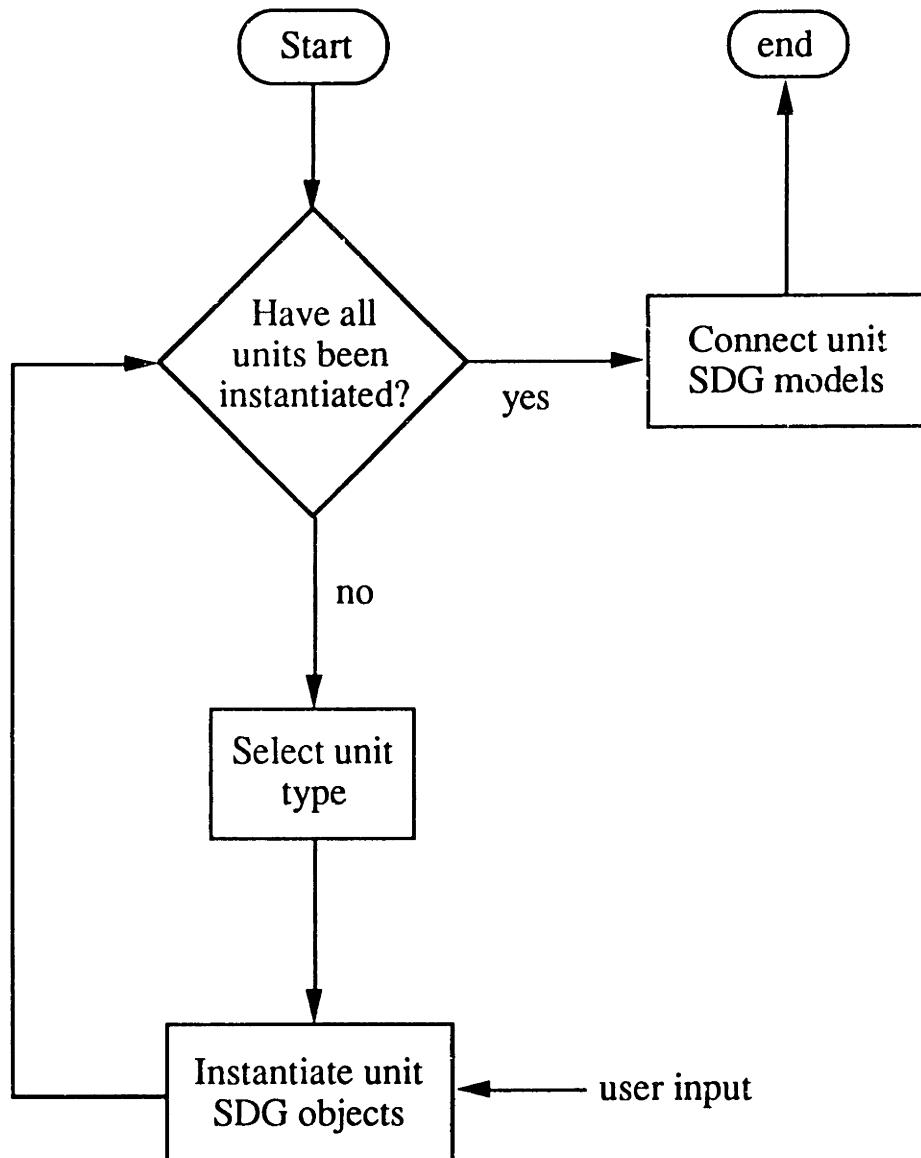


Fig. 6.4. Flowchart of Process SDG Creation Algorithm

- The identification numbers of adjacent units and their topological relationship to the process unit in the flowsheet.
- Relative values of state variables (e.g. temperatures and concentrations) in adjacent units.

After the developer responds to all questions for the unit, instances SDG objects for the unit are created as shown in Fig. 6.5.

Frame instances and their slot values for SDG objects of **pipe-1** of Fig. 6.2. are listed in Appendix G.<sup>12</sup> One notable feature are temporary causal arcs, **sdg-arc-100001** and **sdg-arc-100002** initiating from boundary variables in **pipe-1**'s adjacent units, **feed** and **tank-1**, and terminating at equivalent boundary variables in **pipe-1**.

The second and final step in building the process SDG is to connect and link the unit SDG models according to the flowsheet. The flowsheet structure is declared while responding to questions for the process units. Temporary causal arcs and boundary variables provide the "connecting link" between adjacent units. Connecting the unit models is accomplished by eliminating redundant variables from the pairs of boundary variables and temporary causal arcs. Other causal arcs are then reconnected across the eliminated variables and temporary arcs.

In summary, creating the process SDG requires specifying the flowsheet structure, the signs of relationships such as relative temperatures in adjacent units, and qualitative attributes of phenomena particular to process units.

---

<sup>12</sup>In this process, only flow-pressure interactions are modeled. Objects shown are those created just after instantiation of **pipe-1**.

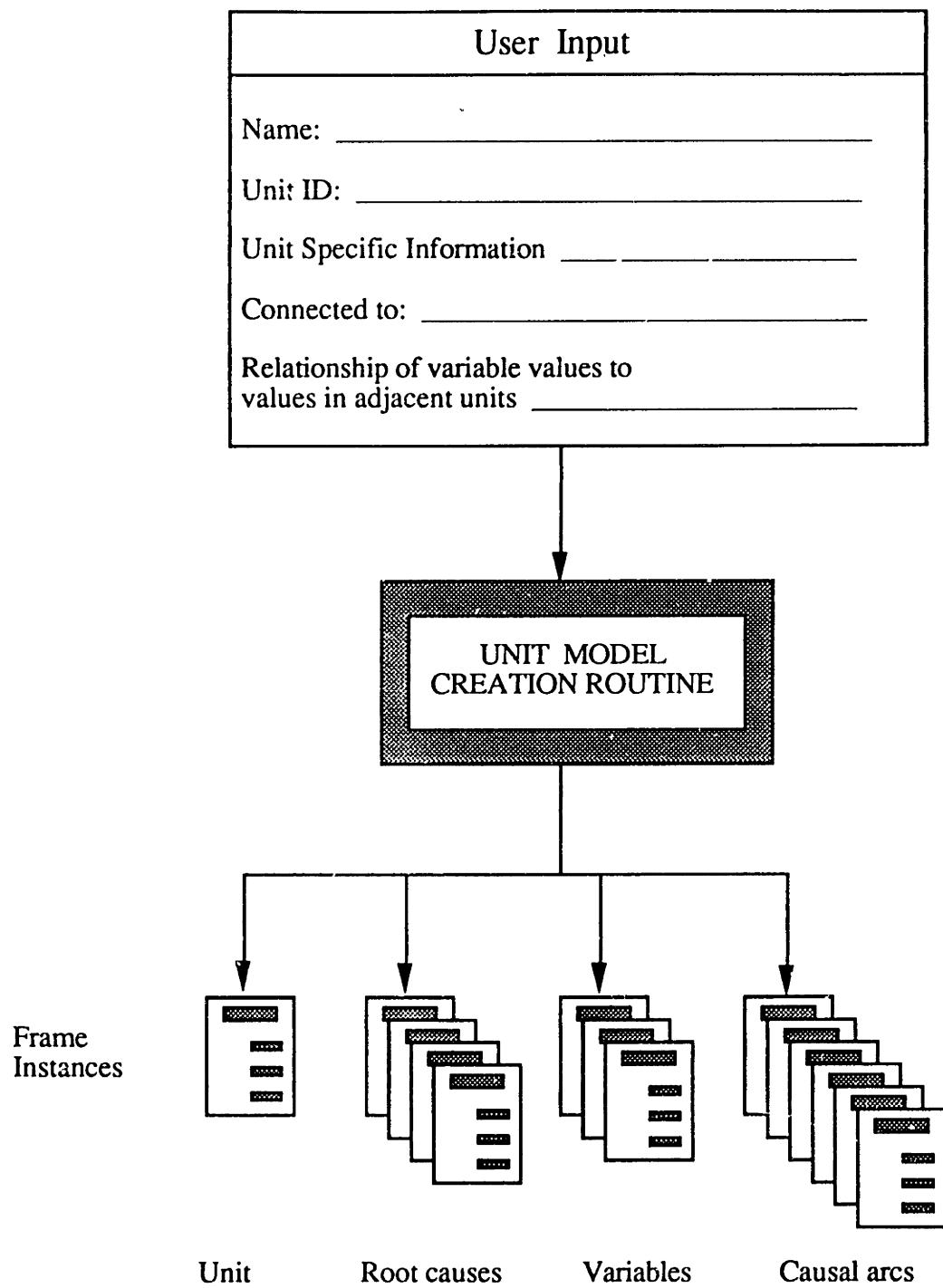


Fig. 6.5. Unit Model Creation Routines

## 6.5. Creating the Process ESDG

The second step in developing the PM is to create the process ESDG. An important limitation of the SDG is that certain non-monotonic behaviors resulting from global process interactions are not apparent from the local SDG models. This limits the use of the SDG as the basis for the PM. The ESDG accounts for these behaviors, and is used a basis for deriving the PM.

In this section, the algorithm used to create the process ESDG is described. Detailed instructions for using the algorithm are provided in Chapter 10. The algorithm is based on the global topological analysis of the SDG presented in Chapter 3. First, in Section 6.5.1, we review the theoretical basis for the topological analysis of the SDG. Additional attributes of ESDG objects are presented in Section 6.5.2. Direct application of the analysis is computationally intractable, and alternative criteria for deriving the ESDG are presented in Section 6.5.3. In Section 6.5.4, the basic algorithm is described. We conclude this section with a discussion about the complexity of the algorithm.

### 6.5.1. Theoretical Basis for Topological Analysis of the SDG

The Extended Signed Directed Graph (ESDG) is a causal model of the process representing both immediate **local** and apparent **global** causality. Derivation of global causality in the ESDG is based on a topological analysis of the SDG. The theoretical basis for the topological analysis was established in Section 3.3 of Chapter 3, and is summarized here.

To be useful in practical applications, a process diagnostic model should facilitate an accurate diagnosis without unnecessarily degrading diagnostic resolution. Furthermore, the process model should also provide explanations for transient process behavior. This requires that the basic models from which the diagnostic model is derived should:

- Potentially produce all actual behaviors for the system.
- Limit the number of spurious behaviors produced.

Process behavior is determined using the SDG, by allowing disturbances to propagate under a set of propagation assumptions. In order to limit the production of spurious behaviors, disturbance propagation may be constrained to acyclic paths in the SDG [4]. Restricting propagation to acyclic paths in the SDG results in the exclusion of certain non-monotonic behaviors due to global feedback interactions in the process. In Chapter 3, it was proven that although all initial system behavior is produced, the ultimate response is excluded when inverse and compensatory response occur in negative feedback loops.

The topological analysis of the SDG identifies all situations where inverse and compensatory response may arise due to "global" process feedback interactions. Non-physical arcs are added to the SDG to produce the ESDG, which account for these non-monotonic behaviors. Since the additional arcs account for the non-monotonic behavior only in situations where they arise, the ESDG produces almost<sup>13</sup> all actual behaviors without unnecessarily predicting spurious behaviors. Information about dynamic behavior incorporated in the ESDG, enables its use as a basis for deriving the PM. Next, the results of the topological analysis are presented.

#### 6.5.1.1. CRITERIA FOR LOCATING ESDG ARCS

In Chapter 3, theorems for identifying variables that exhibit inverse or compensatory response to negative feedback were derived. A qualitative matrix analysis was carried out, relating these aspects of system behavior to global properties of the system (**A**) and input (**B**) matrices. Using the definition of SDG arc signs and signs of self-cycles associated with variables in the SDG, the global properties of the matrices may be related to the topology of the SDG. This resulted in the expression of the theorems in terms of the global topology of the SDG. In the following, we present a summary of the analysis.

We begin by defining inverse and compensatory variables and terms relating to the topology of the SDG.

- An **inverse variable (IV)** is a variable that exhibits inverse response (IR) to a particular disturbance due to negative feedback effects.

---

<sup>13</sup>The ESDG may exclude transient measurement patterns in certain anomalous situations. These situations are described in Section 3.6.

- A **compensatory variable** (CV) is a variable that exhibits compensatory response (CR) to a particular disturbance due to negative feedback effects.
- A **path** from an initial to a terminal variable is a directed sequence of nodes and arcs in the SDG.
- An **acyclic path** is a path where all variables (including initial and terminal) appear only once.
- A **loop** is a path that has the same the initial and terminal variables, where each arc is traversed only once.
- A **cycle** is a loop in the SDG in which the initial and terminal variables are the same and all other variables in the loop are traversed only once.
- The **complementary subsystem** to an acyclic path in the SDG is the subgraph that is obtained if all variables in the acyclic path (including initial and terminal variables) are eliminated.
- The **complementary subsystem** to a cycle in a subgraph of the SDG is the subgraph obtained if all variables in the cycle are eliminated from the original subgraph.
- A **strongly connected component** (SCC) of the SDG, is a subgraph of the SDG for which a path exists from every variables  $u$  to every variable  $v$  (and from  $v$  to  $u$ ) in the SCC; and the SCC is not a proper subgraph of any other SCC.
- A **disturbance variable** to a SCC is an adjacent variable not located in the SCC from which at least one arc initiates to a variable in the SCC.

Consider an  $n^{\text{th}}$  order system with state variables and parameters,  $x$ , and input variables,  $u$ , described by a linearized set of ordinary differential equations:

$$\frac{d(dx)}{dt} = A \cdot (dx) + B \cdot (du) \quad (6.1)$$

Arc signs in the SDG are identical to signs of corresponding terms in the **A** and **B** matrices, and signs of self-cycles associated with variables are signs of diagonal elements of the **A** matrix. Malfunctions are represented by primary deviations in input variables, **u**.

Assuming no transitions across qualitative regimes,<sup>14</sup> a detailed analysis leads to an expression for the ultimate system response.

$$\begin{aligned}
 [dx_i] = & [ b_i(-1)^{n-1} |P_i| + \sum_{j \neq i} a_{ij} b_j (-1)^{n-2} |P_{ij}| \\
 & + \sum_{k \neq i,j} \sum_{j \neq i} a_{ik} a_{kj} b_j (-1)^{n-3} |P_{ijk}| + \dots + \\
 & + \sum_{p \neq \dots} \sum_{j \neq i} a_{ik} \dots a_{pj} b_j (-1) |P_{ijk\dots p}| \\
 & + \sum_{q \neq \dots} \sum_{j \neq i} a_{ik} \dots a_{pq} a_{qj} b_j |P_{ijk\dots pq}| ] \cdot [du]
 \end{aligned} \quad (6.2)$$

Here,  $(-1)^{n-2} |P_{ij}|$  is the signed principal minor (determinant) obtained by deleting elements in the *i*th and *j*th row and columns of **A**, etc. In eq. (6.2), it is sufficient to consider only perturbations from disturbance variables in causally adjacent upstream SCC's.

Defining the signed determinant of dimension *r* recursively in terms of signed determinants of its subsystems,

$$\begin{aligned}
 (-1)^n H = & -h_{ii} (-1)^{r-1} |Q_i| - \sum_{j \neq i} h_{ji} h_{ij} (-1)^{r-2} |Q_{ij}| \\
 & - \sum_{k \neq i,j} \sum_{j \neq i} h_{ji} h_{ik} h_{kj} (-1)^{r-3} |Q_{ijk}| \\
 & - \sum_{p \neq \dots} \sum_{j \neq i} h_{ji} h_{ik} \dots h_{pj} (-1) |Q_{ijk\dots p}| \\
 & - \sum_{q \neq \dots} \sum_{j \neq i} h_{ji} h_{ik} \dots h_{pq} h_{qj} |Q_{ijk\dots pq}|
 \end{aligned} \quad (6.3)$$

recursive necessary conditions for locating IVs and CVs can be derived.

First, in order to identify IVs and CVs, it is sufficient to consider the response of variables in SCCs with respect to perturbations in their respective disturbance variables.

<sup>14</sup>A qualitative regime is a region of process operation where all SDG arcs maintain the same sign. The analysis presented below was extended in Section 3.5 to account for non-monotonic behavior arising from transitions across qualitative regimes.

The necessary conditions for a variable in a SCC to display IR due to negative feedback to perturbations in a disturbance variable are:

- (1) The IV is located in a negative feedback loop or cycle (except self-cycles).
- (2) The complementary subsystem to one of the acyclic paths from the disturbance variable to the IV should contain a positive cycle (or self-cycle).
- (3) The complementary subsystem to the positive cycle in one of the complementary subsystems in (2) should violate either (5a, 5b or 6) below.

Similarly, the recursive necessary and sufficient conditions for a variable in a SCC to display CR due to negative feedback to perturbations in a disturbance variable are:

- (4) The CV is located in a negative feedback loop or cycle (except self-cycles).
- (5) The complementary subsystems to all acyclic paths from the disturbance variable to the CV should each
  - (a) have at least one zero self-cycle (integrator) and
  - (b) not have a cycle containing all variables in the subsystem.
- (6) The complementary subsystems to all non-zero cycles (excluding self-cycles) in each complementary subsystem in (5) should each satisfy (5a, 5b and 6).

The conditions derived for identifying IVs are necessary but not sufficient. Thus, the model developer needs to specify which of the potential IVs exhibit IR in the particular system. In certain situations,<sup>15</sup> stronger conditions for IVs, may be derived. These conditions indicate that certain subsets of the set of potential IVs identified by (1) - (3) above are inconsistent. This reduces the potential sets of IVs that need to be considered.

In addition to SDG arcs, the ESDG contains additional non-physical arcs, created to account for the non-monotonic behavior exhibited by IVs and CVs. The additional arcs are constructed so that disturbance propagation may be restricted to acyclic paths.

---

<sup>15</sup>These are described in Section 3.3.2.

Deriving IR arcs of the ESDG involves identifying IVs on each acyclic path from the disturbance variables based on the criteria presented above. On each of these acyclic paths, IVs occur as one or more strings of adjacent variables. For each string of IVs on an acyclic path, ESDG arcs originate at the node just upstream of the IVs and terminate at the non-inverse SCC variable located just downstream of the string. In effect, the ESDG arcs "jump over" the IVs providing an acyclic path accounting for IR. Similar rules determine the location of CR arcs. The arcs also jump over CVs providing acyclic paths accounting for CR. Additionally, conditions are assigned to the arcs to prevent the prediction of IR by the CVs.

### **6.5.2. Additional Attributes of ESDG Objects**

The analysis presented in the preceding section results in the:

- Identification of IVs and CVs with respect to disturbance variables.
- Creation of additional arcs in the ESDG.

This additional information is represented as attributes of arcs and variables in the ESDG. The list of additional attributes and their values for causal arcs and variables comprising the ESDG are also defined in Appendix F.

### **6.5.3. Alternative Criteria for Locating IVs and CVs**

The criteria presented in Section 6.5.1 for locating IVs and CVs, gives physical insight to the cause of IR and CR in negative feedback loops. Namely, the presence of positive feedback effects and integrators, respectively, in the loop. Determining IVs and CVs by direct application of the criteria, is a tree-recursive process as its involves recursive application of the relation in (6), above. Implementing tree recursive processes, result in algorithms that take an amount of time that grows exponentially with problem size [15]. In this section, recursive aspects of the criteria are recast into an equivalent form, for which known polynomial time algorithms exist.

The recursive criteria in (5) and (6) are the conditions for the qualitative value of the signed determinant of a subsystem to be zero-definite (eq. (6.3)).<sup>16</sup> Alternatively, the qualitative matrix of the subsystem is rank deficient. The problem of determining the qualitative rank of a matrix is identical to performing row and column interchanges so that a zero-free diagonal results. This problem is an instance of the well-known matching "marriage" problem in graph theory [16]. A rank deficient matrix leads to an incomplete matching (a non zero-free diagonal).

In Appendix H, we present the bipartite matching algorithm based on the algorithm of Hall (1956) [17]. Using this algorithm, the correct answer to the matching problem (and hence the qualitative value of the signed determinant) is obtained in  $O(|N| \cdot |N| + |A|)$  time. Here,  $|N|$  is the number of variables in the graph and  $|A|$ , the number of arcs. For sparse matrices this results in an  $O(N^2)$  algorithm. Using this algorithm results in a substantial increase in efficiency in locating IVs and CVs.

Thus, the necessary conditions for a variable in a SCC to display IR due to negative feedback to perturbations in a disturbance variable may be re-expressed as:

- (1) The IV is located in a negative feedback loop or cycle (except self-cycles).
- (2) The complementary subsystem to one of the acyclic paths from the disturbance variable to the IV should contain a positive cycle (or self-cycle).
- (3) The qualitative matrix of the complementary subsystem to the positive cycle in one of the complementary subsystems in (2) should be of maximum rank.

Similarly, the recursive necessary and sufficient conditions for a variable in a SCC to display CR due to negative feedback to perturbations in a disturbance variable may be expressed as:

- (4) The CV is located in a negative feedback loop or cycle (except self-cycles).

---

<sup>16</sup>An expression is zero-definite, if its value is zero independent of numerical values of variables appearing in the expression.

- (5) The qualitative matrices of the complementary subsystems to all acyclic paths from the disturbance variable to the CV should each be rank deficient. Note that for the qualitative matrix to be rank deficient, each complementary subsystem should
  - (a) have at least one zero self-cycle (integrator) and
  - (b) not have a cycle containing all variables in the subsystem.

#### 6.5.4. Algorithm for Constructing the ESDG

In this section, we present an algorithm for constructing the ESDG based on the criteria in the previous section. The algorithm is based on the direct application of criteria (1) through (5) of Section 6.5.3. The algorithm does not implement the extensions of the criteria to account for non-monotonic behaviors that occur as a result of transitions across qualitative regimes.

The criteria for IVs and CVs indicate that a process SDG may be decomposed into its SCCs before carrying out the analysis for IVs and CVs. Decomposing the SDG into its SCCs before determining if variables are IVs or CVs potentially improves the overall efficiency of the algorithm. In addition, cycles in the SDG have to be identified in order to evaluate if variables are IVs or CVs. We begin by stating a theorem relating SCCs to cycles in the SDG.

*Theorem*

If two cycles share at least one common variable then variables in both cycles must belong to the same SCC.

Proof

From the definition of a cycle, all variables in a cycle must belong to the same SCC. If two cycles share a common variable,  $w$ , a path can be traced from  $u$  in the first cycle to  $v$  in the other cycle passing through  $w$ , and similarly a path from  $v$  to  $u$ , also passing through  $w$ .

From this theorem, it follows that a SCC contains the largest set of cycles in which at least two members of the set share common variables. Therefore, it is possible to simultaneously find cycles and SCCs in the SDG.

A simplified flowchart describing the algorithm for creating the ESDG is shown in Fig. 6.6. Although not shown, actual implementation of the algorithm utilizes criteria that may eliminate sets of redundant IVs. The algorithm derives the ESDG in two stages.

- Finding cycles in the SDG and decomposing the SDG into its strongly connected components.
- Locating IVs and CVs in each SCC and creating associated ESDG arcs.

In decomposing the SDG into its SCCs, the relationship between cycles and SCCs is exploited. All cycles in the SDG are found, using a variant of the well-known depth first search algorithm [18] and then grouped into SCCs as described above. The algorithm used for finding cycles in the SDG is described in Appendix H.

Next, IVs and CVs are identified in each SCC, and then ESDG arcs are created accordingly. In order to locate IVs or CVs, further analysis is required only for SCCs,  $S$ , containing zero self-cycles or positive cycles. In essence, the algorithm is implemented as a series of nested loops in which qualitative determinants are evaluated for all combinations of:

- disturbance variables,  $D$ , to the SCC,
- acyclic paths,  $AP$ , within the SCC initiating from the disturbance variables, and
- variables,  $V$ , on the acyclic paths.

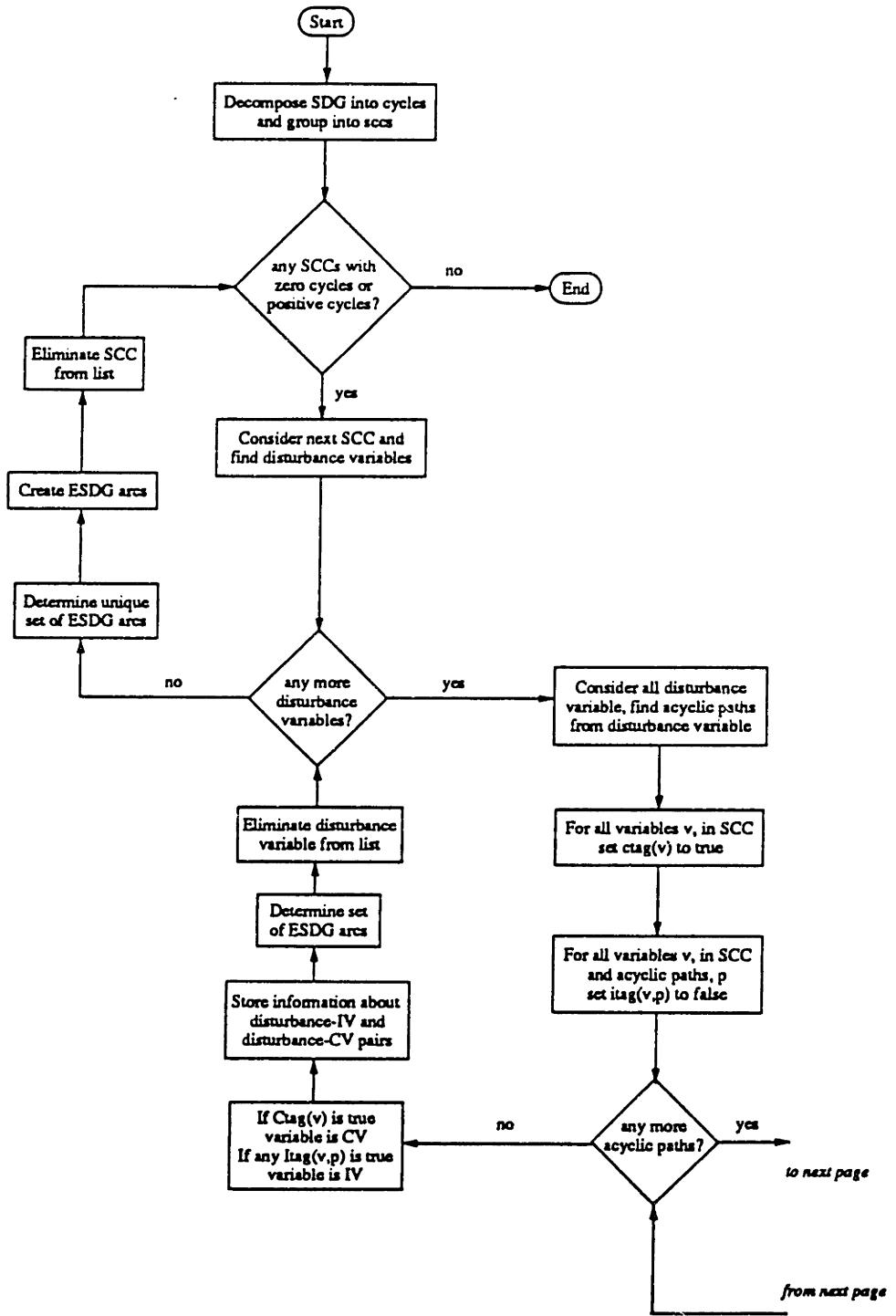


Fig. 6.6. Flowchart of Algorithm for Constructing ESDG

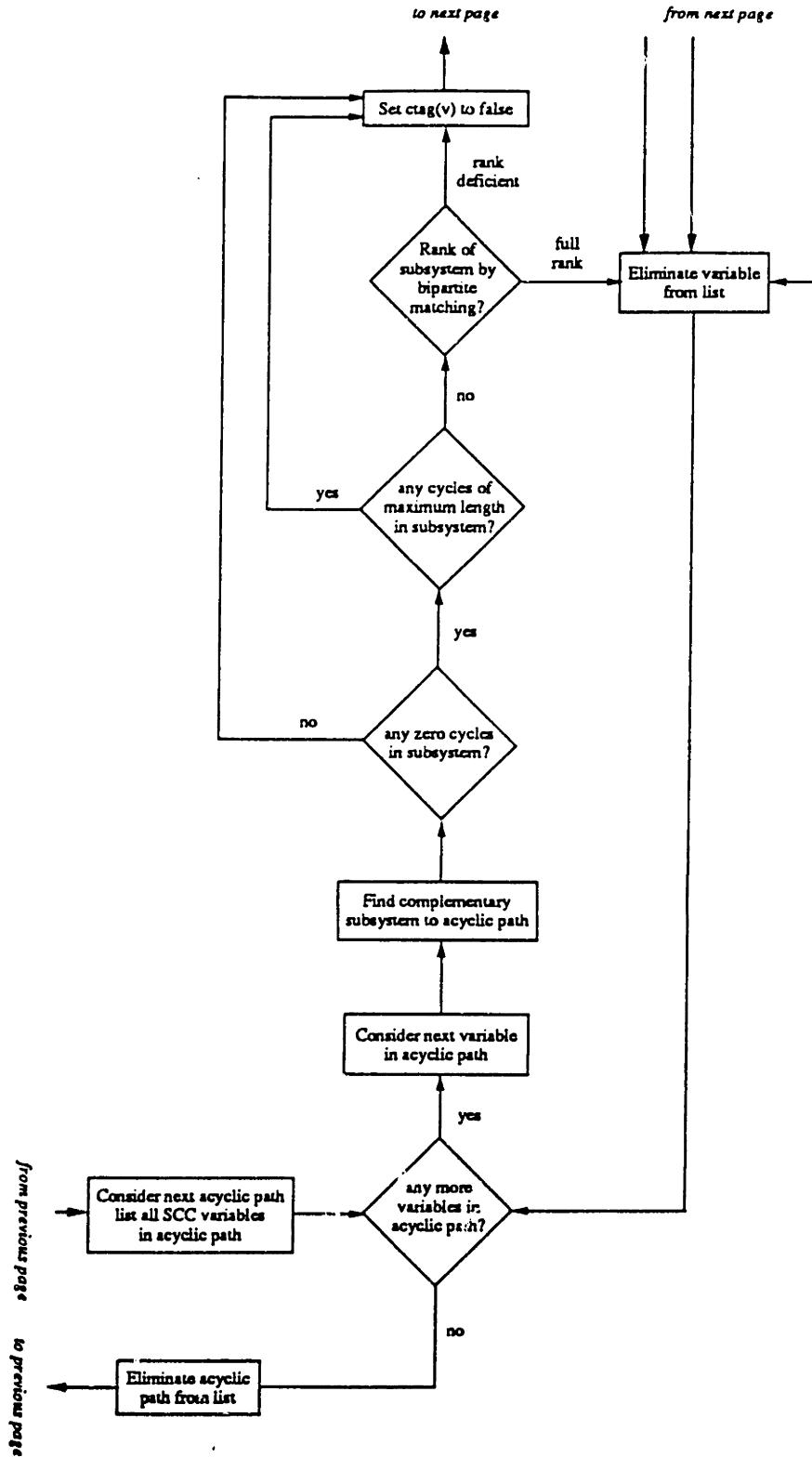


Fig. 6.6. Flowchart of Algorithm for Constructing ESDG (continued)

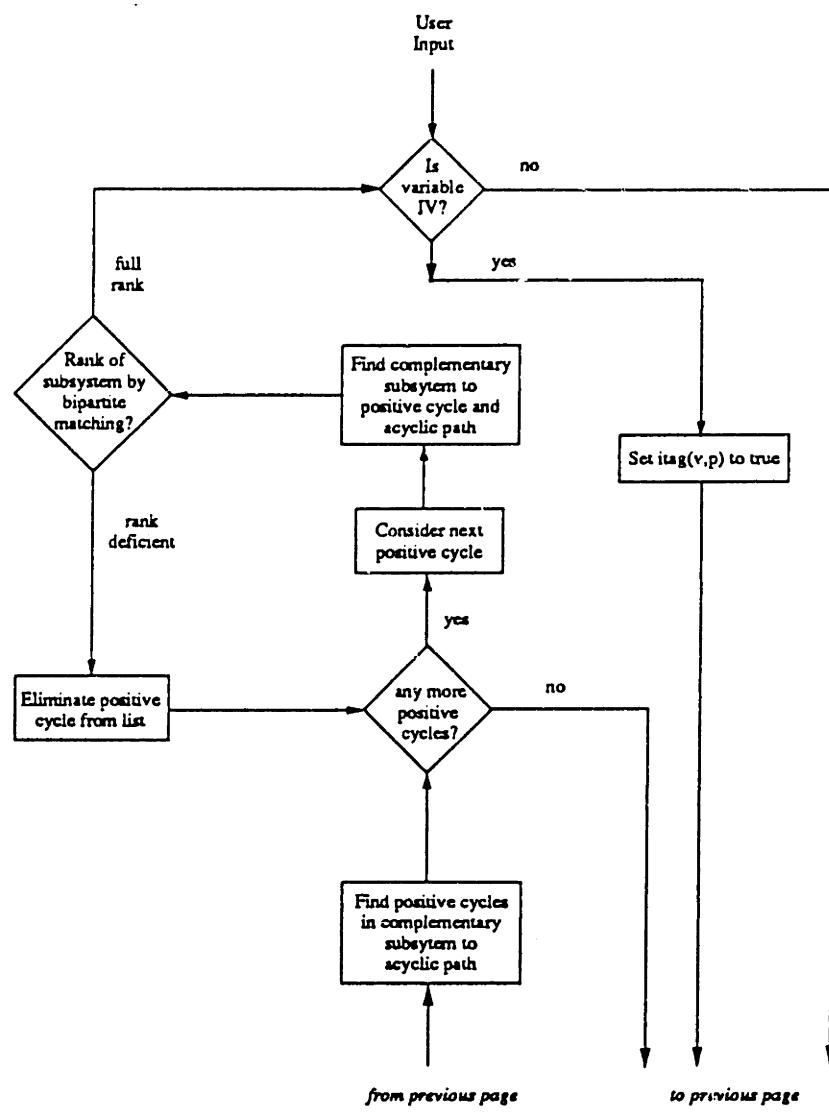


Fig. 6.6. Flowchart of Algorithm Constructing ESDG (continued)

For each of these combinations, the bipartite matching algorithm is used to evaluate the rank of the complementary submatrix to each acyclic path, AP, from disturbance variable, D, to variable, V, in the path. First, the algorithm (for creating the ESDG) determines if the variable is a CV. If the submatrix is rank deficient, V is a CV with respect to D. Before the bipartite matching algorithm is applied, the subsystem is checked to ensure that it has at least one zero self-cycle and no cycles of maximum length. If the variable is not a CV, and the complementary subsystem has a positive cycle, PC, the bipartite matching algorithm is applied to the complementary subsystems to the acyclic path and positive cycles to determine if V is a potential IV. Interactive input from the model developer is required to confirm if the variable is an IV.

After analysis of all acyclic paths from the disturbance variable, information about disturbance-inverse variable pairs and disturbance-compensatory variable pairs is stored in the appropriate slots of (E)SDG variables. Instances of ESDG arcs are created for the SCC after the analysis has been completed for all disturbance variables in the SCC.

### 6.5.5. Complexity of the Algorithm for Constructing the ESDG

In this section, the complexity of the algorithm described in the previous section is discussed. The strategy adopted to reduce the time required to construct the ESDG of large scale processes is also described. Other strategies are discussed in Section 6.8.

In order to estimate the time required to construct the ESDG, the algorithm may be decomposed into its basic steps. The time taken to locate IVs and CVs in the SCCs is usually greater than that required to find cycles and decompose the SDG into its SCCs.<sup>17</sup> Thus, the total time taken to construct the ESDG may be approximated by the time it takes to locate IVs and CVs.

The algorithm locates all IVs and CVs by evaluating qualitative determinants for all combinations of disturbance variables, acyclic paths and variables on acyclic paths of the SCC. Therefore, IVs and CVs in a SCC can be located in  $O(|SI| \cdot |DI| \cdot |API| \cdot |LI| \cdot (1 + PC) \cdot |SV|^2)$  time.  $|LI|$ , and  $|SV|$  are the maximum number of variables in acyclic paths in the

---

<sup>17</sup>In our experience, it takes about 10 times longer to locate IVs and CVs.

SCC and in the SCC, respectively. Large processes also tend to have a limit on the size and connectivity of the SCC. Therefore, it is expected that after decomposition into its SCCs, the time required to construct the ESDG for most processes should increase linearly with problem size.

The algorithm has been successfully applied to several processes. One of the larger examples tried was a continuous stirred tank reactor process with its associated heat exchange and control systems. The process consists of:

- 32 unit models
- 241 variables
- 304 arcs

A flowsheet of the process is shown in Fig. 6.7. Other features of the process are shown in Table 6.7.

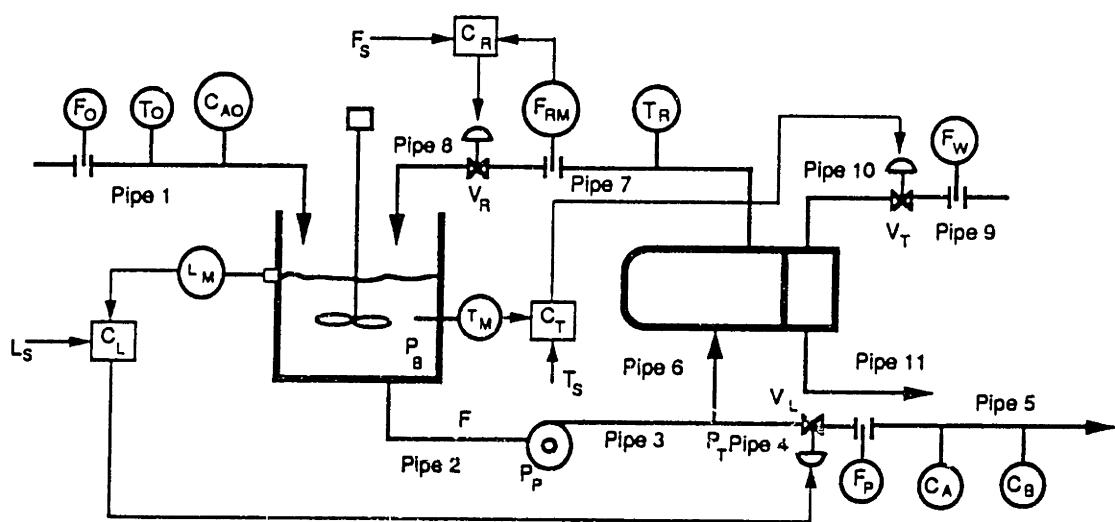


Fig. 6.7. Continuous Stirred Tank Reactor with Recycle

**Table 6.7 Time Taken by Algorithm to Create ESDG  
of Reactor Process**

SCC-NO	No of dist vars	No of vars in SCC	Av. no of APs per dist var	Av length of AP	Time
Decomposition to SCCs	---	---	---	---	1 hr. 6 mns
1	28	54	7.93	17.64	7 hrs. 45 mns
2	33	41	10.85	19.26	4 hrs. 51 mns
3	3	6	1.67	4.00	5 secs
Total	---	---	---	---	13 hrs. 42 mns

Using the basic algorithm, 120 additional ESDG arcs were created for this process, using an IBM 386 class PC running at 20 MHz with 10MB RAM in 13 hours 42 minutes. The time taken for each step is shown in Table 6.7.

#### **6.5.5.1. REDUCING THE TIME REQUIRED TO CREATE ESDG**

In this section, we discuss the approach taken in reducing the time taken to create the ESDG. The strategy adopted bears in mind that variables not representing process measurements in the ESDG will be eliminated to obtain the PM. Implementing this strategy reduced the time taken to construct the ESDG of the example process in the previous section by a factor of five. Comparable savings are expected for other processes.

In the previous section, it was shown that the time required to construct the ESDG of a process depends on both the size and the connectivity of the SDG. In particular, the

complexity in locating IVs and CVs is related to the number of variables located in its largest SCC and the number of disturbance variables to the SCC. The basic strategy adopted is to condense the SDG before subsequent analysis.

The SDG may be condensed by:

- Eliminating root causes that are deemed unimportant for diagnosis and all unmeasured variables in the SDG that are unaffected by other root causes.
- Condensing variable structures by removing variables to obtain equivalent structures with less variables.

Eliminating variables that are affected only by unimportant root causes reduces the number of disturbance variables to SCCs at the boundary of the process. Condensing variable structures primarily reduces the number of cycles in the SDG, number of variables in SCCs and the number and length of acyclic paths in SCCs.

In condensing variable structures, one objective is to preserve the signs of acyclic paths between process measurements in the SDG. The other is to preserve global qualitative effects accounted for by non-physical ESDG paths. This is accomplished by ensuring that:

- No process measurements are within the structure.
- The number of arcs entering and leaving the structure remain unchanged after removing the variables.
- The signs of all paths starting from arcs entering the structure and ending at arcs leaving the structure remains the same.
- The qualitative signed determinant of the structure is unchanged.

Before further analysis, the structures are identified and condensed by the algorithm. A heuristic approach is used in selecting structures that are to be condensed. Commonly occurring structures in the SDG of chemical process that are condensed by the improved algorithm are:

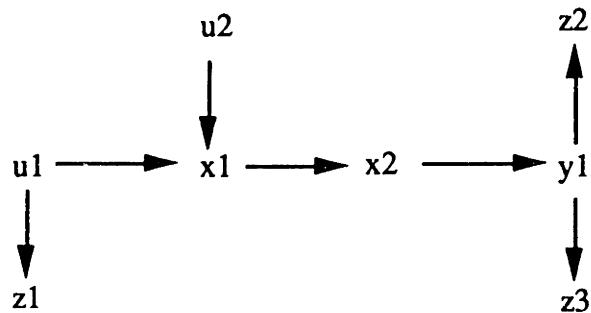


Fig. 6.8a. Linear Path in SDG

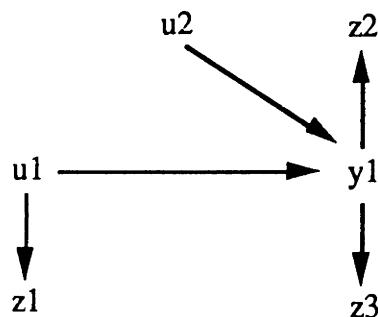


Fig. 6.8b. Linear Path After Condensation

**Linear paths:** These are variables in a path with negative self-cycles and one initiating arc. These paths often arise from modeling convective transport processes and process control loops. All internal variables in the path are eliminated as shown in Figs. 6.8a and 6.8b. The qualitative signed determinant of linear paths remains positive.

**Flow-pressure cycles:** Modeling flow pressure phenomena in incompressible flow results in cycles between flow and pressure nodes. These cycles may be in linear sequences of process equipment, recycle or bypass loops. Provided no arcs initiate from these cycles to other variables in the process, internal cycles in these structures may be eliminated as shown in Figs. 6.9, 6.10 and 6.11. In Appendix I, we show that the qualitative signed determinant of these structures is still zero after removing the cycles.

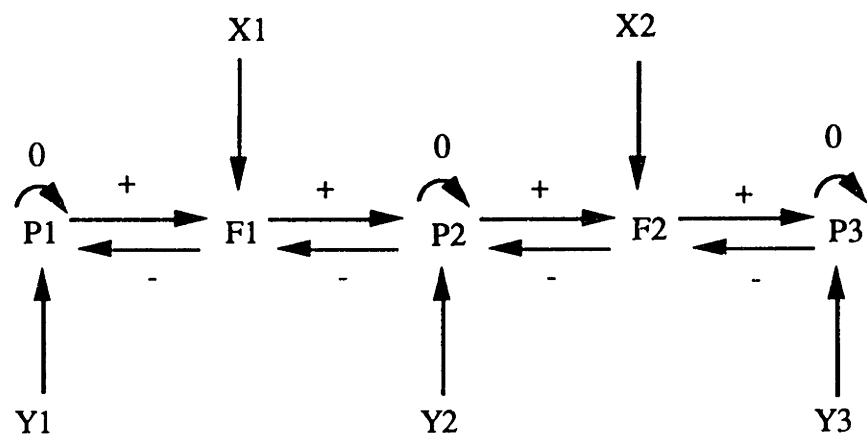


Fig. 6.9a. Flow-Pressure Cycles in Linear Configuration

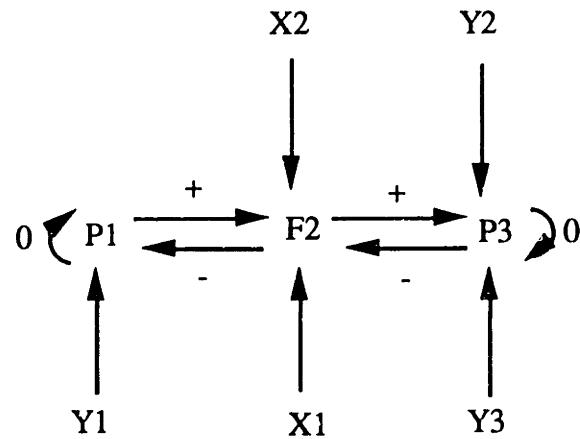


Fig. 6.9b. Linear Flow-Pressure Cycles After Condensation

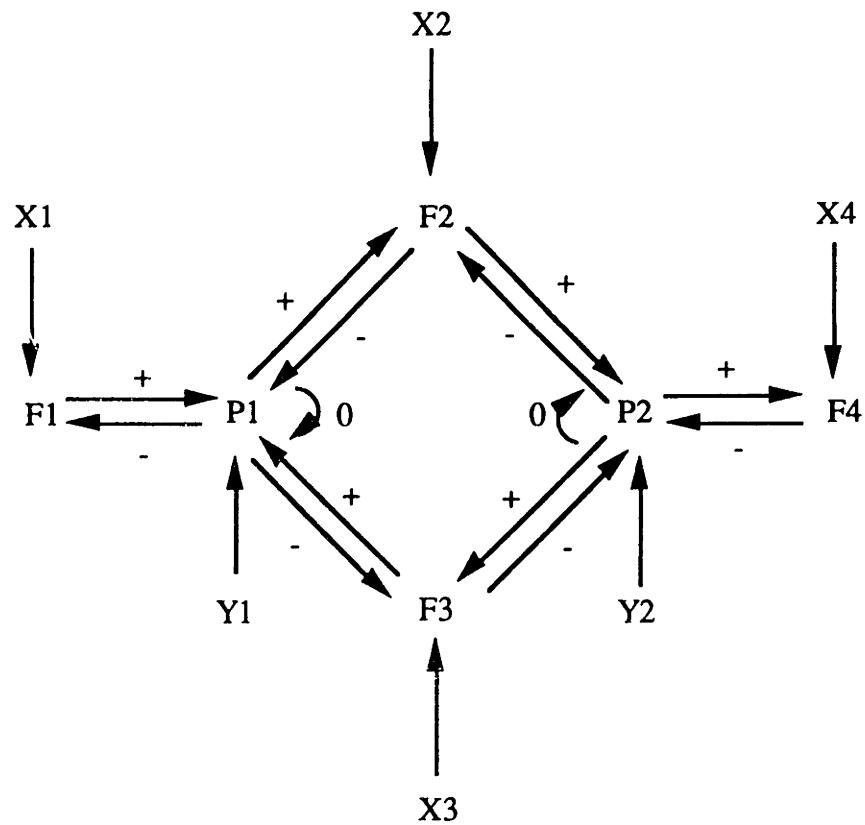


Fig. 6.10a. Flow-Pressure Cycles in Recycle Configuration

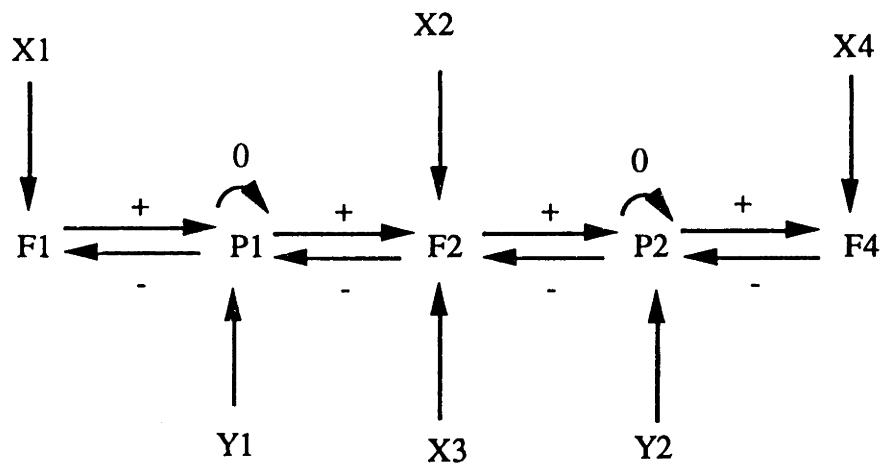


Fig. 6.10b. Recycle Flow-Pressure Cycles After Condensation

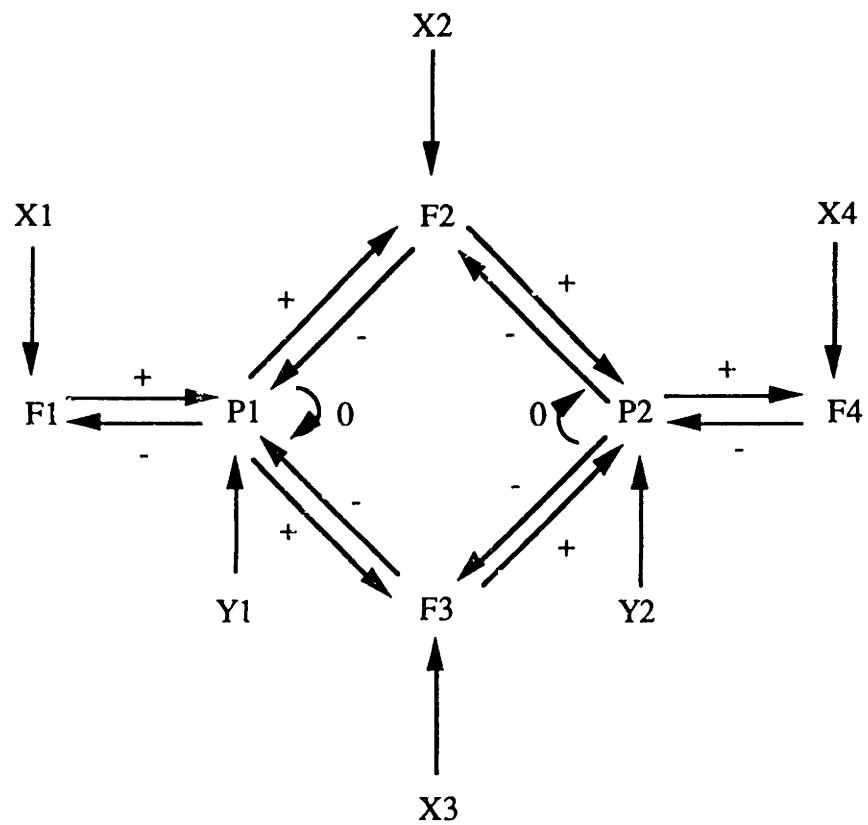


Fig. 6.11a. Flow-Pressure Cycles in Bypass Configuration

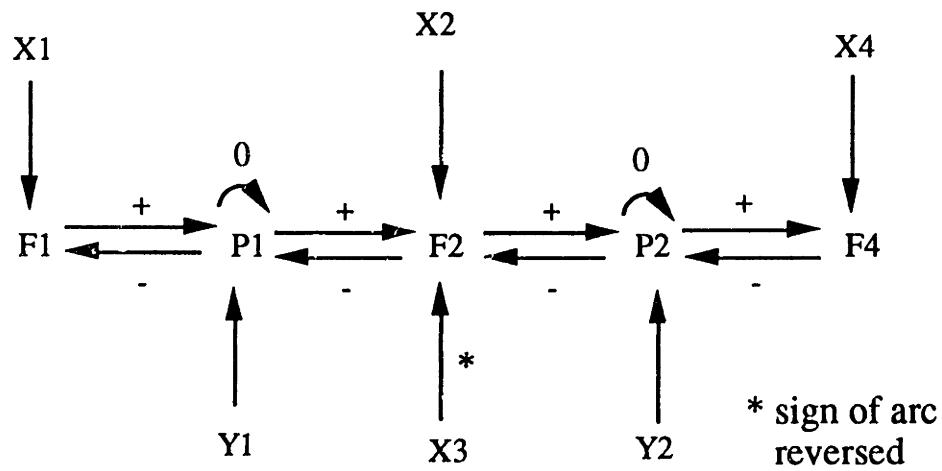


Fig. 6.11b. Bypass Flow-Pressure Cycles After Condensation

83 variables and 97 arcs were eliminated from the above structures in the SDG of the reactor process. 121 ESDG arcs were created in 2 hours 33 mins (using the same computer) from the condensed SDG, representing an overall reduction by a factor of 5. Details of the time taken by the algorithm in each step is shown in Table 6.8 for comparison.

In general, other structures may be identified and the SDG may be condensed further. Condensing the SDG is very useful in staged processes where structures may be repeated many times in the SDG. Other strategies that may be employed to reduce the time taken to create the ESDG are discussed in Section 6.8.

Table 6.8 Time Taken to Create ESDG of Reactor Process  
After Condensation

SCC-NO	No of dist vars	No of vars in SCC	Av. no of APs per dist var	Av length of AP	Time
Eliminating variables	---	---	---	---	1 mn
Decomposition to SCCs	---	---	---	---	18 mns
1	29	27	7.14	11.05	1 hr. 16 mns
2	33	26	8.12	11.99	58 mns
3	3	6	1.67	4.00	5 secs
Total	---	---	---	---	2 hr. 33 mns

## 6.6. Deriving Qualitative Relationships in the PM

The third step in developing the PM is to derive the qualitative relationships in the PM. In this section, we describe an algorithm used to translate the ESDG into qualitative relationships in the PM. Detailed instructions for using the algorithm are provided in Chapter 10. In Section 6.6.1, we describe the basic tasks performed by the algorithm. The algorithm is described in further detail in the next section. In Section 6.6.3, typical relationships represented in the PM are depicted using the feed preheat subsection of a process as an example. Finally in Section 6.6.4, we conclude with a discussion about the complexity of the algorithm.

### 6.6.1. Basic Tasks Performed by the Algorithm

In this section, essential tasks performed by the algorithm that translates the ESDG to the PM are described. In essence, the ESDG describes potential transitions that may occur as a result of process disturbances or malfunctions. On the other hand, the PM expresses causal relationships from which diagnostic conclusions can be inferred from observing sequences of transitions. Thus, relationships depicted in the ESDG must be **inverted** in order to derive the PM.

Qualitative relationships in the PM are expressed as directional links among root causes that potentially affect the process, and transitions in the state and status of sensor signals.<sup>18</sup> Status transitions are directly related to failures of associated sensors to fixed values. For example, the links between **f1-sensor-failed-high** and F1: (F) (failed) and between F1: (F) and F1: (OK) in the PM of the feed preheater process (Fig. 6.16) express the status transitions associated with failures in **f1-sensor**. Deriving relationships between sensor failures and status transitions is straightforward and therefore the focus is on deriving relationships associated with state transitions.

---

<sup>18</sup>Subsequently, we shall refer to variables depicting process observations in the ESDG as sensor signals.

Links in the PM can be classified as **local cause links** or **compiled causal links**. Local cause links initiate from a potential root cause and terminate at a sensor signal.<sup>19</sup> If invoked during diagnosis, potential root causes associated with local cause links are added to the hypothesis set. Compiled causal links initiate and terminate at sensor signals. These can be further distinguished according to the type of actions<sup>20</sup> that are carried out if the link is invoked during diagnosis. The various types of compiled causal links include<sup>21</sup>

**:not links:** Eliminate the set of potential root causes associated with the link from the current hypothesis set.

**:only-if-exists links:** Retain the set of potential root causes associated with the link and eliminate all other potential root causes from the current hypothesis set.

**:only-if-transient links:** Initiate from an abnormal state to the normal state of the same sensor signal. These links indicate a process condition resulting from a transient disturbance or malfunction.

Other types of compiled causal links that may be specified are **:add links**, **:retain links** and **:replace links**.

In order to create these links, the algorithm:

- Searches along acyclic paths in the ESDG, removing variables other than sensor signals.
- Partitions parallel acyclic paths initiating from the same root cause or sensor signal into groups according to their time delays.

---

<sup>19</sup>Local cause links are implemented as attributes of potential root causes and measured variables.

<sup>20</sup>Actions are represented by pairs in the not-conditions and only-if-conditions slots of complied causal links. The first member of the pair is a keyword indicating the type of action and the second, the list of root causes affected by the action.

<sup>21</sup>:not links, :only-if-exists links and :only-if-transient links are created by the algorithm. Currently, other types of links are not created.

- Coalesces paths within a group, and determines allowable state transitions expressed by links in the PM.

To create these links knowledge about the following is required:

- Which of the variables are sensor signals.
- Disturbance variable/inverse (compensatory) variable pairs.
- Signs, time delays, delta variables, necessary disturbance variables and disabling conditions associated with ESDG arcs.

### **6.6.2. Algorithm for Creating the PM**

The taxonomy introduced in the previous section for expressing qualitative relationships does not result in a unique specification of the PM. Thus, different PMs may be derived from the same ESDG may produce an identical diagnosis when used with the inference algorithms in MIDAS. Several criteria may be used in selecting among several PMs describing a process.

- Most importantly, the PM should retain the information contained in the ESDG, facilitating an accurate diagnosis without unnecessary reducing resolution.
- In order to improve efficiency during diagnosis, the PM should be compact, containing as few links as possible. Actions performed by the links should also involve a "minimum" number of potential root causes.
- Transitions associated with links in the PM, should as much as possible reflect the original causal ordering in the ESDG. The ordering may be altered to improve robustness of the diagnosis to out of order events.

Generally, it may not be possible to satisfy all these objectives simultaneously. Where conflicts arise, trade-offs are sought, provided the trade-offs do not result in an adverse reduction of the information content of the PM.

The algorithm used to create the PM for MIDAS attempts to balance these objectives. In the remainder of this section, the algorithm is presented in detail. We commence by introducing terms used in partitioning acyclic paths in the ESDG.

In general, conditions<sup>22</sup> associated with ESDG arcs may result in some acyclic paths being incapable of propagating disturbances. These paths are known as invalid paths. An acyclic path, p, denoted by an ordered sequence of nodes, n,<sup>23</sup> and arcs, a,

$$p = n_1 \ a_{12} \ n_2 \ a_{23} \ n_3 \dots a_{m-1,m} \ n_m \quad (6.4)$$

is **valid**, only if the following two conditions are satisfied by all arcs in the path. First, none of the all-delta-variables of the arc is "upstream" of the arc in the path:

$$\forall a_{jk} \ n_i \notin \text{all-delta-variables}(a_{jk}), i < k \quad (6.5)$$

Second, all the necessary-disturbance-variables of the arc are "upstream" of the arc in the path:

$$\forall a_{jk} \ n_i \in \text{necessary-disturbance-variables}(a_{jk}), i < k \quad (6.6)^{24}$$

The all-delta-variables and necessary-disturbance-variables slots of arcs in the ESDG are defined in Table F.4 in the Appendix F. The set of **deviation paths**, DP, from a node, n, to a set of sensor signals, S, is the set of all valid paths which start at n and end at one of the nodes in S:

$$DP = \{dp: dp \text{ is valid} \wedge \text{start}(dp) = n \wedge \text{end}(dp) \in S\} \quad (6.7)$$

For a given "cost" function, C(dp), **minimum cost deviation paths**, MC, from node n are the subset of deviation paths with the minimum cost. Formally,

$$MC = \{mc: C(mc) = \min_{dp \in DP} C(dp)\} \quad (6.8)$$

<sup>22</sup>These conditions are represented as attributes of ESDG arcs in Table F.4.

<sup>23</sup>A node may be a variable or a root cause.

<sup>24</sup>This condition applies if the acyclic path traverses more than one SCC.

The "cost" function of interest in establishing links in the PM is the time delay of the path. An arc with a delay of 0 indicates instantaneous transmission of effects by the arc, while a delay of 1, indicates that transmission takes a finite amount of time. Since the relative numerical delay of arcs with delay 1 is unknown, it is not always possible to order paths according to the magnitudes of their delays. However, the time delay of path  $p_1$ ,  $TD(p_1)$  is greater than the delay of  $p_2$  if for all arcs in  $p_2$  not shared with  $p_1$ , the time delay is 0, and there is at least one arc of  $p_1$  not shared with  $p_2$  that has a time delay of 1:

$$\forall a_2: a_2 \in p_2 \wedge a_2 \notin p_1, TD(a_2) = 0; \wedge \exists a_1: a_1 \in p_1 \wedge a_1 \notin p_2, TD(a_1) = 1 \quad (6.9)$$

Therefore, **minimum delay deviation paths**, MD, from  $n$  are the subset of deviation paths whose time delays are not greater than the time delay of any other deviation path:

$$MD = \{md: TD(md) \rightarrow TD(dp), \forall dp \in DP\} \quad (6.10)$$

When all minimum delay deviation paths have at least one disabling-condition in common, the set of minimum delay deviation paths is expanded to include activatable<sup>25</sup> minimum delay deviation paths. The root causes which disable all minimum delay deviation paths, DRC, is represented by the set of disabling conditions that are common to all minimum delay deviations paths:

$$DRC = \{drc: drc \in \text{disabling-conditions}(md), \forall md \in MD\} \quad (6.11)$$

Given that the set DRC is not empty, **activatable minimum delay deviation paths**, AMD, are paths with minimum delays among the remaining deviation paths that are not disabled by any member of DRC.

$$\begin{aligned} AMD = \{amd: & \text{disabling-conditions}(amd) \notin DRC \\ & \wedge TD(amd) \rightarrow TD(dp), dp \notin MD \wedge dp \in DP\} \end{aligned} \quad (6.12)$$

Any of the root causes, DRC, disable the minimum delay deviation paths, and allow the activatable minimum delay paths to become minimum delay paths.

---

<sup>25</sup>There is a distinction between activatable minimum delay deviation paths and activatable compiled causal links introduced in Section 6.3.

After the occurrence of a root cause, the "most" useful sources of diagnostic information are the first few (observed) events that may be causally related to the root cause. Subsequent events do not change the relative ranking between the root cause and other competing root causes in the hypothesis. In MIDAS, this effect may be achieved by limiting the "length" of deviation paths when deriving the PM. Limiting the length of deviation paths may break real causal pathways, thus, potentially reducing the information content of the PM. However, an appropriate choice for the maximum length of the paths does not degrade the quality of the diagnosis. Limiting the length of deviation paths results in more compact PMs without degrading diagnostic performance.

The length of a deviation path is defined in terms of the number of strongly connected components containing sensor signals over which the path traverses. Since interactions among variables in a SCC are coupled, deviation paths must include all variables in the same SCC. Furthermore a deviation path must contain at least two signal variables, as the application of :not or :only-if-exists links between events associated with the signal variables may affect the relative ranking between the root cause and other root causes. Since both signal variables may not be in the same SCC, the minimum acceptable value for the length of deviation paths is 2 SCCs (containing signal variables). Because events do not always occur in their ideal sequence,<sup>26</sup> a conservative choice for the length of deviation paths is  $2 + M$ , where  $M$  is the maximum number of missing intermediate events in an ideal sequence of events.

A simplified flowchart of the algorithm for creating the qualitative core of the PM is shown in Fig. 6.12. The dashed lines in the figure refer to information flow between various stages of the algorithm. The first step in deriving the PM is to determine entries in a matrix of allowable state transitions of sensor signals for each potential root cause. Each entry in this matrix are the possible states a sensor signal may attain as a result of a root cause. For example, the entries in Table 7.2 are the allowable states that may be attained by sensor signals in a jacketed reactor process as a result of process malfunctions. Entries in the matrix derive from the signs of deviation paths from root causes in the ESDG. The matrix is used in later steps to derive potential root causes associated with :not links in the PM.

---

<sup>26</sup>The ideal order of an event sequence is the order of events indicated by the 0-1 time delays.

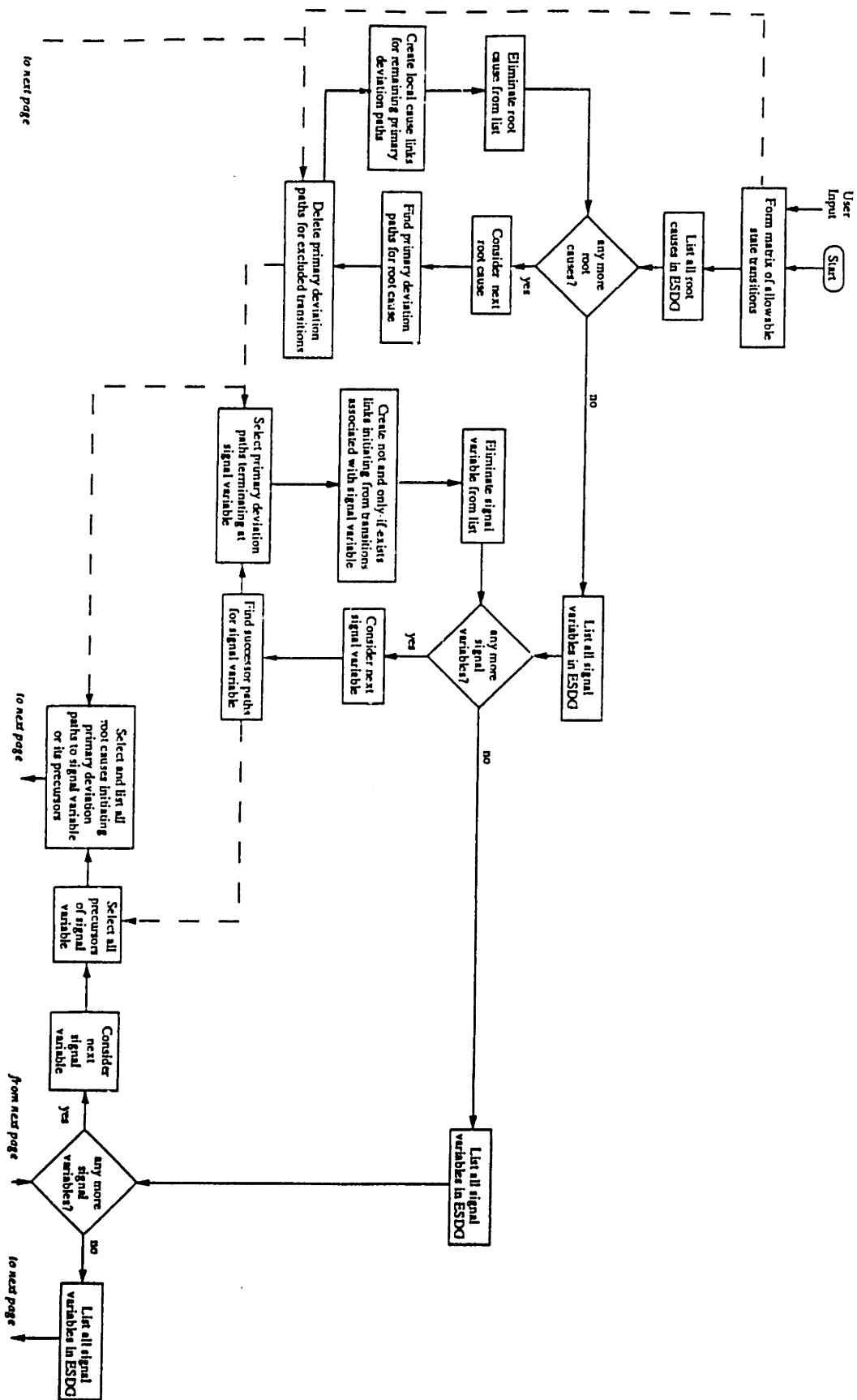


Fig. 6.12. Flowchart of Algorithm for Creating PM

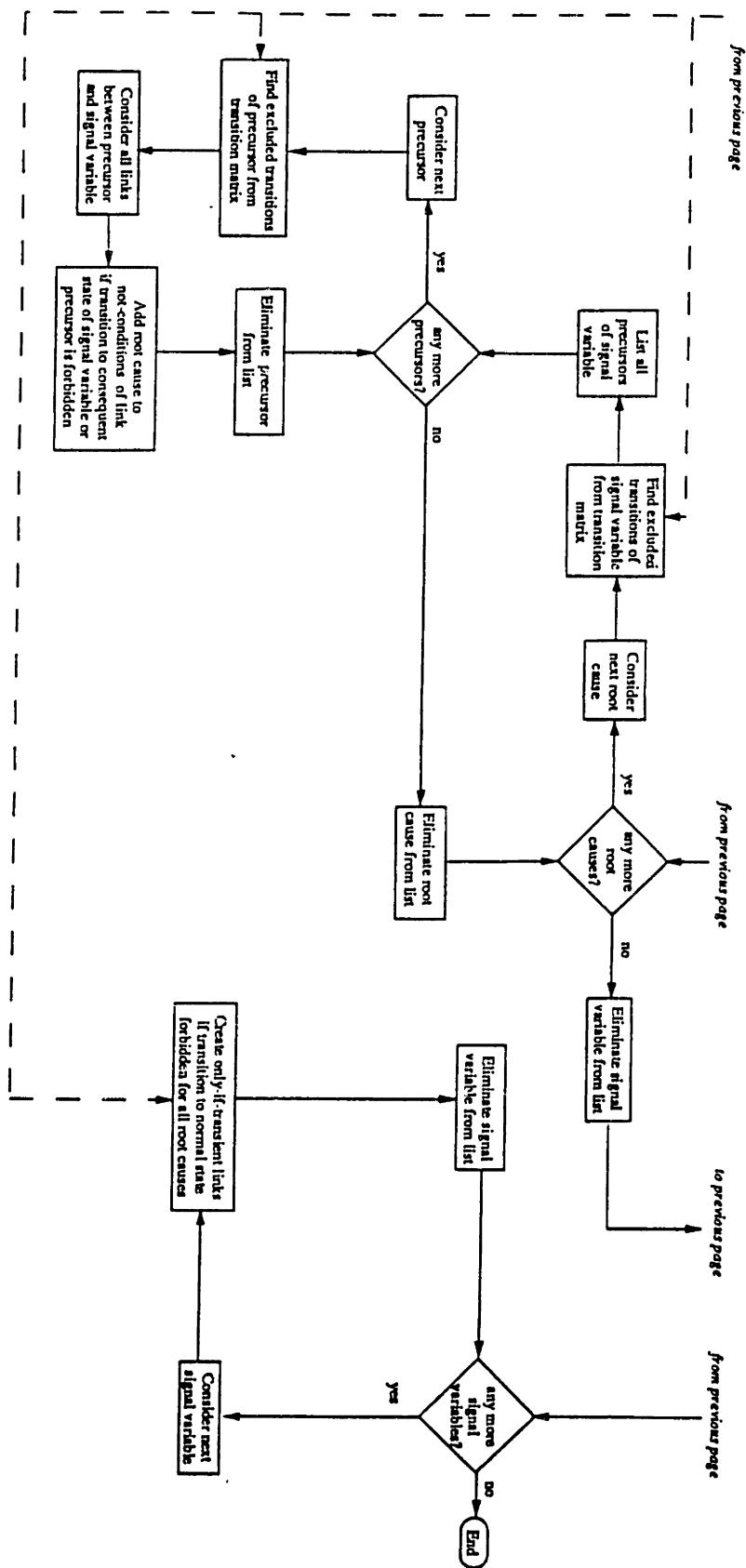


Fig. 6.12. Flowchart of Algorithm for Creating PM (continued)

During this stage, **process specific information** may be supplied by the developer of the PM to resolve qualitative ambiguities. These ambiguities arise when **deviation paths** of opposite signs exist from a root cause to the same sensor signal.

Next, instances of local cause links are created for each potential root cause. Local cause links initiate from potential root causes and terminate at transitions associated with events that are observed first as a result of the root cause. These links are derived from minimum delay deviation paths from the root cause. The result state of transitions associated with these paths are non-normal states that derive from the signs of the respective paths. Excluding links associated with other deviation paths from the potential root cause preserves the original causal ordering in the ESDG and maintains a compact PM. The algorithm for locating minimum deviation paths from root causes<sup>27</sup> is described in Appendix J.

Ambiguities arise when minimum deviation paths with opposite sign initiate from a root cause to the same sensor signal. If an ambiguity was resolved in the previous step, paths that lead to excluded transitions are deleted. Local cause links are created from the remaining primary deviation paths.

The third step is to create instances of **not** and **only-if-exists** compiled causal links associated with transitions of measured variable events. These links express direct successor relationships between the events. Because immediate successor relationships and no others are expressed, causal ordering is maintained in the PM. This also results in a compact PM. **:not** links and **:only-if-exists** links are derived from minimum delay deviation paths initiating from sensed variables associated with sensor signals.<sup>28</sup> Finding successor paths associated with a sensor signal can be decomposed into two steps.

First, starting from the sensor signal a backwards search is used to find **sensed variables** of the sensor signal. Paths between sensed variables and sensor signal are denoted as equivalent paths. The relationship between sensed variables and sensor signals is depicted using the ESDG of the hypothetical process in Fig. 6.13. The time delays on each arc are shown. Simply<sup>29</sup>, when there are other deviation paths initiating from the

---

<sup>27</sup>These deviation paths are known as primary deviation paths.

<sup>28</sup>These deviation paths, known as successor paths initiate from sensed variables of the sensor signal.

<sup>29</sup>In general, sensed variables are the set of furthermost upstream variables on paths with 0 delays from the sensor signal.

immediate precursor of the sensor signal to other sensor signals, the sensed variable is the sensor signal. For example,  $B_s$  and  $C_s$ . Otherwise, in when other deviation paths do not exist, the sensed variable is the variable at which the search from the precursor or root cause was terminated along non-minimum delay deviation paths. For example, A and D.

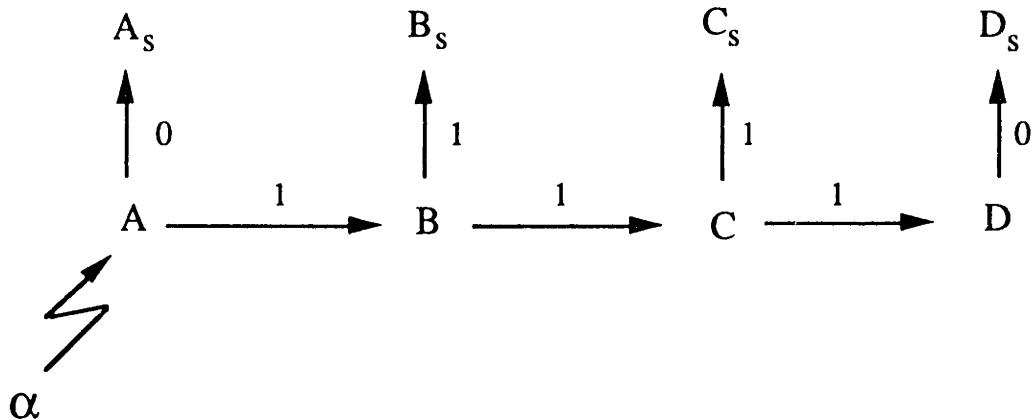


Fig. 6.13. Relationship Between Sensed Variables and Sensor Signals

After locating the sensed variables of a sensor signal, successor paths initiating from each sensed variable are found. The algorithm for finding the successor paths initiating from a sensed variable is described in Appendix J. Because successor paths initiate from sensed variables rather than from sensor signals, effects attributed to all acyclic paths in the ESDG are represented in the PM.

:not links derive from non-activatable successor paths. Transitions associated with result nodes attributed to consequent states of source nodes are derived from the signs of the equivalent paths, non-activatable successor paths, and primary deviation paths<sup>30</sup> terminating at the sensor signal. A table of transitions for different combinations of signs of these paths is presented in Appendix J. A root cause accounts for the transition of the result node if a path which (i) initiates from the root cause and goes along a primary deviation path to a sensed variable of the source node, and (ii) continues along a successor

---

<sup>30</sup>Primary deviation paths terminating at the sensor signal that are associated with transitions to the normal state.

path to the result node, can be constructed.<sup>31</sup> Other potential root causes that initiate primary deviation paths to the source and result nodes are associated with the not link.

On the other hand, :only-if-exists links derive from activatable successor paths. The allowable transitions are derived as for :not links. Potential root causes that activate the relevant successor paths are associated with the :only-if-exists link.

The fourth step in deriving the PM is to deduce additional potential root causes associated with :not links. These root causes are derived using the matrix of allowable state transitions. Root causes that initiate primary deviation paths to the source or result node but cannot lead to a transition to the consequent state of either node associated with the link are added to the list of root causes associated with the link.

Finally instances of :only-if-transient links are created. :only-if-transient links are associated with sensor signals that cannot display inverse or compensatory response to process disturbances. The links represent an event in which the sensor signal undergoes a transition from a non normal state (high or low) to the normal state. This information is contained in the matrix of allowable state transitions.

### 6.6.3. Features of the PM of a Heater Process

We illustrate typical relationships derived by the algorithm by considering the PM of a process feed preheater shown in Fig. 6.14. The hot process effluent exchanges heat with the feed stream in a countercurrent heat exchanger. The temperature of the process feed is controlled by manipulating the flow of process effluent through the heat exchanger.

The SDG of this process was created by linking together 16 units from the model library. The SDG consists of 67 root causes, 91 variables (of which 6 are sensor signals) and 113 causal arcs. After the SDG was condensed by removing 22 variables and 28 arcs, 29 additional ESDG arcs were created. The ESDG was subsequently translated to the PM. In addition to 67 potential root causes and 6 measured variables, the PM consists of 73 compiled causal links of which 15 are activatable. 8 events, 2 associated with status

---

<sup>31</sup>The sign of this primary deviation path and the constructed path must be consistent with the transitions of the source node and result node.

transitions and 6 with state transitions are defined for each measured variable. A distribution of the number of local causes for each abnormal event is shown in Fig. 6.15a. Figure 6.15b shows a distribution of the number of compiled causal links initiating at each event.

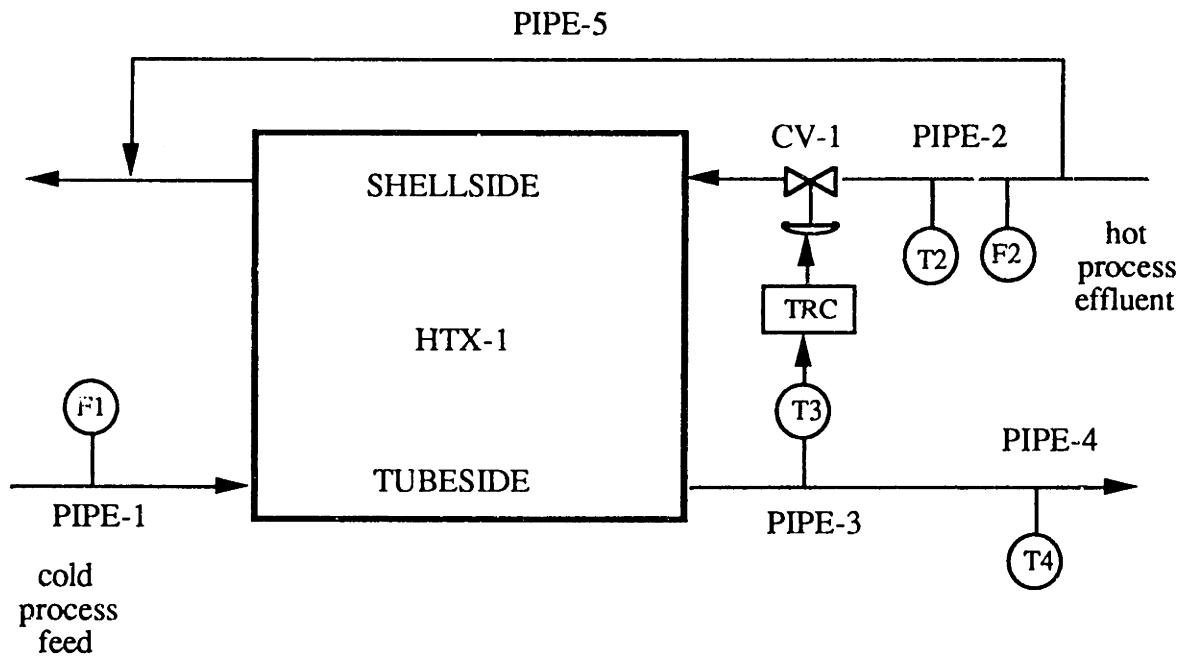


Fig. 6.14. Process Feed Preheater

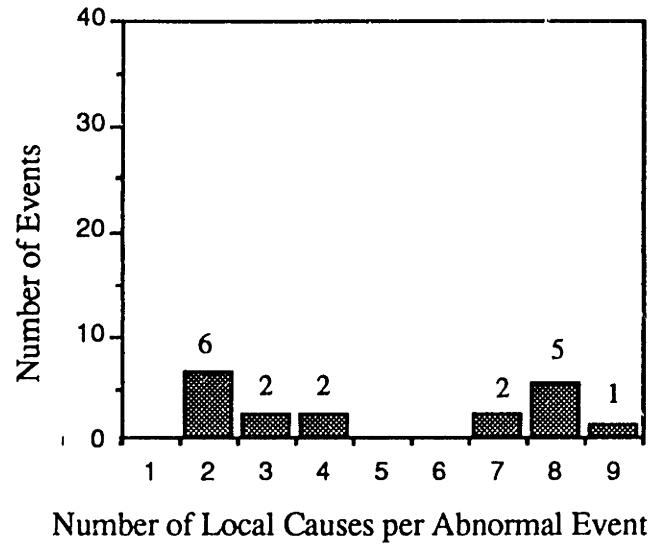


Fig. 6.15a. Distribution of Local Causes

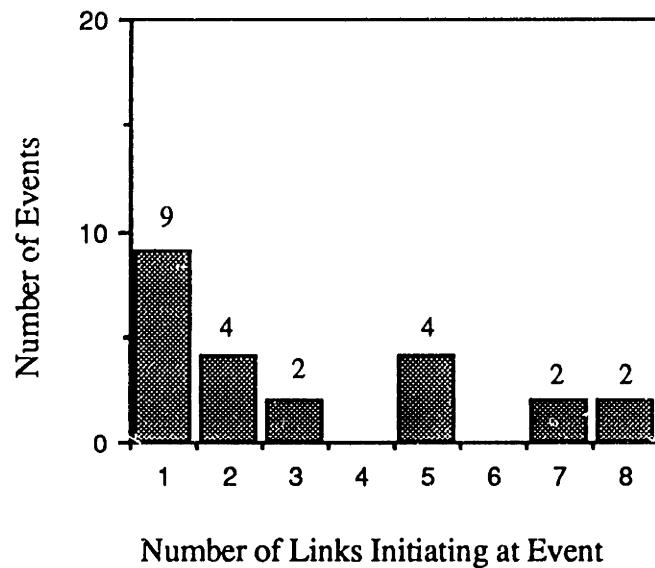


Fig. 6.15b. Distribution of Compiled Causal Links

Some of the relationships represented in the PM are shown in Fig. 6.16.

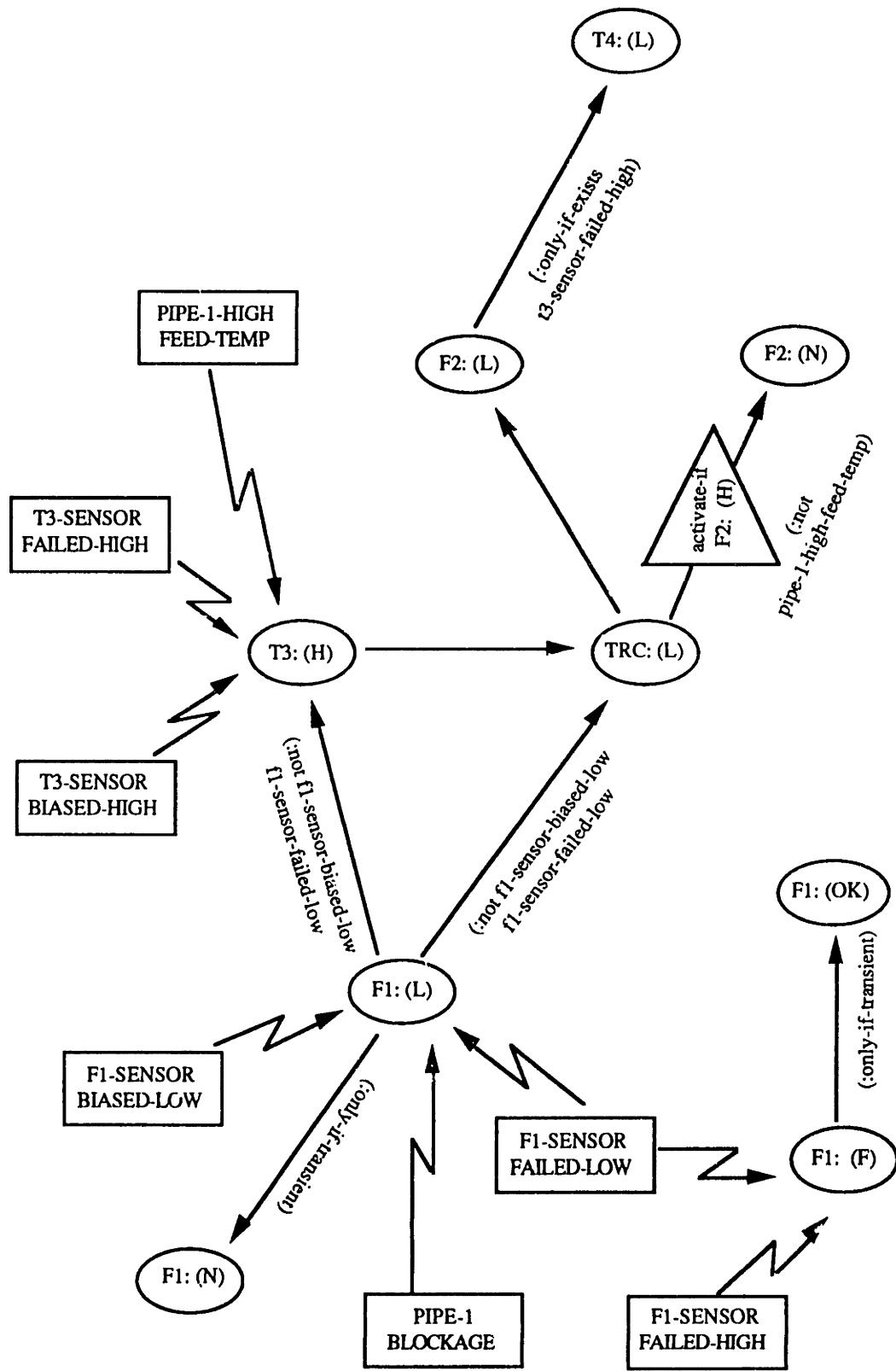


Fig. 6.16. Portion of PM of Feed Preheater

In the following, these relationships are discussed in more detail.

**Local Cause Relationships:** Eight of the local cause relationships in the PM are shown in Fig. 6.16. For example, **f1-sensor-failed-low** is a local cause of **f1-sensor-normal-to-low** and **f1-sensor-ok-to-failed**. Local cause links initiate from this potential root cause to both events.

**Status Transitions:** The section of the PM associated with status transitions of with f1-sensor is shown. Potential root causes **f1-sensor-failed-low** and **f1-sensor-failed-high** can lead to a failure of the f1 sensor signal. However, if the status of the sensor signal becomes OK, the sequence of status transitions is attributed to a process transient as indicated by the **:only-if-transient** link.

**:not Links:** When certain events occur, some potential root causes may be removed from the hypothesis set as indicated by the **:not** links. A successor relationships exists between **f1-sensor-normal-to-low** and **t3-sensor-normal-to-high**. When the latter event occurs, both **f1-sensor-failed-low** and **f1-sensor-biased-low** may be eliminated as viable root causes, as malfunctions in f1-sensor cannot propagate to t3-sensor.

**:only-if-exists Links:** During normal operations a successor relationships exists from events associated with f2-sensor to events associated with t3-sensor. However if t3-sensor fails these relationships no longer exists and t4-sensor events become successors of f2-sensor events. These relationships are **activated** by the failure of t3-sensor. These activatable-successor relationships are represented by **:only-if-exists** links. For example, the link between **f2-sensor-normal-to-low** and **t4-sensor-normal-to-low** exists only if **t3-sensor-failed-high**.

**:only-if-transient Links:** F1 does not exhibit inverse or compensatory response to any potential root cause in the system. Therefore an **:only-if-transient** link initiates from **f1-normal-to-low** and terminates at **f1-low-to-normal**.

**Activatable-Links:** The compiled causal link initiating at **trc-signal** and terminating at **f2-sensor** is an example of an activatable link. This link is not always active and thus cannot be invoked at all times during the diagnosis. The link may only be activated if the prior state of f2-sensor was high. This link represents a relationship between **trc-signal-normal-to-low** and **f2-sensor-high-to-normal**. The relationship between **trc-**

**signal-normal-to-low** and **f2-sensor-low-to-normal** is excluded because of the condition for which the link may be activated is violated.

**Global Causality:** Control systems are designed to transfer disturbances from outside the control loop over undeviated controlled variables to the manipulated variables. The compiled causal link from **f1-sensor-normal-to-low** to **trc-signal-normal-to-low** represents this non-local causality. Other situations where non-local causality may be exhibited in the process are represented in the ESDG and subsequently the PM.

#### 6.6.3.1. MODIFYING TEMPORAL ORDERING TO IMPROVE ROBUSTNESS TO OUT OF ORDER EVENTS

The algorithm that derives qualitative relationships in the PM (starting from SDG models in the model library<sup>32</sup>) is designed so that events in the PM are related in their ideal temporal order. The resulting PMs provide accurate diagnoses when event sequences occur in their ideal order. For example, in the PM of the feed preheater process (Fig. 6.16), the primary deviations of biases affecting **t3-sensor** are events associated with **t3-sensor**. The ideal sequence of events for these malfunctions begins with an event associated with **t3-sensor** followed by an event associated with **trc-signal**. Using the process model an accurate diagnosis is produced with the ideal sequence.

Events in the process may not always occur in their ideal order. If the event associated with **trc-signal** occurs before the event associated with **t3-sensor**, features of MIDAS inference algorithms guarantee that an accurate diagnosis is produced, provided an appreciable deviation in **t3-sensor** is detected. An appreciable deviation is detected when the magnitude of the bias is large enough or when the time scale of the malfunction is not much larger than the time scale of the (control) system response. If an appreciable deviation in **t3-sensor** is not detected, an inaccurate diagnosis results.

---

<sup>32</sup>Rules for specifying the delay of SDG arcs for process units in the model library are specified in Chapter 4.

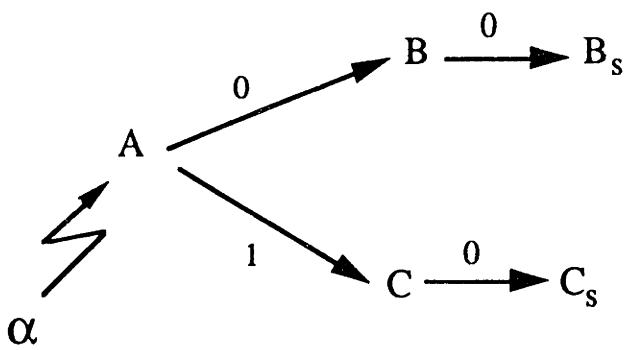


Fig. 6.17a. ESDG of Hypothetical Process



Fig. 6.17b. PM of Hypothetical Process

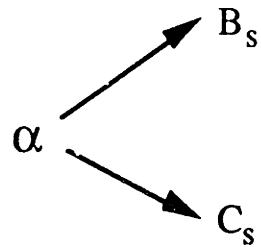


Fig. 6.17c. PM of Hypothetical Process  
(robust to out of order sequences)

The situation described above is an instance of the more general case where deviations associated with parallel paths in the ESDG are ordered according to their ideal temporal order. Figure 6.17a shows the ESDG of a hypothetical process. The time delays

associated with the arcs are depicted in the figure. The resulting PM reflects the temporal order of event transitions, and is shown in Fig. 6.17b. Because the delay of the deviation path from  $\alpha$  to BS is less than the delay from  $\alpha$  to Cs, events associated with sensor signal Cs are successors of events associated with sensor signal Bs. If Cs occurs and there is no appreciable deviation in Bs, no active causal link can be formed from  $\alpha$  to Cs, and an inaccurate diagnosis results.

To guarantee that events occur in their ideal order, ESDG arcs must express **will-cause** relationships. In other words, a deviation in the arc's upstream variable necessarily results in a deviation of the arc's downstream variable. However, arcs in the ESDG express **may-cause** relationships, as deviations in upstream variables do not necessarily result in deviations of downstream variables. Therefore, events cannot be guaranteed to occur in their ideal order. **The algorithm for deriving the PM implicitly assumes that arcs in the ESDG express will-cause relationships, thus, retaining the ideal temporal order of events.** The result is that relationships deriving from parallel paths in the ESDG (e.g. Bs or Cs in Fig. 6.17c) are replaced by more restrictive "sequential" relationships (e.g. Bs followed by Cs in Fig 6.17b).

Two approaches may be taken in order to improve on-line diagnostic performance using PMs derived by the algorithm when out of order sequences are observed. The first approach involves modifying the algorithm, and the second, selectively modifying the SDG unit models. Modifying the algorithm is more general and, thus, is the preferred approach. However because SDG arcs express **may-cause** relationships rather than **will-cause** relationships, it is difficult to devise an algorithm that will produce a PM that is robust to out of order sequences while at the same time relate events in their ideal order. This can be seen from the PM in Fig. 6.17c. The PM is robust to out of order sequences, however, the temporal order of events is lost.

The approach taken was to selectively alter the time delay of arcs in the SDG. In our experience PMs derived using the model library usually produce accurate diagnoses from out of order sequences because appreciable deviations are observed for variables associated with missing events. Situations where out of order sequences have been observed with no appreciable deviation in intermediate variables, are associated with compensatory response

of controlled variables in control loops. The following modification<sup>33</sup> was made to the rules for developing unit models presented in Chapter 4:

- Time delays from the internal sensor variable to the sensor signal variable in sensor units belonging to control loops are assigned a delay of 1 rather than 0.

This results in a slight difference in the relationships expressed in the PM. On making this modification, one noticeable difference in the PM of the feed preheater process is the presence of a local cause link initiating from **t3-sensor-biased-high** to **trc-signal-low**, which is non-existent in Fig. 6.16. This ensures that an accurate diagnosis is produced even when an appreciable deviation in t3-sensor is not detected.

#### 6.6.3.2. MODIFYING TEMPORAL ORDERING TO REFLECT CAUSALITY

The PM does not represent causal relationships between events associated with sensor signals, but instead reflects the temporal order between these events. For example, in the PM of the gravity flow tank (Fig. 6.3), no causal relationships exists between sensors in the process. It may be desirable to derive PMs which reflect "intuitions" about the underlying causality among the variables that are measured by process sensors. This may be accomplished by modifying delays in the SDG models as follows:

- Time delays of arcs derived from differential equations with fast dynamics are assigned a delay of 1 rather than 0.

This modification does not alter diagnostic conclusions when events occur in their ideal order. An alternate set of model library algorithms that derive units according to the modified guidelines presented in Sections 6.6.3.1 and 6.3.3.2 was developed. The modified library was used to develop the PM of the process selected for the case study in Chapter 7.<sup>34</sup>

---

<sup>33</sup>In our experience this modification has been sufficient to produce an accurate diagnosis when event sequences occur out of ideal order.

<sup>34</sup>The diagnostic conclusions were identical for simple process where events occurred in their ideal order. However, a comparative evaluation of diagnostic performance using PMs developed using models from both libraries should be carried out.

## 6.6.4. Complexity of the Algorithm for Deriving the PM

In this section, the complexity of the algorithm for translating the ESDG to qualitative relationships in the PM is discussed. Strategies adopted to reduce the time taken to translate the ESDG into the PM are suggested.

Deriving relationships in the PM entails search over valid acyclic paths from root causes to sensor signals in the ESDG and from sensor signals to other sensor signals. In order to ensure correctness, the search procedure explicitly enumerates all acyclic paths evaluating causal relationships for each path before determining aggregate relationships expressed in the PM. This results in the following approximate expression for the total time taken for search:

$$T \approx f_p(L) \cdot b^L \quad (6.13)$$

where  $L$  is the length of the path and  $b$ , the geometric average branching factor.  $f_p$  is a polynomial function of  $L$ . Here, the branching factor is the number of arcs initiating from a variable that do not lead to the formation of a cycle.

The algorithm may be decomposed into its basic steps when in evaluating the total time required to derive the PM. In deriving the matrix of allowable transitions, the decomposition of the ESDG into its SCCs is exploited. Matrices for each SCC are derived and then transitions are deduced from input-output relationships among the SCCs. The matrices for each SCC are obtained by search through standard SDG arcs and information explicit relationships of disturbance variable/inverse variable and disturbance variable/compensatory variable pairs. Thus the time taken for this stage of the algorithm scales as

$$T = f_p(L_s) \cdot S \cdot (D + R_s) \cdot b_s^{L_s} \quad (6.14)$$

where  $S$  is the number of SCCs,  $D$  and  $R_s$ , the number of disturbance variables and root causes directly affecting variables in the SCC. An upper bound for  $L_s$  is the number of variables in the SCC, and  $b_s$  the branching factor for the SDG.

In finding primary deviation and successor paths, all arcs in the ESDG are searched in order to maintain causal ordering in the PM. The time taken for both these steps scales as

$$T \approx f_p(L_d) \cdot (R + SV) \cdot bd^{L_d} \quad (6.15)$$

R and SV are the number of root causes and sensor signals, and  $bd^{35}$  is bounded by the branching factor of the ESDG.  $L_d$  is the length of primary deviation or successor path.

The branching factor in the ESDG is always greater than or equal to the branching factor for the SDG of the corresponding process. For example, values for the branching factor of the SDG and ESDG of the feed preheater process are 1.47 and 1.63 respectively.<sup>36</sup> Furthermore, large processes tend to have a limit on the size of SCCs, hence  $L_s$ . Thus, the most critical parameters in deriving the PM of large processes are the branching factor,  $bd$ , of the ESDG and the average separation between sensor signals,  $L_d$ . The time taken to derive the PM may be reduced by decreasing either of both parameters.

Condensing the SDG before creating the ESDG reduces the average length of paths. By condensing the SDG, the time taken to translate the ESDG to the PM for the feed preheater process was reduced from 48 minutes to 27 minutes. This represents a reduction by a factor of 2 for this simple process.

## 6.7. Specifying Quantitative Constraints and Tests in the PM

The final step in developing the PM is to enhance the qualitative relationships with quantitative constraint information. Constraints improve diagnostic resolution by eliminating potential root causes from the hypothesis set, consistent with the qualitative relationships in the PM. During this stage the relationships between the results of visual inspections and malfunctions may also be specified. Detailed instructions for specifying these relationships using an input specification language are provided in Chapter 10.

---

<sup>35</sup>The search does not necessarily involve all ESDG arcs. When arcs with delay 0 and 1 initiate from the same variable search is along the 0 delay arc until an arc with delay 1 is reached.

<sup>36</sup>These branching factors are for the uncondensed SDG and the resulting ESDG.

In this section, guidelines for deriving relationships between quantitative constraints and process malfunctions are presented. First, in Section 6.7.1, we describe how constraint relationships in the PM are represented. Guidelines for deriving constraint relationships are presented in Section 6.7.2. Finally, in Section 6.7.3, we conclude by describing how relationships between tests and malfunctions are represented in the PM.

### 6.7.1. Representing Constraints in the PM

In this section, the representation of quantitative constraint information in the PM is described. Most continuous processes may be described by a set of constraints relating its signal variables,  $x$ .

$$F(x) = R \quad (6.16)$$

Generally, the set  $F$  is a combination of different types of constraints, including algebraic, integral and differential equations. The constraints are derived so that under nominal conditions, the residual for each expression is less than the specified tolerance:

$$|R_i| < tol_i ; i = 1,2,\dots \quad (6.17)$$

Typically, a process disturbance or malfunction results in the violation of one or more constraints in the process. Thus, violated constraints represent a source of diagnostic information. The combined use of constraints and qualitative information results in more redundancy relations among process measurements. Constraints potentially provide additional information not contained in the qualitative PM and may eliminate potential root causes that are consistent with the qualitative model, from the hypothesis set. Combining constraints with qualitative information is facilitated by the event-oriented representation of the PM [3].

The formulation of constraints is quite general, as qualitative causal relationships expressed in the PM may be explicitly represented as constraints.

$$x_d = x - x_{ss} < tol \quad (6.18)$$

Equation (6.18) expresses the deviation of  $x$  from its steady state value,  $x_{ss}$ . With this transformation, the behavior of the sensor signal,  $x$ , is described by the behavior of its corresponding deviation variable,  $x_d$ .

The representation of constraint residuals is similar to the representation of signal variables (Table 6.4). Like deviations variables non-monotonic behaviors are allowed by constraint residuals. Figure 6.18 shows a PM describing state transitions for constraint residuals, MC-1, MC-2 and MC-3 to their respective potential root causes. The initial transition of each of the residuals in response to its respective root cause is from normal to high. MC-1 does not exhibit non-monotonic behavior, MC-2 exhibits compensatory response and MC-3 exhibits inverse response. Other conditions, such as not conditions may be associated with the compiled causal links in situations where a subset of constraints in the current hypothesis set may result in subsequent transitions.

Generally, the following must be specified for each constraint residual:

- A list of potential root causes that results in an initial state transition of normal to high.
- Root causes resulting in compensatory response of the residual for which the initial state transition is high.
- Root causes resulting in inverse response of the residual for which the initial state transition is high.
- A list of potential root causes that results in an initial state transition of normal to low.
- Root causes resulting in compensatory response of the residual for which the initial state transition is low.
- Root causes resulting in inverse response of the residual for which the initial state transition is low.
- Potential root causes that may result in status transitions of sensor signals associated with the constraint.

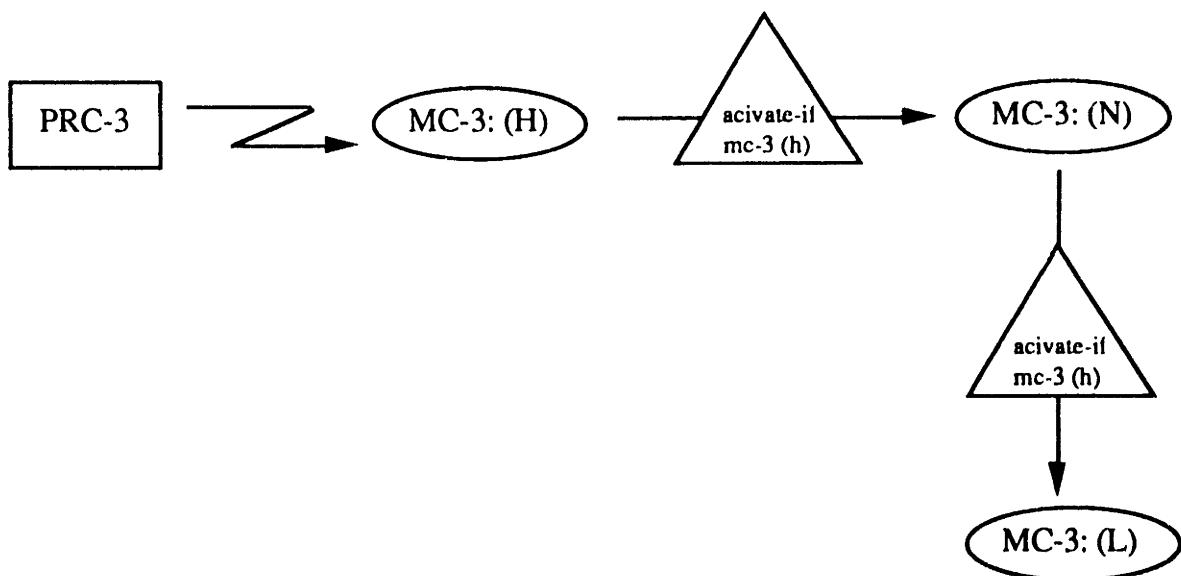
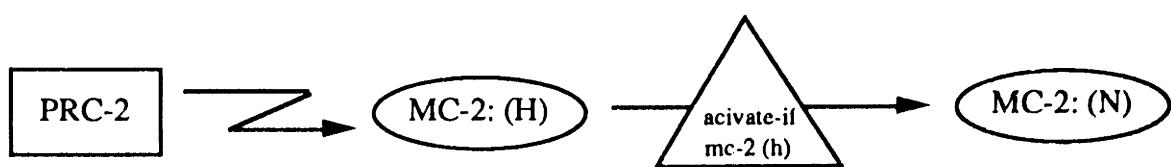
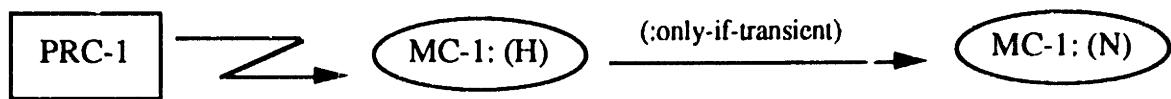


Fig. 6.18. Representing Dynamic Behavior of Constraint Residuals

## 6.7.2. Guidelines for Specifying Constraint Information

In this section, we present guidelines for specifying dynamic relationships between constraint residuals and potential root causes. Unlike qualitative relationships in the PM, the ability to derive constraints is highly limited by the location of sensors in the process. Residuals can usually be formulated so that they do not exhibit non-monotonic behavior to process malfunctions. Thus, our focus will be on deriving the initial state transitions associated with malfunctions.

The first step in deriving relationships between root causes and constraint residuals is to specify the constraints. Conventional mathematical modeling of continuous chemical processes from fundamental principles has been thoroughly studied and will not be addressed here. Luyben [19] and Franks [20] provide excellent introductions to this field. By combining these principles with empirical modeling studies a set of equations describing normal process operation may be derived. Unmeasured variables may be eliminated from the system of equations to obtain a set of equations which consists solely of sensor signals and known process parameters. Equation (6.16) forms the basis for relationships between constraint residuals and root causes in the PM.

Next root causes that may result in violations of individual constraint residuals are specified. For each residual, the set of root causes is a subset of all root causes that may causally affect<sup>37</sup> sensor signals that are expressed in the constraint. These root causes may be classified as root causes indicating a malfunction of the sensor signal (sensor malfunctions) and other root causes (process malfunctions).

For a malfunction in sensor  $x_j$ , the output signal may be expressed as:

$$x_{jm}(t) = x_j(t) + \xi_j(t) \quad (6.19)$$

where  $\xi_j(t)$  is the profile of the parameter indicating the sensor malfunction. Substituting into eq. (6.16), the qualitative sign of the residual of each constraint involving the sensor,  $R_i$ , is

---

<sup>37</sup>These root causes can be obtained from the process SDG.

$$[ R_i ] = [ f_i(x_m) ] = [ \partial f_i / \partial x_j ] \cdot [ \xi_j(t) ] \quad (6.20)$$

The initial state transition associated with the residual derives from the sign of the residual.

For process malfunctions, each constraint in eq. (6.16) becomes

$$f_i(x, \xi_j(t)) = R_i ; \quad x_k = x_{km} , \forall k \quad (6.21)$$

$\xi_j(t)$  indicates the process malfunction. The qualitative sign of the residual of each constraint affected by the malfunction is

$$[ R_i ] = [ f_i(x_m) ] = - [ \partial f_i / \partial \xi_j ] \cdot [ \xi_j(t) ] \quad (6.22)$$

The initial state transition also derives from the sign of the residual.

### 6.7.3. Specifying Tests in the PM

Visual and other similar inspections may be ordered by the process operator. The relationships between outcomes of these inspections<sup>38</sup> and potential root causes may be specified in the PM. The outcomes may be used to refine the hypothesis obtained using evidence from process observations and knowledge about qualitative and constraint relationships expressed in the PM.

The result of each inspection may be positive or negative. The following must be specified for each test:

- A list of root causes whose existence may be confirmed by the test.
- A list of root causes whose existence is contraindicated by the test.
- A list of tests that are excluded by the test (i.e. mutually exclusive tests).

---

<sup>38</sup>The outcomes of the inspections are referred to as tests.

## **6.8. Improvements to Procedures for Deriving the Qualitative Core of the PM**

The previous four sections described in detail the procedures and algorithms used in deriving relationships represented in the PM. Most of time and effort in developing the algorithms was spent on algorithms that derive the ESDG from the SDG and translate the ESDG to the PM. The unit model library algorithms and the programs that specify quantitative constraint and test relationships using batch input files were of secondary importance and will not be discussed in this section.

Several avenues exist for making improvements to procedures and algorithms used in creating the qualitative core of the PM. In this section, possible improvements to the algorithms are suggested. The improvements include aspects for which theoretical results were derived but were not implemented. Other aspects relate to improving the efficiency<sup>39</sup> of the algorithms that create the ESDG from the SDG and translate the ESDG to the PM. In the following, we discuss these improvements.

### **6.8.1. Transitions Across Qualitative Regimes**

The algorithm implemented for deriving additional ESDG arcs from the SDG is based on the criteria presented in Section 6.5.3. These criteria assume that transitions from the nominal qualitative regime do not occur during process transients. This assumption may be violated in certain situations, especially when control loops become saturated. The current algorithm may be enhanced, so that it creates additional ESDG arcs that account for process behavior during control loop saturation.

Theorems which extend the basic criteria when control loops are saturated were derived in Section 3.5. These theorems indicate that additional ESDG arcs are needed only in situations where a variable that does not exhibit IR in its nominal qualitative regime exhibits IR as a result of the transition. In addition, the extensions indicate that identifying variables that may exhibit IR as a result of controller saturation, is necessary only in strongly connected components of the SDG that contain positive cycles and control systems.

---

<sup>39</sup>Here we refer to improvements other than reimplementing the algorithms in a faster programming language or on a faster hardware platform.

Modifications to the current algorithm should create additional ESDG arcs for all qualitative regimes<sup>40</sup> that may be traversed during the transient. Associated with each of these arcs are conditions relating to the qualitative regime for which the arc is valid. Modifications should also be made to the algorithm translating the ESDG to the PM. Compiled causal links which derive from paths containing ESDG arcs conditional on the saturation of control loops should reflect these conditions.

### 6.8.2. Improving the Algorithm for Deriving the ESDG

Creating the ESDG for large scale process systems starts from a process SDG with more variables and arcs. Because analysis takes place at the level of individual SCCs, the time taken to construct the ESDG for large processes can be expected to grow linearly with problem size if there is a bound on the size of the SCC. After initial decomposition of the SDG into its SCCs<sup>41</sup>, the complexity of the algorithm depends on the size and connectivity of the SCC.

In Section 6.5.5, a method for reducing the time for creating the ESDG was presented. The technique condensed the SDG before applying the algorithm for creating the ESDG. The solution significantly reduced the time taken to create the ESDG. However, it may still take "too long" to create the ESDG of systems containing large SCCs using this strategy. In this section, two other strategies for reducing the time are suggested.

---

<sup>40</sup>These qualitative regimes are due to combinations of (non)functioning individual control loops in the SCC.

<sup>41</sup>Linear search algorithms exist [16] to decompose a directed graph into its SCCs. Cycles in an SCC may be found after decomposition of the SDG into its SCCs.

### 6.8.2.1. DEVISING A MORE EFFICIENT ALGORITHM

In Section 6.5.5, it was shown that ESDG arcs for a SCC can be constructed in  $O(|DI| \cdot |API| \cdot |LI| \cdot |(1 + PC)| \cdot |SVI|)$  time. The major factor contributing to the "long" time required to construct the ESDG is that explicit enumeration of all acyclic paths in the SCC during search initiating from disturbance variables to the SCC. The "optimal" strategy for reducing the time is to devise a more efficient algorithm for creating the ESDG.

It may be possible to devise a search algorithm with an order of growth that is linear in the number of arcs in the SCC. However, in performing search, information about disturbance variables and their respective IVs and CVs, and the location of ESDG arcs (with their associated conditions) must be determined. Evaluating IVs and CVs, and conditions associated with ESDG arcs should be performed in a non-combinatorial manner. This may require that the criteria for locating IVs and CVs be recast in an alternative equivalent form.

### 6.8.2.2. DECOMPOSING LARGE SCCS

The optimal strategy for reducing the time required to create the ESDG was discussed in the previous section. Failing this, another solution that may be adopted for systems with large SCCs is to decompose the SCC, creating ESDG arcs for the smaller SCCs.

In order to decompose the SCC, a set of arcs should be selected so that removing the arcs from the SCC results in the partitioning of the SCC into two or more SCCs. We define each of these sets of arcs as **sets of separation arcs** of the SCC. Several criteria may be used to decide among different sets of separation arcs:

- The set should contain as few arcs as possible.
- The set should result in the largest sub-SCC having the smallest size.
- The delay of the negative loop with the shortest delay in the original SCC containing arcs in the separation set should be as large as possible.

After decomposition, ESDG arcs are created for the resulting SCCs, after which the separation arcs are reconnected. The resulting ESDG may be different from the ESDG derived using the original SCC. For example, the resulting ESDG may contain additional arcs if an integrator suppressing IR is separated from the positive cycle causing IR. Similarly, some ESDG arcs may be excluded if a negative feedback loop containing an integrator causing CR contains one of the separation arcs.

In general, decomposing the SCC before creating the ESDG potentially results in a modified set of non-local global qualitative effects. This potentially affects the quality of diagnosis. However if the affected global interactions are associated with very long process time constants, the quality of diagnosis may not be adversely affected. In particular, when the time scale associated with the malfunction origin is smaller than time scale of the affected interaction, diagnostic quality is not affected. This is because information from the evolution of events would have been used to resolve the diagnosis, before events attributed to these interactions are detected.

Where, the diagnostic quality may be affected the original ESDG should be reconstructed based on the location of the separation arcs and structures in the sub-SCCs. Deriving criteria for reconciling differences between the ESDG obtained after decomposition of SCCs and the original ESDG is a subject for further research.

### **6.8.3. Improving the Algorithm for Deriving the PM**

In this section, two issues involved in translating the ESDG of large scale process systems into PMs are discussed. One pertains to deriving more compact PMs. This results in faster on-line diagnosis by MIDAS. The other relates to improving the efficiency of the algorithm for deriving the PM from the ESDG.

### 6.8.3.1. REDUCING THE CONNECTIVITY OF THE PM

Inference in MIDAS basically entails relating new events observed in the process to existing events by searching through successor and precursor links in the PM, and applying actions associated with the links to the current diagnostic hypothesis [3]. Thus any reduction in the number of links and the number of potential root causes associated with link actions results in faster on-line diagnostic performance. In this section, we suggest modifications to the algorithm that would result in more compact PMs.

The approach described in Section 6.6. for deriving the PM is conservative and thus, the resulting PM contains redundant relationships. One source of redundancy is the existence of intra-variable and inter-variable causal links terminating at events with normal consequent states. Eliminating some of these inter-variable compiled causal links significantly decreases the connectivity of the PM and results in a faster diagnosis by MIDAS. It may be possible to modify the algorithm so that the resulting PM contains no inter-variable links terminating at events with normal consequent states, without adversely affecting the causal ordering among the events in the PM.

Another source of redundancy is the association of some potential root causes with certain :not links in the PM. Irrespective of the previous sequence of events, these root causes can never be listed in the current hypothesis set when the :not link is invoked during diagnosis. This occurs when no upstream causal path in the PM exists from the potential root cause to the :not link. Additionally, if the root cause is associated with at least one :not link on all upstream paths from the potential root cause to the :not link, associating the root cause with the :not link is redundant. The algorithm may be modified so that these potential root causes are not associated with particular :not links in the resulting PM. In addition to a faster diagnosis, the memory storage requirements for the PM is potentially reduced.

### 6.8.3.2. DEVISING A MORE EFFICIENT ALGORITHM

In Section 6.6.4, it was shown that the ESDG can be translated to the PM in  $O(bd^{L_d})$  time. The strategy adopted to reduce this time for large scale processes was to condense the SDG (hence the ESDG). This results in a "smaller" average separation,  $L_d$ , between sensor signals in the ESDG and an overall reduction in the time required to create the PM.

Alternatively, the time taken to derive the PM may be reduced by decreasing the average branching factor during search. This may be achieved by developing an algorithm that avoids search through additional non-physical ESDG arcs. Knowledge about non-monotonic behavior exhibited by process variables due to feedback effects is (implicitly) represented in non-physical ESDG arcs. Instead of searching through paths containing these non-physical arcs, knowledge about these behaviors as is (explicitly) represented in slots<sup>42</sup> of (E)SDG variables could be used to infer relationships involving these variables. It may be possible to modify the algorithm, avoiding search through the non-physical arcs without adversely affecting the causal ordering among events in the PM.

None of the above strategies is optimal as it may still take "too long" to translate the ESDG to the PM. A better approach is outlined below.

The major factor contributing to the "long" time required to translate the ESDG to the PM is the explicit enumeration of all acyclic paths in the ESDG initiating from all potential root causes and sensor variables during the search for primary deviations and successors. An "optimal" strategy is to devise a more efficient algorithm for deriving relationships in the PM.

It may be possible to devise a search algorithm with an order of growth that is linear in the number of ESDG arcs. However, during search, the relationships expressed in the PM must be derived. These relationships should be deduced in a manner that does not require combinatorial evaluation of conditions associated with individual ESDG arcs.

---

<sup>42</sup>These slots are the `displays_inverse_response_to`, `causes_inverse_response_of`, `displays_compensatory_response_to` and `causes_compensatory_response-of` slots.

## 6.9. Conclusion

In this chapter, a structured approach for expressing process behavioral knowledge in the modeling formalism of the Model Integrated Diagnostic Analysis System was presented. The approach mainly utilizes process independent information and significantly increases the reliability and reduces the time and effort in developing diagnostic knowledge base when compared with conventional rule based expert systems. The task of developing the core of the knowledge base is reduced to specifying the flowsheet structure and the signs of certain parameter relationships, such as relative temperatures in adjacent process units.

The starting point for deriving the process model is from unit models describing local causality. One very important aspect of the procedures used in the approach is the uncovering of all "apparent non-local" causalities stemming from global interactions in the process. The representation of these global effects in the process model ensures an accurate diagnosis is produced by MIDAS when variables exhibit non-monotonic dynamic behavior such as inverse or compensatory responses. Additionally, the process model relates process observations directly to potential root causes affecting the process and its derivation is independent of the existing process instrumentation scheme. This avoids postulating and retracting assumptions about unmeasured variable states during diagnosis. Diagnostic resolution is improved by incorporating quantitative constraint information using a uniform representation.

Suggestions for reducing the time taken to develop process models of large scale process systems, using the approach were made. In Chapter 7, we verify process models developed using the approach. The diagnostic performance of MIDAS using the process models is characterized in the context of a case study of an example process.

## 6.10. Notation and Abbreviations for Chapter 6

### 6.10.1. List of Abbreviations

AMD	activatable minimum delay deviation path
AP	acyclic path in strongly connected component

CR	compensatory response
CV	compensatory variable
DP	deviation path in extended signed directed graph
ESDG	extended signed directed graph
IR	inverse response
IV	inverse variable
MC	minimum cost deviation path
MD	minimum delay deviation path
MIDAS	Model Integrated Diagnostic Analysis System
PM	process model
RBES	rule based expert system
S	Sensor signal
SCC	strongly connected component
SDG	signed directed graph

### 6.10.2. Notation

a	arc in the ESDG
A	system matrix
b	branching factor in (E)SDG
B	input matrix
D	disturbance variables to strongly connected component
F, F <sub>p</sub>	function, polynomial function
f, F	volumetric flowrate
H	subsystem matrix
l	tank level
L	number of variables in acyclic path
n	node in ESDG
P	pressure at node
P ,  Q	principal minor of subsystem matrix
PC	positive cycle in strongly connected component
R	constraint residual
S	number of strongly connected components
SV	number of variables in strongly connected component
T	temperature

$t, T$	time
$TD$	time delay of path
$u$	input variable
$V$	variable in acyclic path
$x, X, Y$	process variable

### Special Symbols

$\forall$	universal quantifier, for all
$\exists$	existential quantifier, there exists
$\in$	element of set
$\notin$	not an element of set
$\neg$	not
$\wedge$	and
$\vee$	or
$\xi$	parameter indicating process malfunction
$[ ]$	sign of expression
$   $	absolute value of expression
$\cdot$	qualitative multiplication

### Subscripts

$d$	deviation from steady state value
$i, j, k, \dots$	dummy variables
$m$	process observation
$ss$	steady state value

## 6.11. References for Chapter 6

1. Davis, R. (1984). Diagnostic reasoning based on structure and behavior. Artificial Intelligence 24, 347-410.
2. Kramer, M.A., F.E. Finch and O.O. Oyeleye (1989). Operating Manual and User's Guide for MIDAS (version 1.0a). Massachusetts Institute of Technology, Laboratory for Intelligent Systems in Process Engineering.
3. Finch, F.E. (1989). Automated Fault Diagnosis of Chemical Process Plants using Model-Based Reasoning. Sc.D. Thesis, Massachusetts Institute of Technology.
4. Iri, M., K. Aoki, E. O'Shima and H. Matsuyama (1979). An algorithm for diagnosis of system failures in the chemical process. Comput. & Chem. Eng. 3, 489-493.
5. Kramer, M.A. (1986). Integration of heuristic and model-based inference in chemical process fault diagnosis. Proc. IFAC Workshop on Fault Detection and Safety in Chemical Plants, 111-115, Kyoto, Japan.
6. Goff, K.W. (1985). Artificial Intelligence in Process Control. Mechanical Engineering, October, 53-57.
7. Moore, R.L. and M.A. Kramer (1986). Expert systems in on-line process control. Chemical Process Control CPC-III, 839-865, M. Morari and T.J. McAvoy (Eds), Elsevier.
8. Fink, P.K. (1985). Control and integration of diverse knowledge in a diagnostic expert system. Proc. 9th IJCAI 426-431.
9. Gertler, J. and D. Singer (1985). Augmented models for statistical fault isolation in complex dynamic systems. Proc. Amer. Control Conf. 317 -322, Boston, MA.
10. Dhujarti, P.S., D.E. Lamb and D.L. Chester (1987). Experience in the development of an expert system for fault diagnosis in a commercial scale chemical process. in Proc. 1st Intl. Conf. on the Foundations of Computer-Aided Process

Operations 589, Park City, UT. G.V. Reklaitis and H.D. Spriggs (Eds). Elsevier, Amsterdam.

11. Berenblut, B.J. and H.B. Whitehouse (1977). Monitoring process plant based on a decision table analysis. The Chemical Engineer, March, 175-181.
12. Rosenberg, J., R.S.H. Mah and C. Iordache (1987). Evaluation of schemes for detecting and identifying gross errors in process data. Ind. Eng. Chem. Res. **26**, 555-564.
13. Polenta, H.P., A. Ray, P.T. Menadier, J.A. Benard and D.D. Lanning (1986). Implementation of a fault detection procedure. Proc. Am. Contr. Conf. 176-181, Seattle, WA.
14. Gold Hill Computers Inc. (1987). Goldworks Expert System Interfaces and Examples (Version 1.0 Developer 286/2.4)
15. Abelson, H., G.J. Sussman and J. Sussman (1985). Structure and Interpretation of Computer Programs. MIT Press, Cambridge, MA.
16. Even, S. (1979). Graph Algorithms. Computer Science Press, Rockville, MD.
17. Papadimitriou, C.H. and K. Steiglitz (1982). Combinatorial Optimization; Algorithms and Complexity. Prentice-Hall Inc, Englewood Cliffs, NJ.
18. Winston, P.H.K. (1984). Artificial Intelligence. Addison Wesley, Reading, MA.
19. Luyben, W.L. (1973). Process Modeling, Simulation, and Control for Chemical Engineers. McGraw-Hill, New York.
20. Franks, R.G.E. (1966). Mathematical Modeling in Chemical Engineering. John Wiley, New York.

## 7. MIDAS Case Study

In Chapters 5 and 6, an on-line diagnostic aid for continuous refinery and chemical processes, the Model Integrated Diagnostic Analysis System (MIDAS), was described. Using artificial intelligence programming techniques, MIDAS is specifically designed to diagnose malfunctions in the dynamic process environment. MIDAS is a "deep knowledge" system combining qualitative and quantitative process information. To assist in "knowledge engineering", MIDAS has built-in process independent algorithmic techniques for developing the qualitative "core" of the process diagnostic knowledge base. Using these techniques, the task of developing the core of the knowledge base is reduced to specifying the flowsheet structure and signs of parameter relationships, such as relative temperatures in adjacent process units. The techniques allow non-monotonic dynamic process behaviors<sup>1</sup> to be represented in the knowledge base, and knowledge of these behaviors to be utilized to produce a diagnosis. Representing knowledge about these behaviors allow an accurate diagnosis with a high degree of resolution to be obtained. In this chapter, we verify the techniques for developing the knowledge base by presenting the results of an extensive case study detailing the diagnostic performance of MIDAS.

The performance of MIDAS was evaluated for a simulated jacketed continuous stirred tank reactor process. In 40% of the runs, at least one variable exhibited **non-monotonic dynamic behavior** in response to the malfunction. Based on a sample of 77 cases, (i) an accurate initial diagnosis was achieved in 93% of the cases, (ii) an accurate final diagnosis was achieved in 100% of the cases and (iii) on average, 96% of the malfunctions in the process were eliminated by the end of the transient.

### 7.1. Introduction

The Model Integrated Diagnostic Analysis System [1], is an operators' aid for malfunction diagnosis in continuous chemical and refinery processes. Using artificial intelligence techniques, inference in MIDAS is designed to address problems not treated in previous diagnostic systems, including [2]:

- Robustness of diagnostic conclusions to symptom variations, out of order events

---

<sup>1</sup>Namely, inverse and compensatory responses.

and false alarms.

- Diagnosis of multiple malfunctions, particularly simultaneous independent malfunctions and induced sensor failures.

In order to obtain a diagnosis, MIDAS relies on a process knowledge base<sup>2</sup> consisting of qualitative causal and quantitative constraint relationships. The qualitative core of the process model is constructed off-line using the algorithmic techniques outlined in Chapter 6. User input involves specifying the flowsheet structure and signs of certain parameter relationships, such as relative temperatures in adjacent process units. One important feature of the algorithms is the identification of all apparent non-local causalities stemming from global interactions in the flowsheet. These non-local causalities result in process variables exhibiting non-monotonic dynamic responses, specifically, inverse, control system and other compensatory responses. The resulting model consists of qualitative relationships between process observations and malfunctions, and may be supplemented by specifying quantitative constraint information.

The model building technique enables MIDAS to:

- Represent non-monotonic dynamic behaviors exhibited by variables in the process knowledge base.
- Utilize knowledge of the dynamic behaviors, producing an accurate diagnosis with a high degree of resolution.
- Obtain a diagnosis using only information from existing process sensors.

The model building technique produces knowledge bases that are more easily verified for completeness and consistency when compared to rule sets in conventional rule based expert systems. The effort expended in interviewing domain experts for expert systems is largely reduced to specifying the flowsheet structure and signs of certain parameter relationships such as relative temperatures in adjacent units. Additionally, the time and expense required

---

<sup>2</sup>The knowledge base is also referred to as the Process Model (PM).

for process parameter identification in approaches based on conventional mathematical models is avoided.

The objective of this chapter is to verify the algorithmic techniques for developing the process model. The diagnostic performance of MIDAS for an extensive case study of a simulated jacketed reactor process is presented. Diagnostic resolution and accuracy achieved by MIDAS in the case study are reported.

In the following, the results of the case study are presented. In Section 7.2, we describe the development of the process model for the process selected for the case study. The steps carried out in performing the case study are described in Section 7.3. Next, an analysis of the results of the case study is presented in Section 7.4. Finally, in Section 7.5, fundamental limitations of qualitative modeling resulting in sub-optimal performance of MIDAS are addressed.

## 7.2. Jacketed Reactor Process

In this section, steps outlining the development of the process model (PM) for the process selected for the case study are described. We begin by discussing features of the process.

### 7.2.1. Process Description

The process selected for the study was a jacketed reactor process and its associated control systems. The system is shown in Fig. 7.1. At the heart of the process is a continuous stirred tank reactor in which two liquid phase, catalyzed, irreversible, first order reactions  $A \rightarrow B$  and  $A \rightarrow C$  take place. The primary reaction,  $A \rightarrow B$ , is exothermic. The rate dependence on temperature is described by an Arrhenius type relationship.

$$r_1 = k_1 \cdot \exp(-E_1/RT) \cdot CA \quad (7.1)$$

The endothermic side reaction,  $A \rightarrow C$ , is slow and nominally accounts for 0.23% of the consumption of A. The rate dependence on temperature is also described by an Arrhenius type relationship.

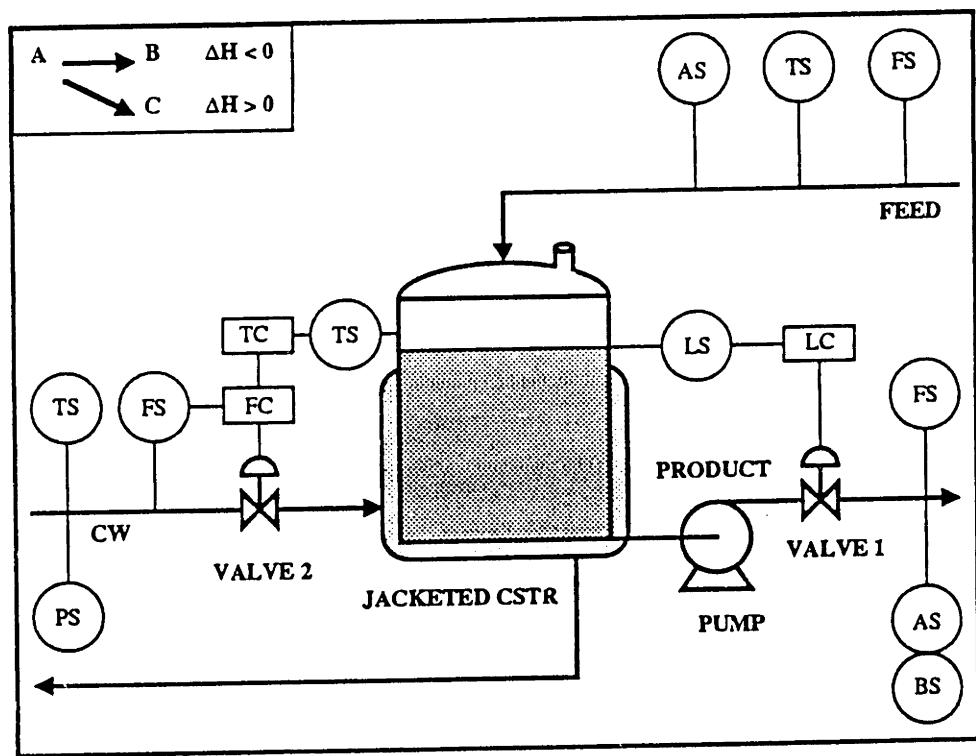


Fig. 7.1. Jacketed Continuous Stirred Tank Reactor Process

$$r_2 = k_2 \cdot \exp(-E_2/RT) \cdot C_A \quad (7.2)$$

The reacting mixture is cooled by cooling water flowing through the external jacket.

In order to provide temperature control, a cascade control system using measurements of the reactor temperature and cooling water flowrate manipulates the flow of cooling water to the jacket. Temperature is the primary controlled variable of the cascade loop and the set point of the cooling water flow controller is determined by the output of the temperature controller. The residence time of reactants in the reactor is controlled by maintaining the level of the reactants in the reactor. In addition to the flow, temperature and concentration sensors shown in the figure, output signals of all controllers are available for diagnosis.

Although limited in size, this process has several features which present a challenge for dynamic process diagnosis. A unique malfunction in the process can lead to different symptoms depending on its magnitude and extent. Different malfunctions may also result in the same set of symptoms. Furthermore, the process:

- Is highly non-linear.
- Has interacting<sup>3</sup> control loops.
- Has multiple causal pathways with opposing tendencies between variables. Positive feedback and non-control integrating effects are additional sources of non-monotonic dynamic behavior by the variables.
- Attains multiple steady states in the absence of temperature control.

### 7.2.2. Developing the Process Model

In this section, we summarize the development of the PM for the jacketed reactor process. The PM was developed using the procedures and algorithms presented in Chapter 6. First, we describe the construction of the qualitative core of the PM.

---

<sup>3</sup>Unidirectional interaction from the level control loop to the temperature control loop.

### 7.2.2.1. DEVELOPING THE QUALITATIVE CORE OF THE MODEL

The Signed Directed Graph (SDG) of the process was created by linking together 22 unit SDG models from the MIDAS unit model library.<sup>4</sup> The units selected were:

- 1 CSTR with external jacket.
- 1 Centrifugal Pump.
- 3 Controllers with proportional and integral action.
- 2 Control valves.
- 2 Feed streams.
- 2 Effluent streams.
- 11 Sensors.

Qualitative information about the reaction network, relative species concentrations and temperatures of the reactant feed stream and the reacting mixture were specified. In addition, the relationship between rates of the net heat generated by the reactions and the heat removed through the jacket was specified. The rate of heat generation was greater than the heat removal rate. The process SDG was created in about 1 hour<sup>5</sup> and consists of 107 root causes, 147 variables (14 sensor signals) and 185 causal arcs. A list of the root causes affecting the process is shown in Table 7.1.

---

<sup>4</sup>The unit models were selected from the alternate model library described in Section 6.6.

<sup>5</sup>Times quoted in this chapter refer to the current implementation in Goldworks on a PC compatible 386 (20MHz) computer.

**TABLE 7.1 Unit Failure Modes for Jacketed CSTR Process**

Unit	Failure Mode	Comments
Reactor Feed	Source Concentration A High	
Reactor Feed	Source Concentration A Low	
Reactor Feed	Source Pressure High	
Reactor Feed	Source Pressure Low	
Reactor Feed	Source Temperature High	
Reactor Feed	Source Temperature Low	
CSTR	Inlet Blockage	
CSTR	Outlet Blockage	
CSTR	Leak to Environment	Prior Probability = 0.0
CSTR	Loss of Mixing	
CSTR	Catalyst 1 Deactivation	
CSTR	Catalyst 2 Deactivation	
CSTR	Fire	
CSTR	Loss of Insulation	
Pump	Blockage	Prior Probability = 0.0
Pump	Fire	
Pump	Loss of Insulation	
Pump	Leak to Environment	
Pump	Cavitation	Prior Probability = 0.0
Pump	Overheating	
Pump	Motor Speed High	
Pump	Motor Speed Low	
Pump	Motor Failed	Prior Probability = 0.0
Pump	Shaft Broken	Prior Probability = 0.0
Valve 1	Blockage	Prior Probability = 0.0
Valve 1	Fire	
Valve 1	Loss of Insulation	
Valve 1	Leak to Environment	Prior Probability = 0.0
Valve 1	Stuck Open	
Valve 1	Stuck Closed	
Reactor Effluent	Sink Pressure High	
Reactor Effluent	Sink Pressure Low	
Jacket Feed	Source Pressure High	

TABLE 7.1 Unit Failure Modes for Jacketed CSTR Process (continued)

Unit	Failure Mode	Comments
Jacket Feed	Source Pressure Low	
Jacket Feed	Source Temperature High	
Jacket Feed	Source Temperature Low	
Valve 2	Blockage	
Valve 2	Fire	
Valve 2	Loss of Insulation	
Valve 2	Leak to Environment	Prior Probability = 0.0
Valve 2	Stuck Open	
Valve 2	Stuck Closed	
CSTR Jacket	Fouling	
CSTR Jacket	Leak to Reactor	
CSTR Jacket	Leak to Environment	
Jacket Effluent	Sink Pressure High	
Jacket Effluent	Sink Pressure Low	
Level Controller	Failed High	
Level Controller	Failed Low	
Level Controller	Biased High	
Level Controller	Biased Low	
Level Controller	Setpoint High	
Level Controller	Setpoint Low	
Temperature Controller	Failed High	
Temperature Controller	Failed Low	
Temperature Controller	Biased High	
Temperature Controller	Biased Low	
Temperature Controller	Setpoint High	
Temperature Controller	Setpoint Low	
CW Flow Controller	Failed High	
CW Flow Controller	Failed Low	
CW Flow Controller	Biased High	
CW Flow Controller	Biased Low	
Feed Flow Sensor	Failed High	

TABLE 7.1 Unit Failure Modes for Jacketed CSTR Process (continued)

Unit	Failure Mode	Comments
Feed Flow Sensor	Failed Low	
Feed Flow Sensor	Biased High	
Feed Flow Sensor	Biased Low	
Feed Temperature Sensor	Failed High	
Feed Temperature Sensor	Failed Low	
Feed Temperature Sensor	Biased High	
Feed Temperature Sensor	Biased Low	
Feed Concentration Sensor	Failed High	
Feed Concentration Sensor	Failed Low	
Feed Concentration Sensor	Biased High	
Feed Concentration Sensor	Biased Low	
Reactor Temperature Sensor	Failed High	
Reactor Temperature Sensor	Failed Low	
Reactor Temperature Sensor	Biased High	
Reactor Temperature Sensor	Biased Low	
Reactor Level Sensor	Failed High	
Reactor Level Sensor	Failed Low	
Reactor Level Sensor	Biased High	
Reactor Level Sensor	Biased Low	
Product Flow Sensor	Failed High	
Product Flow Sensor	Failed Low	
Product Flow Sensor	Biased High	
Product Flow Sensor	Biased Low	
Concentration A Sensor	Failed High	
Concentration A Sensor	Failed Low	
Concentration A Sensor	Biased High	
Concentration A Sensor	Biased Low	
Concentration B Sensor	Failed High	
Concentration B Sensor	Failed Low	
Concentration B Sensor	Biased High	
Concentration B Sensor	Biased Low	
CW Flow Sensor	Failed High	
CW Flow Sensor	Failed Low	

TABLE 7.1 Unit Failure Modes for Jacketed CSTR Process (continued)

Unit	Failure Mode	Comments
CW Flow Sensor	Biased High	
CW Flow Sensor	Biased Low	
CW Temperature Sensor	Failed High	
CW Temperature Sensor	Failed Low	
CW Temperature Sensor	Biased High	
CW Temperature Sensor	Biased Low	
CW Pressure Sensor	Failed High	
CW Pressure Sensor	Failed Low	
CW Pressure Sensor	Biased High	
CW Pressure Sensor	Biased Low	

Next, the Extended Signed Directed Graph (ESDG) of the process was created from the SDG, using the algorithm described in Section 6.5. Thirty eight (38) variables and 38 causal arcs were removed by condensing the SDG, after which it took less than 2 hours to create the ESDG. 69 additional non-physical arcs were created.

Finally, qualitative relationships in the PM were derived as described in Section 6.6. During this stage, interactive input was used to resolve ambiguities arising from root causes that could result in a subset of measured variables exhibiting non-monotonic behavior (as modeled by the ESDG). The program indicates that these variables could possibly attain both "high" and "low" states during the transient. The combinations of root causes and variables are shown in Table 7.2. Root causes not shown in the table produced unambiguous effects for all measured variables in the process. Variables not shown were uniquely determined for all root causes. Of the possible 1498 (107 x 14) entries, 220 (14.7%) resulted in ambiguity.

107 (48.6%) of the ambiguous entries were resolved. In the table, N indicates that the state of the variable is unambiguous. H, L and ? indicate ambiguous states of the variable. H and L indicate that the ambiguity was resolved, low and high states being excluded, respectively. Ambiguity was not resolved in cases where the entry is ?.

Table 7.2 Resolution of Ambiguity During PM Construction

	Product_Flowrate	Concentration_B	Concentration_A	Reactor_Temperature	Cooling_Water_Flowrate	CW_Flow_Controller_Signal	Cooling_Water_Setpoint	†
Reactor_Feed_Source_Concentration_A_High †	N	H	N	H	H	L	H	H
Reactor_Feed_Source_Concentration_A_Low	N	L	N	L	L	L	L	L
Reactor_Feed_Source_Pressure_High	N	H	H	?	?	?	?	?
Reactor_Feed_Source_Pressure_Low	N	L	L	?	?	?	?	?
Reactor_Feed_Source_Temperature_High	N	N	N	N	N	N	N	N
Reactor_Feed_Source_Temperature_Low	N	L	N	N	N	N	N	N
Jacket_Effluent_Sink_Pressure_High	N	N	N	N	N	N	N	N
Jacket_Effluent_Sink_Pressure_Low	N	L	N	N	N	N	N	N
CSTR_Inlet_Blockage	N	H	L	?	?	?	?	?
CSTR_Outlet_Blockage	L	N	N	?	?	?	?	?
CSTR_Leak_to_Environment	N	N	N	?	?	?	?	?
CSTR_Fire	N	H	N	N	N	N	N	N
CSTR_Jacket_Fouling	N	H	N	N	N	N	N	N
CSTR_Loss_of_Insulation	N	L	L	N	N	N	N	N
CSTR_Loss_of_Mixing	N	L	L	N	N	N	N	N
CSTR_Catalyst_2_Deactivation	N	N	L	L	L	L	N	N
CSTR_Jacket_Leak_to_Reactor	N	N	L	N	N	N	N	N
CSTR_Jacket_Leak_to_Environment	N	N	N	N	N	N	N	N
Pump_Blockage	L	H	L	L	L	L	L	L
Pump_Leak_to_Environment	N	L	N	N	N	N	N	N
Pump_Cavitation	L	H	L	L	L	L	L	L
Pump_Shift_Broken	L	L	N	N	N	N	N	N
Pump_Motor_Failed	L	H	N	N	N	N	N	N

† States of variables not shown were uniquely determined for all root causes

‡ Root causes not shown produced unambiguous effects for all variables

Table 7.2 Resolution of Ambiguity During PM Construction (continued)

	Product_Flowrate	Concentration_B	Concentration_A	Reactor_Temperature	Cooling_Water_Flowrate	CW_Flow_Controller_Signal	Cooling_Water_Setpoint
Pump_Motor_Speed_High	H	L	N	?	?	?	?
Pump_Motor_Speed_Low	L	H	N	?	?	?	?
Reactor_Effluent_Sink_Pressure_High	N	L	H	?	?	?	?
Reactor_Effluent_Sink_Pressure_Low	N	L	N	?	?	?	?
Valve_1_Blockage	N	H	N	?	?	?	?
Valve_1_Leak_to_Environment	N	N	L	?	?	?	?
Valve_1_Stuck_Closed	N	N	L	?	?	?	?
Valve_1_Stuck_Open	N	N	H	?	?	?	?
Level_Controller_Failed_High	N	L	L	?	?	?	?
Level_Controller_Biased_High	N	N	N	?	?	?	?
Level_Controller_Biased_Low	N	N	N	?	?	?	?
Level_Controller_Failed_Low	N	N	H	?	?	?	?
Level_Controller_Biased_Low	N	N	H	?	?	?	?
Level_Controller_Setpoint_High	N	N	H	?	?	?	?
Level_Controller_Setpoint_Low	N	N	N	?	?	?	?
Jacket_Feed_Source_Pressure_High	N	N	L	?	?	?	?
Jacket_Feed_Source_Pressure_Low	N	N	L	?	?	?	?
Jacket_Feed_Source_Temperature_High	N	N	N	?	?	?	?
Jacket_Feed_Source_Temperature_Low	N	N	N	?	?	?	?
Valve_2_Blockage	N	N	N	?	?	?	?
Valve_2_Leak_to_Environment	N	N	N	?	?	?	?
Valve_2_Stuck_Open	N	N	N	?	?	?	?
Valve_2_Stuck_Closed	N	N	N	?	?	?	?
Valve_2_Fire	N	H	N	?	?	?	?

Table 7.2 Resolution of Ambiguity During PM Construction (continued)

	Product_Flowrate	Concentration_B	Concentration_A	Reactor_Temperature	Cooling_Water_Flowrate	CW_Flow_Controller_Signal	Cooling_Water_Signal
Valve_2_Loss_of_Insulation	N	L	L	N	N	N	N
CW_Flow_Controller_Failed_High	N	L	L	N	N	N	N
CW_Flow_Controller_Biased_High	N	L	L	N	N	N	N
CW_Flow_Controller_Failed_Low	N	H	H	N	N	N	N
CW_Flow_Controller_Biased_Low	N	H	H	N	N	N	N
Temperature_Controller_Failed_High	N	L	L	N	N	N	N
Temperature_Controller_Biased_High	N	L	L	N	N	N	N
Temperature_Controller_Failed_Low	N	H	H	N	N	N	N
Temperature_Controller_Biased_Low	N	H	H	N	N	N	N
Temperature_Controller_Sequencer_High	N	H	H	N	N	N	N
Temperature_Controller_Sequencer_Low	N	L	L	N	N	N	N
CW_Flow_Sensor_Failed_High	N	H	H	N	N	N	N
CW_Flow_Sensor_Biased_High	N	H	H	N	N	N	N
CW_Flow_Sensor_Failed_Low	N	L	L	N	N	N	N
CW_Flow_Sensor_Biased_Low	N	L	L	N	N	N	N
Reactor_Level_Sensor_Failed_High	N	H	H	N	N	N	N
Reactor_Level_Sensor_Biased_High	N	H	H	N	N	N	N
Reactor_Level_Sensor_Failed_Low	N	L	L	N	N	N	N
Reactor_Level_Sensor_Biased_Low	N	L	L	N	N	N	N
Reactor_Temperature_Sensor_Failed_High	N	H	H	N	N	N	N
Reactor_Temperature_Sensor_Biased_High	N	H	H	N	N	N	N
Reactor_Temperature_Sensor_Failed_Low	N	L	L	N	N	N	N
Reactor_Temperature_Sensor_Biased_Low	N	L	L	N	N	N	N

The ambiguities were resolved by reasoning using approximate magnitudes of opposing causal effects. For example, two paths exist in the ESDG from a blockage at the outlet of the reactor to the product flowrate indicator. The blockage tends to cause a decrease in the product flowrate through a direct causal effect. Opposing this effect, is an indirect effect through the level control system which tends to cause an increase in the product flowrate, resulting in possible non-monotonic behavior. Assuming the controller is well tuned, the direct effect has a larger magnitude, and the product flowrate does not attain the "high" state during the transient. The ambiguities that were resolved were later confirmed by the process simulation. In general, process-specific information could be used to resolve more of the ambiguous entries.

In addition to 107 potential root causes and 14 measured variables in the PM, the qualitative relationships are expressed as 322 compiled causal links, of which 58 are activatable. 8 events, 2 associated with status transitions and 6 with state transitions are defined for each measured variable. The qualitative relationships were created in 7 hours (2 hours was used for interactive input to resolve ambiguities and 5 hours used in the subsequent analysis by the program).

#### 7.2.2.2. ADDING CONSTRAINTS TO THE PROCESS MODEL

After creating the qualitative process model, constraints and tests were added to the model using the procedures described in Section 6.7. Although not used for diagnosis, relationships between outcomes of 10 tests and the root causes were specified. In this section, we describe relationships between process constraints and root causes potentially affecting the process.

Four (4) constraint equations, determined from the available instrumentation were added to the PM. The constraints were an overall reactor material balance, cooling water pressure drop equation, reactor effluent pressure drop equation and a reactor chemical species balance. The parameters used in the pressure drop constraints identical to those used in the simulation model. The constraints are formulated as shown in eqs. (7.3) through (7.6), below.

*Inventory Constraint:*

$$(L - L_{t=0}) \cdot A - \int_0^t (F_{in} - F_{out}) \cdot dt = R1 \quad (7.3)$$

*Cooling Water Pressure Drop Constraint:*

$$F_{cw} - P_{cw}^{0.5}/(R_{sjacket} + 5.0 \cdot \exp(0.0545 \cdot V_F)) = R2 \quad (7.4)$$

*Effluent Pressure Drop Constraint:*

$$F_{out} - (\rho Lg + \Delta P_{pump})^{0.5}/(R_{seffluent} + 5.0 \cdot \exp(0.0545 \cdot V_L)) = R3 \quad (7.5)$$

*Mole Balance Constraint:*

$$(C_A + C_B + C_{C,nominal}) \cdot L \cdot A - (C_{A,t=0} + C_{B,t=0} + C_{C,t=0}) \cdot L_{t=0} \cdot A \\ + \int_0^t (F_{out} \cdot (C_A + C_B + C_{C,nominal}) - F_{in} \cdot C_{A,feed}) \cdot dt = R4 \quad (7.6)$$

In these equations, the constraint residuals, R1 through R4 are nominally zero.  $F_{in}$  and  $F_{out}$  are the reactor feed and product flow rates, respectively.  $C_A$ ,  $C_B$  and  $C_C$  are concentrations of chemical species in the reactor, while  $L$  and  $A$  are the reactor liquid level and reactor cross sectional area.  $V_L$  and  $V_F$  are output signals of the level and cooling water flow controllers, and  $R_s$  is the nominal resistance to fluid flow in the indicated flow path.

Qualitative relationships between the potential root causes and constraints are shown in Table 7.3. In the table N/A indicates that the constraint is not affected by the particular root cause. Initial state transitions of the constraints for each of the root causes is shown. None of the constraints display non-monotonic dynamic behavior in response to the root causes.

Table 7.3 Relationships Between Root Causes and Constraints

	Inventory	CW_Pressure_Drop	Efl_Pressure_Drop	Mol_Balance
Reactor_Feed_Source_Concentration_A_High	N/A	N/A	N/A	Low
Reactor_Feed_Source_Concentration_A_Low	N/A	N/A	N/A	High
Reactor_Feed_Source_Pressure_High	N/A	N/A	N/A	High
Reactor_Feed_Source_Pressure_Low	N/A	N/A	N/A	Low
Reactor_Feed_Source_Temperature_High	N/A	N/A	N/A	Low
Reactor_Feed_Source_Temperature_Low	N/A	N/A	N/A	High
Jacket_Effluent_Sink_Pressure_High	N/A	Low	N/A	High
Jacket_Effluent_Sink_Pressure_Low	N/A	High	N/A	Low
CSTR_Inlet_Blockage	N/A	N/A	N/A	Low
CSTR_Outlet_Blockage	N/A	N/A	Low	Low
CSTR_Leak_to_Environment	Low	N/A	Low	Low
CSTR_Fire	N/A	N/A	N/A	Low
CSTR_Jacket_Fouling	N/A	N/A	N/A	Low
CSTR_Loss_of_Insulation	N/A	N/A	N/A	High
CSTR_Loss_of_Mixing	N/A	N/A	N/A	High
CSTR_Catalyst_1_Deactivation	N/A	N/A	N/A	Low
CSTR_Catalyst_2_Deactivation	N/A	N/A	N/A	High
CSTR_Jacket_Leak_to_Reactor	High	High	N/A	High
CSTR_Jacket_Leak_to_Environment	N/A	High	N/A	High
Pump_Blockage	N/A	N/A	Low	Low
Pump_Leak_to_Environment	Low	N/A	Low	Low
Pump_Cavitation	N/A	N/A	Low	Low
Pump_Shift_Broken	N/A	N/A	Low	Low
Pump_Motor_Failed	N/A	N/A	Low	Low
Pump_Motor_Speed_High	N/A	N/A	High	High
Pump_Motor_Speed_Low	N/A	N/A	Low	Low
Reactor_Effluent_Sink_Pressure_High	N/A	N/A	Low	Low
Reactor_Effluent_Sink_Pressure_Low	N/A	N/A	High	High
Valve_1_Blockage	N/A	N/A	Low	Low
Valve_1_Leak_to_Environment	Low	N/A	Low	Low
Valve_1_Stuck_Closed	N/A	N/A	Low	Low
Valve_1_Stuck_Open	N/A	N/A	High	High
Level_Controller_Failed_High	N/A	N/A	N/A	High
Level_Controller_Biased_High	N/A	N/A	N/A	High
Level_Controller_Failed_Low	N/A	N/A	N/A	Low
Level_Controller_Biased_Low	N/A	N/A	N/A	Low

Table 7.3 Relationships Between Root Causes and Constraints (continued)

	Inventory	CW_Pressure_Drop	Eff_Pressure_Drop	Mol_Balance
Level_Controller_Setpoint_High	N/A	N/A	N/A	Low
Level_Controller_Setpoint_Low	N/A	N/A	N/A	High
Jacket_Feed_Source_Pressure_High	N/A	N/A	N/A	High
Jacket_Feed_Source_Pressure_Low	N/A	N/A	N/A	Low
Jacket_Feed_Source_Temperature_High	N/A	N/A	N/A	Low
Jacket_Feed_Source_Temperature_Low	N/A	N/A	N/A	High
Valve_2_Blockage	N/A	Low	N/A	Low
Valve_2_Leak_to_Environment	N/A	High	N/A	Low
Valve_2_Stuck_Open	N/A	High	N/A	High
Valve_2_Stuck_Closed	N/A	Low	N/A	Low
Valve_2_Fire	N/A	N/A	N/A	Low
Valve_2_Loss_of_Insulation	N/A	N/A	N/A	High
CW_Flow_Controller_Failed_High	N/A	N/A	N/A	High
CW_Flow_Controller_Biased_High	N/A	N/A	N/A	High
CW_Flow_Controller_Failed_Low	N/A	N/A	N/A	Low
CW_Flow_Controller_Biased_Low	N/A	N/A	N/A	Low
Temperature_Controller_Failed_High	N/A	N/A	N/A	High
Temperature_Controller_Biased_High	N/A	N/A	N/A	High
Temperature_Controller_Failed_Low	N/A	N/A	N/A	Low
Temperature_Controller_Biased_Low	N/A	N/A	N/A	Low
Temperature_Controller_Setpoint_High	N/A	N/A	N/A	Low
Temperature_Controller_Setpoint_Low	N/A	N/A	N/A	High
CW_Flow_Sensor_Failed_High	N/A	High	N/A	Low
CW_Flow_Sensor_Biased_High	N/A	High	N/A	Low
CW_Flow_Sensor_Failed_Low	N/A	Low	N/A	High
CW_Flow_Sensor_Biased_Low	N/A	Low	N/A	High
Reactor_Level_Sensor_Failed_High	High	N/A	Low	High
Reactor_Level_Sensor_Biased_High	High	N/A	Low	High
Reactor_Level_Sensor_Failed_Low	Low	N/A	High	Low
Reactor_Level_Sensor_Biased_Low	Low	N/A	High	Low
Reactor_Temperature_Sensor_Failed_High	N/A	N/A	N/A	High
Reactor_Temperature_Sensor_Biased_High	N/A	N/A	N/A	High
Reactor_Temperature_Sensor_Failed_Low	N/A	N/A	N/A	Low
Reactor_Temperature_Sensor_Biased_Low	N/A	N/A	N/A	Low

Table 7.3 Relationships Between Root Causes and Constraints (continued)

	Inventory	CW_Pressure_Drop	Efl_Pressure_Drop	Mol_Balance
Feed_Flow_Sensor_Failed_High	Low	N/A	N/A	Low
Feed_Flow_Sensor_Biased_High	Low	N/A	N/A	Low
Feed_Flow_Sensor_Failed_Low	High	N/A	N/A	High
Feed_Flow_Sensor_Biased_Low	High	N/A	N/A	High
Feed_Concentration_Sensor_Failed_High	N/A	N/A	N/A	Low
Feed_Concentration_Sensor_Biased_High	N/A	N/A	N/A	Low
Feed_Concentration_Sensor_Failed_Low	N/A	N/A	N/A	High
Feed_Concentration_Sensor_Biased_Low	N/A	N/A	N/A	High
Product_Flow_Sensor_Failed_High	High	N/A	High	High
Product_Flow_Sensor_Biased_High	High	N/A	High	High
Product_Flow_Sensor_Failed_Low	Low	N/A	Low	Low
Product_Flow_Sensor_Biased_Low	Low	N/A	Low	Low
CW_Pressure_Sensor_Failed_High	N/A	Low	N/A	N/A
CW_Pressure_Sensor_Biased_High	N/A	Low	N/A	N/A
CW_Pressure_Sensor_Failed_Low	N/A	Low	N/A	N/A
CW_Pressure_Sensor_Biased_Low	N/A	Low	N/A	N/A
Concentration_A_Sensor_Failed_High	N/A	N/A	N/A	High
Concentration_A_Sensor_Biased_High	N/A	N/A	N/A	High
Concentration_A_Sensor_Failed_Low	N/A	N/A	N/A	Low
Concentration_A_Sensor_Biased_Low	N/A	N/A	N/A	Low
Concentration_B_Sensor_Failed_High	N/A	N/A	N/A	High
Concentration_B_Sensor_Biased_High	N/A	N/A	N/A	High
Concentration_B_Sensor_Failed_Low	N/A	N/A	N/A	Low
Concentration_B_Sensor_Biased_Low	N/A	N/A	N/A	Low

### 7.2.2.3. THE PROCESS MODEL

After supplementing the qualitative model with constraints, prior probabilities of 8 of the potential root causes were set to 0.0. In effect, this removed these root causes from the set that potentially affects the process. The potential root causes removed were those that produced qualitatively identical symptoms as other root causes affecting the process and whose effects were not simulated.<sup>6</sup> The prior probabilities of all other root causes were specified as 1.0. Nominal parameter values for sensor and constraint monitors were set by the method described in Section 7.3.2.

The final PM for the jacketed reactor process consisted of

- 99 Potential-Root-Cause instances,
- 14 Measured-Variable instances,
- 4 Measured-Constraint instances,
- 14 Sensor-Monitor instances,
- 4 Constraint-Monitor instances,
- 144 Potential-Event instances,
- 20 Test instances, and
- 334 Compiled-Causal-Link instances (58 activatable links).

---

<sup>6</sup> The simulation program for the jacketed reactor process is described in Section 7.3.1.

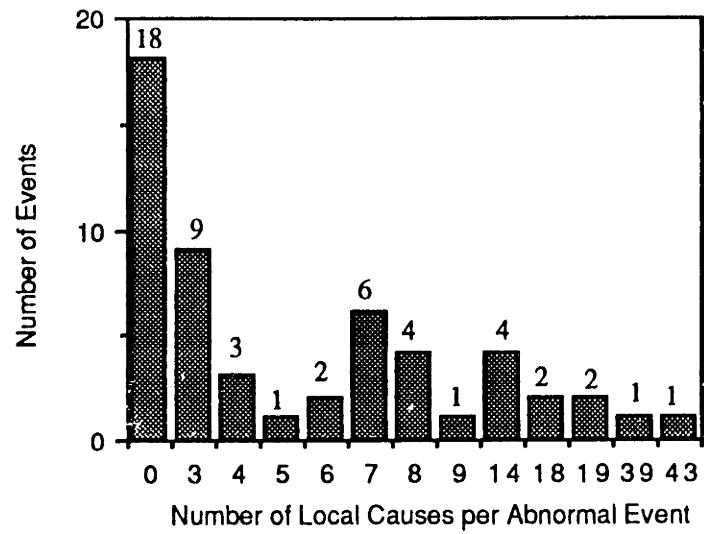


Fig. 7.2a. Distribution of Local Causes

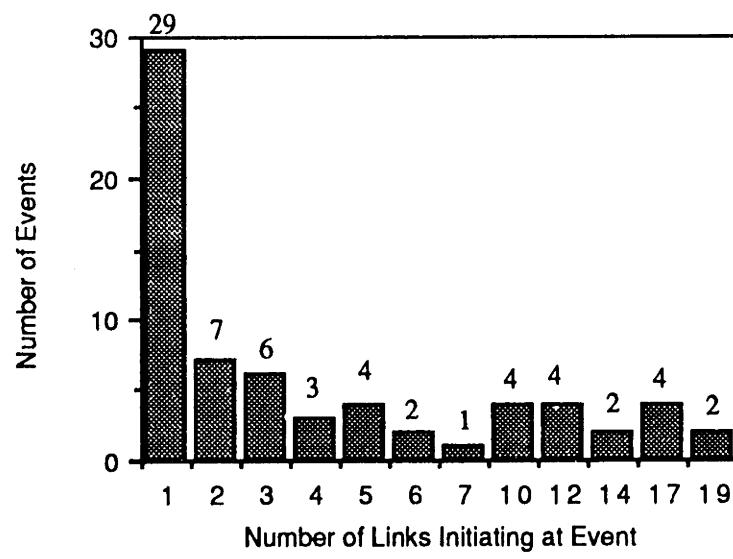


Fig. 7.2b. Distribution of Compiled Causal Links

Examples of object instances for the PM of the process are listed in Appendix K. A distribution of local causes for event transitions from the normal state<sup>7</sup> (i.e. abnormal events) is shown in Fig. 7.2a. The average number of local causes per event is 9.75.<sup>8</sup> Figure 7.2b shows a distribution of the number of compiled causal links initiating at each event. On average, 4.91 links initiate at each event.

## 7.3. Case Study

The process model was verified by evaluating the diagnostic performance of MIDAS using numerical data from a simulation of the process as input. In this section, the steps carried out in performing the case study are discussed. First, we describe the numerical simulation of the process.

### 7.3.1. Process Simulation

A FORTRAN 77 dynamic simulation program for the jacketed continuous stirred tank reactor process was developed by Finch [2]. The program simulates the effects of 106 malfunction modes on the process and produces output files containing the simulated sensor data. Constraint residuals are calculated by the program and their values are also recorded in output data files. These sensor and constraint values may be written to the data files with variable frequencies and in a random order. In effect, this simulates differences in sampling frequencies and the random order of event detection that might be expected in realistic process environments.

Noisy process data is simulated by adding random errors to simulated measurement values. The noisy data is used in all subsequent controller calculations.

The malfunction modes that may be simulated include biases and fixed failures in sensors and controllers, process malfunctions and external process disturbances. For each of these malfunctions, the magnitude, rapidity (indicated by a time constant) and the time of

---

<sup>7</sup> Events indicating a transition from the normal state are referred to as abnormal events.

<sup>8</sup> This value is based on abnormal events that are associated with local causes.

occurrence can be specified. Multiple malfunctions can also be simulated, the malfunctions occurring simultaneously or at different times.

### 7.3.2. Diagnosis of Simulated Malfunctions

In the following, steps carried out in diagnosing root causes from files containing abnormal process data are outlined.

Monitors translate numerical process data into discrete qualitative events. First, the sensor and constraint monitors in the PM were tuned. In tuning the monitors, numerical values of parameters associated with the variable or constraint were determined. These parameters included nominal steady state values, thresholds and standard deviations. The values were set using malfunction-free data generated by the simulation program using the **Set Monitor Thresholds** Utility of MIDAS [1].

Next, process data for a randomly selected set of 100 malfunctions was generated. The extents and rapidity for each malfunction were also assigned randomly. The selected malfunctions are presented in [2]. Of the 100 malfunctions, 23 could not be used to evaluate MIDAS' performance either because they were beyond the scope of the simulation, their extents were too small to produce observable events or the malfunctions were unobservable given the process instrumentation.

Data from the remaining 77 simulation runs formed the basis for the case study. Of the 77 malfunctions, 34 were sensor or controller failures with the following distribution:

- 13 biases,
- 11 failures to fixed values that were out of range,
- 10 failures to fixed values within range.

Data output for each run was for 90 simulated minutes, with the malfunction introduced after 15 minutes of normal operation.

The data for each simulation run was input to MIDAS using its monitors. The MIDAS settings used for diagnosis during the case study were:

Maximum Precursor Search Depth	-	2
Maximum Successor Search Depth	-	2
Interrogation Default Setting	-	NONE
Interrupt Default Setting	-	SAVE
Level of Evidence Evaluation	-	VARIABLE
Maximum Event Processing Time	-	10
Minimum Event Processing Time	-	2
Type of Likelihood Displayed	-	CONDITIONAL-PROBABILITY
Probabilistic Likelihood Display Threshold	-	0.0

The data produced RECORDED-EVENT sequences which were diagnosed using the MIDAS' inference algorithms developed by Finch [2]. After each recorded event the current state of the diagnostic hypothesis was recorded. The current state of the diagnosis includes:

- A list of INFERRED-MALFUNCTION clusters, each cluster representing an independent malfunction.
- A list of HYPOTHESIZED-ROOT-CAUSE candidates for each cluster. Associated with each candidate in the list is its relative ranking.

## 7.4. Case Study Results and Analysis

In this section, we present the results of the case study. In summary, (i) an accurate diagnosis was achieved in 93% of the cases, based on the diagnosis after the initial event, (ii) an accurate diagnosis was achieved in 100% of the cases, at the end of the transient, and (iii) on average, 96% of the malfunctions in the process were eliminated by the end of the transient. Cases in which an inaccurate diagnosis was obtained were due to inadequacies in the event detection scheme. These results indicate very good diagnostic performance and confirm that the PM of the jacketed reactor process is correct. A detailed analysis of the case study results follows.

### 7.4.1. Distribution of Events

The number of recorded events detected from each run of simulated data varied. Several cases produced only 1 event while there was a case where 35 events were detected. Figure 7.3 shows a distribution of the number of events produced for the runs. On average, 6.4 events were detected for each run of data. Cases in which 11 or more events were detected represent a small sample, and reliable statistical conclusions cannot be inferred from these cases. In the rest of this section, most of the results presented represent the performance for cases in which 10 or less events were detected.

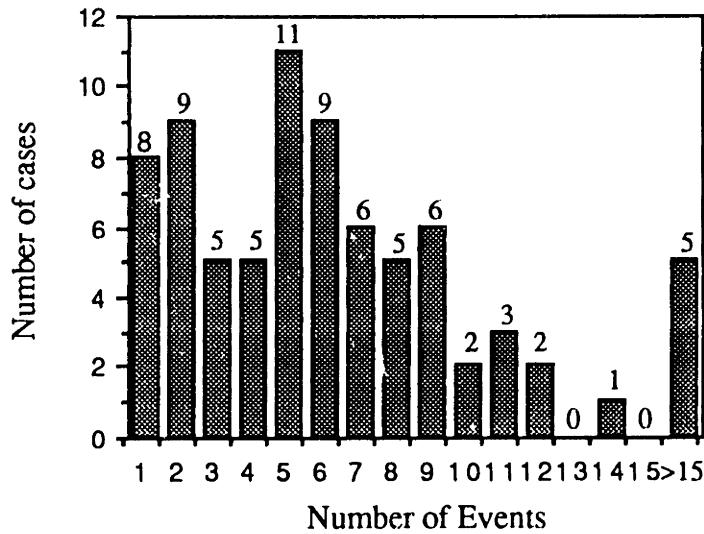


Fig. 7.3. Distribution of Events

### 7.4.2. Non Monotonic Dynamic Behavior

One of the objectives of the techniques for developing the PM was to create PMs in which explicit knowledge about non-monotonic dynamic behavior could be represented and utilized as sources of diagnostic information. Knowledge about these kinds of behavior should be represented so that an accurate diagnosis with a high degree of resolution is produced. At least one measured variable displayed compensatory response in 31 of the runs, representing 40% of the total number of runs. Inverse response was exhibited by at least one variable in 16 (21%) of the cases. The significant number of cases for which

non-monotonic behavior was exhibited provides a basis for verifying the process model and evaluating diagnostic performance in a dynamic process environment.

### 7.4.3. Diagnostic Performance

In this section, we discuss the overall performance of MIDAS for the cases run. We begin by defining the criteria used in evaluating diagnostic performance.

By far the two most important criteria for evaluating the diagnostic performance of a diagnostic aid are the **accuracy** of the diagnosis and the degree of diagnostic **resolution**. Accuracy refers to the ability to produce a diagnostic candidate set that includes the true malfunction origin. Producing an accurate diagnosis is of primary importance because an inaccurate diagnosis diverts the attention of the operator away from the true malfunction origin. Formally, if the true malfunction is contained in the candidate set, the accuracy, AC, of the diagnosis is 1. Otherwise, AC is 0.

Resolution refers to the size of the candidate set. If the candidate set contains many members, the degree of diagnostic resolution is low. Conversely, a set with few members constitutes a diagnosis with a high degree of resolution. Producing a diagnosis with a low degree of resolution does not provide much help to the operator in discriminating among the possible malfunction sources.

A candidate set produced by MIDAS consists of a list of root causes, each with its relative ranking. With these rankings, the candidates can be partitioned into tiers. Within each tier the relative ranking of the root causes are equal. We define the resolution, RE, in terms of the number of root causes, Nr that have a relative ranking equal to or greater than the relative ranking of the true malfunction.

$$RE = (Nr - 1) / (Nt - 1) \quad (7.7)$$

In eq. (7.7), Nt is the total number of malfunctions that may affect the process. Nr corresponds to the maximum number of malfunctions that would be examined by the operator before identifying the true malfunction, if the candidate root causes were examined according to their relative rankings. When the true malfunction is the only malfunction with the highest ranking in the candidate set, Nr is 1 and RE is 1. When all malfunctions in

the process are in the candidate set and the true fault has the lowest ranking, Nr is Nt and RE is 0.

A performance parameter,  $\Phi$ , is defined as the product of the accuracy and resolution.

$$\Phi = AC \cdot RE \quad (7.8)$$

A value of 1 indicates a perfect diagnosis, while 0 indicates a useless diagnosis. A useless diagnosis is obtained either when the diagnosis is inaccurate or the candidate set contains all malfunctions in the process.

Table 7.4. Average Diagnostic Performance of MIDAS

Events Observed	Accuracy (% of Cases) AC	Ranked in 1st or 2nd Tier (% of Cases)	Average Resolution RE	Average Performance $\Phi$
1	93	97	0.892	0.87
2	100	97	0.950	0.93
3	100	100	0.960	0.94
4	100	98	0.964	0.96
5	100	98	0.968	0.96
6	100	92	0.966	0.94
7	97	90	0.960	0.93
8	100	80	0.954	0.92
9	100	84	0.944	0.92
10	100	76	0.932	0.91

Table 7.4 shows the average diagnostic performance of MIDAS. The entries in each row are based on the number of cases in which at least the indicated number of events is observed. For example, only 1 event was observed in 8 cases, and 2 events were observed in an additional 9 cases. Therefore, the performance criteria for the first three rows are based on 77, 69 and 60 cases, respectively.

From the table, it can be seen that MIDAS produces an accurate diagnosis most of the time. There were 6 cases in which an inaccurate diagnosis was obtained; 5 cases after detection of the 1st event and 1 case, after the 7th event was detected. The inaccurate diagnosis obtained in these cases was not due to errors in the PM, but due to inadequacies in the event detection scheme. In all 5 cases where an inaccurate diagnosis was produced after the 1st event, the observed event did not occur in its expected causal order.<sup>9</sup> The case where an inaccurate diagnosis was obtained after the 7th event was a situation where apparent inverse response was exhibited by a variable in a control loop. A possible solution to this problem is to use non-stationary threshold values as discussed in Section 3.6.

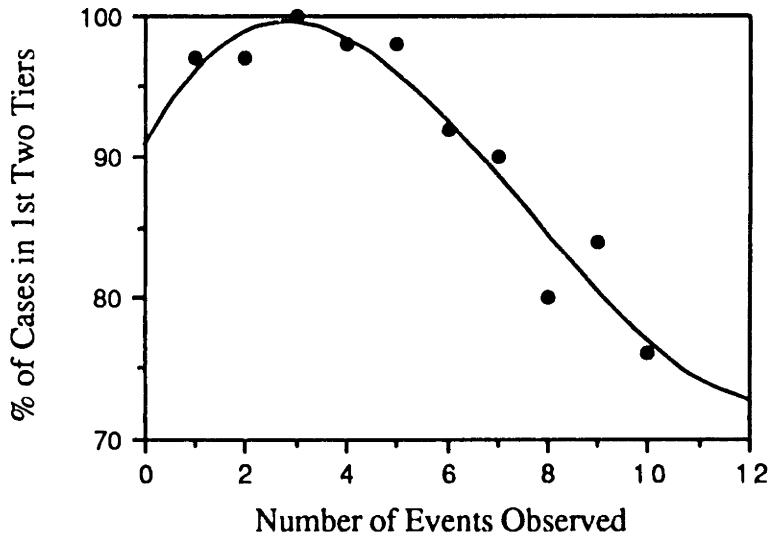


Fig. 7.4. Tier of True Malfunction

---

<sup>9</sup> The robustness features of MIDAS' inference [2] did not result in an accurate diagnosis because a sub-threshold event was not detected for the first causally expected event. A remedy is to retune the affected monitors so that the sub-threshold values are closer to the nominal value.

Figure 7.4 shows how tier ranking varies with the number of observed events. The relative rankings of faults in the first two tiers were generally within an order of magnitude, therefore the percentage of cases in which the true malfunction was ranked in the first two tiers is shown. MIDAS listed the true malfunction in the 1st or 2nd tier in over 90% of the cases. Even when the malfunction was not in the top two tiers, average resolution and average performance were good. Figures 7.5 and 7.6 show how average resolution and average performance vary with the number of observed events. Average resolution and average performance remained above 0.90 for cases in which the malfunction was not in the top two tiers.

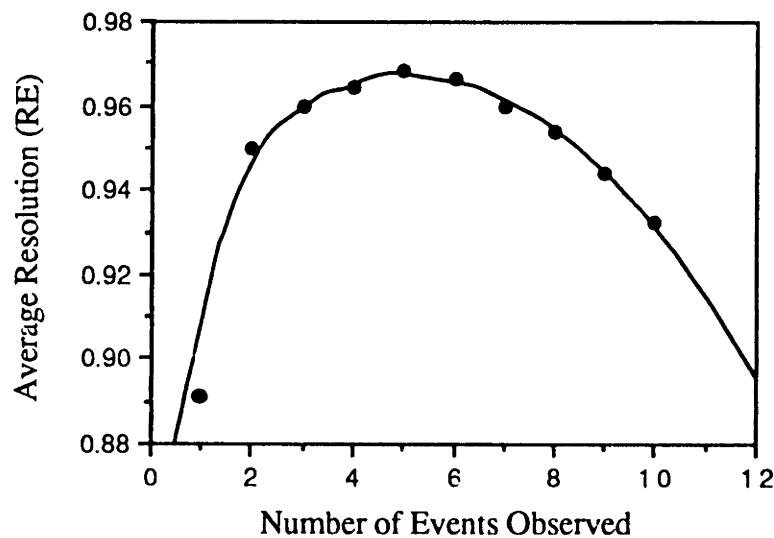


Fig. 7.5. Average Resolution

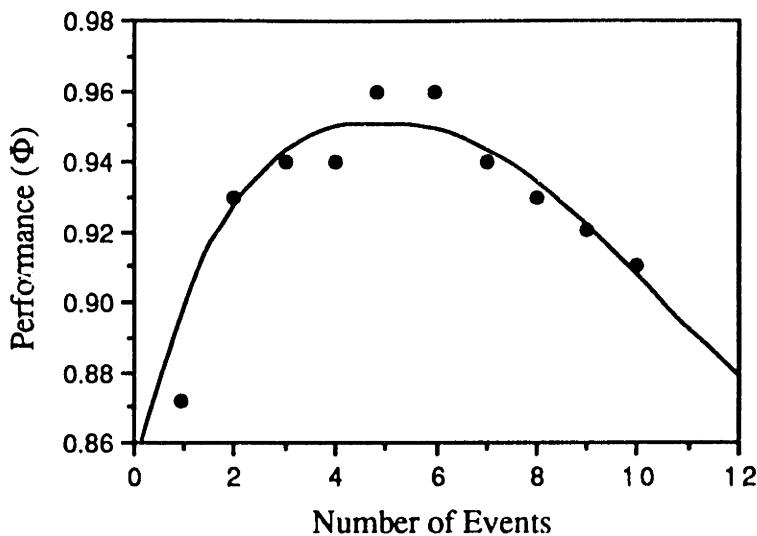


Fig. 7.6. Average Performance

There was a noticeable drop in performance for cases where more than 6 events were observed. This drop in performance is primarily due to the formation of "active"<sup>10</sup> causal loops in inferred malfunction clusters [2] and the inability of the inference algorithm to identify source events in the loop. In general, these causal loops arise from negative feedback (control and non-control) or positive feedback loops in the process. Finch [2], suggests that this problem may be solved by widening thresholds of monitors, resolving ambiguities during model creation or by incorporating semi-quantitative knowledge in the links. Resolving ambiguities during model creation does not break the loop although it may lead to improved resolution. One possible remedy not suggested in [2], is to restrict source events to be among the first event observed for each variable in the process. This "breaks" active causal loops arising from negative feedback loops, identifying a single source event. Breaking causal loops in this manner can be accomplished by modifying the inference algorithm. This would have resolved the temperature control causal loop that led to the drop in performance of MIDAS.

Figure 7.7 shows a distribution of the resolution achieved at the end of the transient. Based on all 77 cases, the average resolution was 0.97. This corresponds to an average candidate set size of 4 (i.e. 96% of the process malfunctions eliminated). Perfect resolution corresponds to a candidate set size of 1. Given the instrumentation scheme some

---

<sup>10</sup> A causal loop was activated when a variable in the temperature control loop exhibited apparent inverse response.

malfunctions are qualitatively identical<sup>11</sup> and perfect resolution cannot always be achieved. In most cases where resolution was greater than or equal to 0.97, the best diagnosis given the available instrumentation was produced. The sub-optimal performance is primarily due to the formation of causal loops and to fundamental limitations of qualitative modeling. In Section 7.5, we address these limitations.

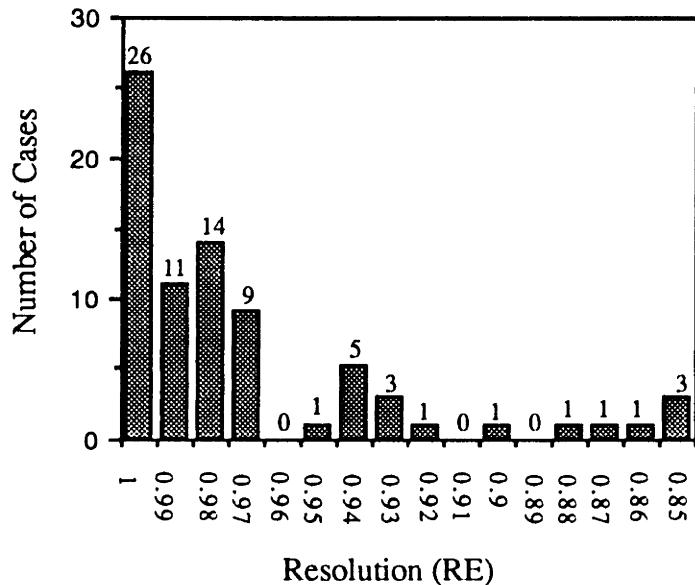


Fig. 7.7. Resolution at End of Transient

In the final analysis, an accurate diagnosis was produced using the PM in most situations. The few cases where an inaccurate diagnosis was obtained was not due to errors in the PM, but due to limitations in the detection scheme of MIDAS. Furthermore, the performance can be improved by modifying MIDAS inference to break causal loops as suggested. Although perfect performance was not achieved, the level of performance achieved verifies the technique for developing the PM.

---

<sup>11</sup> For example pump blockage and CSTR outlet blockage.

## 7.5. Limitations of Qualitative Modeling

In this section, we address some of the fundamental limitations of diagnostic systems based on qualitative models. In MIDAS, the primary limitations derive from the:

- Inherent ambiguity in qualitative modeling.
- Point values and semi-infinite intervals associated with qualitative variables and relationships.

The primary effect of the ambiguity in qualitative modeling is the existence of causal links initiating from the same event and terminating at events whose consequent states indicate qualitatively opposite effects. For example in the hypothetical PM shown in Fig. 7.8, events **BS-normal-to-high** and **BS-normal-to-low** are successors of **AS-normal-to-high**.

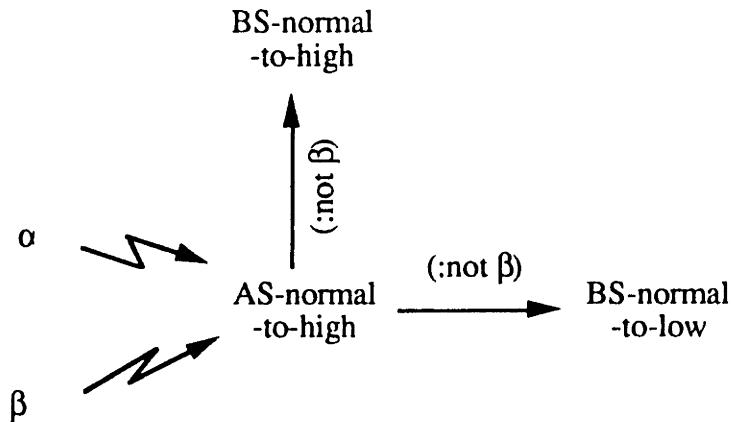


Fig. 7.8. PM of a Hypothetical Process

This ambiguity results in a highly connected PM and consequently an increase in the time required for diagnosis by MIDAS. A detailed discussion of the effect of model connectivity on diagnostic speed appears in [2].

The major consequence of the ambiguity is a loss in diagnostic resolution, since each malfunction could lead a larger number of event sequences.<sup>12</sup> Qualitative relationships in the PM are derived so that the ambiguity is minimized while ensuring that the relationships account for the behavior of all qualitatively similar processes.<sup>13</sup> However, this ambiguity may be resolved further by the use of specific quantitative information for the process that is to be diagnosed. The not-condition associated with the link between **AS-normal-to-high** and **BS-normal-to-low** implies that if the link is invoked during diagnosis,  $\beta$  may be removed from the candidate set. If it is known that the event **BS-normal-to-low** cannot occur as a result of malfunction,  $\alpha$ ,  $\alpha$  is added to the set of malfunctions referred to by the not-condition.<sup>14</sup> Resolving ambiguity requires information about relative quantitative effects for the particular process and potentially improves diagnostic resolution.

The algorithm for deriving qualitative relationships in the PM has a facility by which ambiguities may be resolved by the user. Using this facility, 48.6% of the potential ambiguities were resolved while creating the PM of the jacketed reactor process. The diagnostic resolution attained by MIDAS could have been improved if a greater percentage of the ambiguous entries in Table 7.2 were resolved.

Another limitation of qualitative modeling arises from the point values and semi-infinite intervals associated with variables in qualitative algebra. The consequence of this is that links in the PM express **may-cause** relationships rather than **will-cause** relationships. Thus a successor link can be interpreted as follows; the occurrence of the successor event may occur as a result of the precursor event. It does not imply that the successor event will necessarily occur at any finite time after the precursor event occurs.

This asymmetry limits the type of information that can be used in performing inference with the PM. The occurrence of an event may be used as a positive source of diagnostic information. However, no information is deduced from the non occurrence of an expected event. For example, in Fig. 7.8, if **BS-normal-to-high** is observed after **AS-normal-to-high**,  $\beta$  may be eliminated from the diagnostic candidate set. On the other hand, if

---

<sup>12</sup> Thus, an event sequence can be explained by a greater number of malfunctions.

<sup>13</sup> The reduction of ambiguity arising from negative feedback is discussed in Chapter 3.

<sup>14</sup> Alternatively, if  $\alpha$  and  $\beta$  are the only root-causes that may result in the event **AS-normal-to-high**, the link may be removed, thus, reducing the connectivity of the PM.

**BS-normal-to-high** is not observed, even after a very long time, no statement can be made about the relative likelihood of  $\alpha$  and  $\beta$ . While humans are capable of using both occurrences and non-occurrences of events in diagnostic inference, MIDAS is limited to using only event occurrences.

In order to express **will-cause** relationships in the PM, knowledge of approximate time delays associated with the causal links would be required. Thus, if a subsequent event does not occur after a sufficient amount of time has elapsed, the non occurrence of the event may be used as a source of diagnostic information. Because the time delay associated with links will vary, depending on threshold values associated with event transitions, information about the strengths (gains) of effects attributed to the links would also be required.

It may be possible to derive will-cause links using **semi-quantitative** algebraic approaches. Techniques based on the order-of-magnitude reasoning formalism [3] and interval arithmetic [4] may provide promising areas of research.

## 7.6. Conclusions

In this chapter, the results of a case study detailing the diagnostic performance of MIDAS for a jacketed reactor process was presented. The diagnostic model of the process was developed using the algorithmic procedures described in Chapter 6. The results of the study indicate that (i) an accurate diagnosis was achieved in 93% of the cases, based on the diagnosis after the initial event, (ii) an accurate diagnosis was achieved in 100% of the cases, at the end of the transient, and (iii) on average, 96% of the malfunctions in the process were eliminated by the end of the transient. In a significant number of cases, non-monotonic dynamic behavior was exhibited by variables in the process.

The acceptable level of performance achieved verifies the PM as a diagnostic model that could be used in a dynamic process environment. The level of performance also validates the techniques for developing the PM. Methods for improving diagnostic performance in MIDAS by enhancing the qualitative models were suggested.

## 7.7. Notation for Chapter 7

A	reactor cross sectional area
A,B,C	chemical species
C	chemical species concentration
F	volumetric flowrate
g	gravitational constant
k	reaction rate constant
L	reactor level
P	pressure
r	reaction rate
R	constraint residual
Rs	flow resistance
t	time
T	temperature
V <sub>L</sub> , V <sub>F</sub>	controller output signals

### Special symbols

$\Delta P$	pressure head
$\rho$	density

### Subscripts

cw	cooling water
in	reactor inlet
out	reactor outlet

## 7.8. References for Chapter 7

1. Kramer, M.A., F.E. Finch and O.O. Oyeleye (1989). Operating Manual and User's Guide for MIDAS (version 1.0a), Laboratory for Intelligent Systems in Process Engineering, Massachusetts Institute of Technology.

2. Finch, F.E. (1989). Automated Fault Diagnosis of Chemical Process Plants using Model-Based Reasoning. Sc.D. Thesis, Massachusetts Institute of Technology.
3. Mavrovouniotis, M.L. and G. Stephanopoulos (1987). Reasoning with orders of magnitude and approximate relations. Proc. 6th Natl. Conf. on A.I. (AAAI-87), 626-630.
4. Moore, R.E. (1979). Methods and Applications of Interval Analysis SIAM.

## 8. Conclusions and Recommendations

This thesis has addressed the problem of qualitative modeling and its applications in the automated diagnosis of process malfunctions in continuous chemical processes. Although emphasis has been placed on the modeling of continuous chemical processes, the results and concepts presented in this thesis are applicable to continuous dynamic processes in other domains. In this chapter, we summarize the major contributions of the research and indicate directions for future research.

Several types of modeling approaches may be employed in developing diagnostic aids. These include conventional quantitative approaches such as parameter estimation techniques, rule based expert systems and approaches based on "formal" qualitative models of process behavior. The major attractions in using qualitative models over quantitative approaches stem from the ease of model development and the ease of explanation of diagnostic conclusions. When compared to rule based expert systems, qualitative modeling provides a more structured approach for developing diagnostic knowledge bases. This results in knowledge bases that can be more easily verified for completeness and consistency.

The major factor limiting the use of qualitative models for diagnostic applications is the inherent ambiguity in qualitative modeling. The consequence of this ambiguity is that previous methods of qualitative modeling have tended to generate behaviors<sup>1</sup> that are not exhibited by any process that satisfies the qualitative description provided by the model. In an attempt to limit the production of spurious behaviors, some methods introduce assumptions that exclude actual behaviors of the process. These limitations lead to unnecessary losses of diagnostic resolution and the production of inaccurate diagnoses by diagnostic systems based on qualitative models. The major challenge in qualitative modeling is to reduce the production of spurious behaviors while guaranteeing the inclusion of all actual process behaviors.

This research has resulted in the development and implementation of a process independent, algorithmic technique for deriving the qualitative core of knowledge bases for an on-line diagnostic reasoning program<sup>2</sup> for continuous refinery and chemical process

---

<sup>1</sup>These behaviors are referred to as spurious behaviors.

<sup>2</sup>The Model Integrated Diagnostic Analysis System (MIDAS).

plants. Using the technique, knowledge bases are derived from qualitative unit models and a description of the flowsheet. A novel feature of the technique is that certain behaviors that are not apparent from local unit models, but result from global process interactions are automatically represented in the diagnostic knowledge base. Representing knowledge about these behaviors, allows an **accurate** diagnosis to be obtained in the dynamic process environment without unnecessary losses of diagnostic **resolution**. In the following, we describe in more detail the major results of this research.

## 8.1. Steady State Qualitative Modeling

In order to understand the fundamental limitations of qualitative modeling, an investigation of steady state qualitative modeling was carried out. The steady state qualitative modeling problem may be stated as that of predicting the ultimate qualitative change of a continuous process in response to process perturbations. Qualitative models of continuous processes may be formulated in terms of qualitative constraints derived from fundamental principles governing the process. A constraint satisfaction algorithm was developed to determine process behavior from qualitative models.

A novel systematic procedure for reducing the ambiguity inherent in steady state qualitative modeling was developed. It was discovered that **redundant** constraint information, some of which can only be deduced from **global** process topology is necessary to reduce this ambiguity. The need for global constraint information indicates an inherent trade-off between modularity and solution multiplicity in steady state qualitative modeling.

Specifically, ambiguity is primarily reduced by a set of **latent** qualitative constraints derived from algebraic manipulation of a basic set of quantitative process equations. A set of heuristics was suggested to guide in the search for useful local and global non-causal latent constraints. Spurious qualitative solutions corresponding to responses from unstable steady states are admitted by the set of basic and latent non-causal constraints. These solutions are eliminated by additional latent constraints derived from local and global causal considerations. Combining basic, non-causal latent and causal latent constraints theoretically eliminates all spurious behaviors (solutions) in systems where analytic steady state solutions exist.

The utility of the multiple sources of constraint information was demonstrated using a chemical reactor process with its associated heat exchange and control systems, as an example. The number of qualitative solutions obtained was reduced by up to **two orders of magnitude**, when compared to other qualitative modeling formalisms.

## 8.2. A Novel Representation of Causality for Continuous Processes

Steady state qualitative models may exclude initial and transient measurement patterns and are therefore not appropriate for diagnosis in the dynamic chemical process environment. In order to represent the dynamic qualitative behavior of continuous process systems, a novel representation of process causality, the Extended Signed Directed Graph (ESDG) was developed. The ESDG represents both the immediate local and apparent global causality in the process.

Some of the most useful representations of causality that have appeared previously are based on the concept of the single stage Signed Directed Graph (SDG). Previous work on signed directed graphs have not adequately addressed the role of feedback and complex dynamics. In order to reduce the inherent ambiguity in SDG models, assumptions which lead to the exclusion of non-monotonic behaviors, namely, inverse and compensatory responses have been introduced, previously. These behaviors arise from global interactions in negative feedback loops. For example process behavior is excluded in feedback control loops, when the effect of a disturbance is passed to the manipulated variable of a controller, without an apparent deviation of the controlled variable. Using previous approaches based on signed directed graphs, such instances of non-monotonic process behaviors cannot always be accounted for.

**Rigorous theorems** for identifying all such non-monotonic "non-local" behaviors were derived from a **global** qualitative analysis of the SDG. Assuming no transitions to other qualitative regimes,<sup>3</sup> the theorems indicate that:

- Compensatory response occurs as a result of the effect of integrators within negative feedback loops.

---

<sup>3</sup> Control loop saturation is an example of a transition to another qualitative regime.

- Inverse response may occur as a result of positive feedback effects associated with a negative feedback loop.

When controller saturation occurs, an extension of the theorems indicates that:

- Inverse response may be exhibited if loop saturation inhibits the effect of an integrator that suppresses positive feedback effects in the negative feedback loop.

The Extended Signed Directed Graph derives from the result of the global analysis of the process SDG. In order to derive the SDG of a process, SDG models of the basic process units are required. Guidelines for deriving SDG models for process units were developed. The unit models are developed so that a modular approach may be adopted, when constructing the SDG of a process from its constituent units and from information about process structure. The guidelines ensure the correct identification of non-local causalities in the ESDG.

The non-monotonic behavior exhibited by variables in continuous chemical processes makes the diagnosis of malfunctions in these processes especially difficult. While the SDG account: for the initial process response, the ESDG is guaranteed to account for both the initial and ultimate process response. In addition, the number of spurious patterns produced in simulating process behavior with the ESDG is "minimized". Furthermore, most process transients are accounted for by the ESDG. Because all non-local causalities are identified, the inclusion of the correct malfunction origin can be assured without degrading diagnostic resolution in systems based on the ESDG. This makes the ESDG an appropriate model for malfunction diagnosis of continuous dynamic processes.

### **8.3. Developing Process Diagnostic Models for MIDAS**

The Model Integrated Diagnostic Analysis System (MIDAS) is an on-line diagnostic aid for continuous chemical and refinery process. Using artificial intelligence techniques, MIDAS is designed to solve diagnostic problems in the dynamic process environment. In order to obtain a diagnosis, MIDAS relies on a process diagnostic knowledge base<sup>4</sup> consisting of qualitative causal and quantitative constraint relationships.

An algorithmic process independent procedure for deriving the diagnostic PM from process flowsheet information was implemented. Using the procedure, diagnostic knowledge bases can be systematically produced with more reliability and consistency than rule sets in conventional rule based expert systems. The effort in developing the PM is largely reduced to specifying units from a model library according to the flowsheet structure and specifying the signs of certain variable and parameter relationships, such as relative temperatures in adjacent process units. Knowledge bases for diagnostic systems may be produced in hours or days, rather than weeks or months as is usually the case for rule based expert systems.

The PM is constructed in several stages. First, the process SDG is created from its constituent units and information about flowsheet structure. Algorithms for instantiating SDG unit models from a model library and linking the models according to the flowsheet structure were developed. Next, non-local causalities stemming from global interactions in the flowsheet are uncovered, resulting in the process ESDG. An algorithm that applies the theorems<sup>5</sup> for deriving the ESDG was implemented. Following this step, another algorithm was developed to eliminate unmeasured variables from the ESDG to obtain the causal relationships expressed in the PM. Finally, causal relationships in the PM are enhanced with quantitative constraint information. Guidelines for deriving relationships between quantitative constraints in the PM and process malfunctions were suggested.

The procedure for developing the PM gives MIDAS the following characteristics:

---

<sup>4</sup> The diagnostic knowledge base is also referred to as the Process Model (PM).

<sup>5</sup> The criteria implemented assume no transitions between qualitative regimes occur.

- Knowledge of non-monotonic behaviors arising from global process interactions is represented. Therefore, an accurate diagnosis can be obtained in a dynamic process environment without unnecessary losses of diagnostic resolution.
- The PM relates malfunctions directly to process observations. Thus a diagnosis can be made using the existing process instrumentation scheme. Postulating and retracting assumptions about states of unmeasured variables is avoided during diagnostic inference.
- A "family" of PMs can be created by utilizing varying amounts of process-specific information. Thus, diagnostic conclusions are "tailored" to the amount of available quantitative process information.

An evaluation of the performance of MIDAS for a simulated jacketed reactor process was performed. Based on a sample of 77 simulation runs (in which non-monotonic behavior was observed in at least 40% of the cases), an accurate diagnosis was obtained in 93% of the cases after the initial event and in 100% of the cases at the end of the transient. On average 96% of the malfunctions in the process were eliminated<sup>6</sup> by the end of the transient. The acceptable performance of MIDAS validates the use of the ESDG as a basis for process diagnostic models and verifies the procedures implemented for developing the PM.

## **8.4. Recommendations**

The initial success of MIDAS has stimulated interest in applying MIDAS to industrial scale problems. Plans are underway to further evaluate MIDAS in actual or simulated process environments. However, a disappointing result of this research project was the long times required for creating the PM. In order to construct the PM of large scale processes in reasonable time, more efficient algorithms are needed. Furthermore, PMs derived by the current implementation of the algorithms are valid only when transitions to other qualitative regimes do not occur.

---

<sup>6</sup> The resolution obtained was near the theoretical maximum, given the available process instrumentation.

Thus, useful avenues for future research in developing qualitative process models of continuous processes for MIDAS would be to:

- Extend the theorems for identifying non-local causalities to the more general case of transitions across qualitative regimes.
- Incorporate criteria for identifying non-local causalities arising from controller saturation and other transitions across qualitative regimes in the algorithms for deriving the PM.
- Devise and implement more efficient algorithms for deriving the PM.

Other avenues for further research on automated malfunction diagnosis for continuous process systems derive from fundamental limitations of qualitative modeling. One of the limitations of qualitative modeling in MIDAS is the inability to use evidence about the non occurrence of events in diagnostic inference. This situation can be alleviated by associating approximate quantitative delays and strengths with causal effects between process observations. The use of **semi-quantitative** modeling formalisms to derive these approximate relationships is suggested as an area for future research. Possible approaches that may be considered are those based on interval arithmetic, and on order of magnitude relationships among process parameter groups.

Another logical extension of this research would be to apply concepts developed in this thesis to qualitative modeling of **semi-continuous** and **batch** processes. In diagnostic applications, abnormal process conditions may be identified from deviations of variable states and variable trends from nominal values. Thus, rather than explaining deviations in patterns of states alone, the model should also account for patterns of deviations of process trends. The major challenge once again will be to deal with the inherent ambiguity in qualitative modeling when deriving the diagnostic knowledge base. MIDAS is capable of reasoning with process trend information once it is represented in the PM. The ability to derive process diagnostic knowledge bases for semi-continuous and batch processes will extend the range of applicability of MIDAS in industrial processes.

## 9. MIDAS Model Builder

The model builder utility of MIDAS is used to build the SDG of a process from its flowsheet. The utility consists of six (6) files in the C:\GW\KBS\MIDAS\BUILDER directory and a directory of unit model creation programs, C:\GW\KBS\MIDAS\BUILDER\MODELLIB. Files in BUILDER\MODELLIB<sup>1</sup> contain lisp programs for building prespecified Signed Directed Graph (SDG) models<sup>2</sup> for particular types of process units. Files in BUILDER contain other utility programs to assist the user in building the process SDG from the flowsheet of the process. The files and their descriptions are summarized in Tables 9.1 and 9.2.

Table 9.1 Builder Utility Files

Name	Description
CODES10B	Table of codes for SDG objects
FRAME10B	SDG object frame definitions
LOADF10B	Builder utility files loading program
MODEL10B	Batch input file reader
MPROC10B	Interactive flowsheet building programs
MUNIT10B	General unit model creation programs

Loading these files into the GOLDWORKS environment and running programs contained in the files are order sensitive operations. All files in BUILDER must be loaded prior to

<sup>1</sup> In the rest of this chapter, \GW\KBS\MIDAS\BUILDER\MODELLIB and \GW\KBS\MIDAS\BUILDER will be referred to as BUILDER and MODELLIB, respectively.

<sup>2</sup>In this chapter, subsequent use of "model" refers to SDG models.

running any of the programs in MODELLIB. The programs are designed to ensure that the files are loaded and run in the correct order when used as recommended. A program in the file LOADF10B (Section 9.1) is provided for this purpose.

Table 9.2 Model Library Files

Name	Process Unit
CNTL10L	Controller with integral action
CPUMP10L	Centrifugal pump
CSTR10L	Continuous stirred tank reactor
CVAL10L	Control valve
EFFL10L	Effluent stream
FEED10L	Feed stream
JCSTR10L	Continuous stirred tank reactor with jacket
JUNC10L	Splitter/mixing junction
MVAL10L	Manual valve
PIPE10L	Pipe
SENS10L	Sensor (measuring device)
STHTX10L	Shell and tube heat exchanger
TANK10L	Tank

## 9.1. Loading Builder Utility Files

The LOADF10B file contains a program to load all files in BUILDER. To run the program:

- 1) Load Goldworks. If Goldworks has been previously loaded, omit this step.
- 2) From the Goldworks Menu Interface, click on the TOP LEVEL LISP option under the SYSTEM menu item.
- 3a) If the LOADF10B file is located in the BUILDER directory in the C: device, at the GOLDWORKS prompt, enter:  
**(load "C:\\GW\\KBS\\MIDAS\\BUILDER\\LOADF10B")<sup>3</sup>**

or,

- 3b) Set the default directory to the directory containing the LOADF10B file. This can be done at the GOLDWORKS prompt by entering: **(cd pathname)<sup>4</sup>**

and,

- 3c) Then enter: **(load "LOADF10B")**

Utility programs in BUILDER will be loaded into the GOLDWORKS environment in the correct order as indicated by the following messages.

"Loading frame definitions"  
"Loading SDG object codes"  
"Loading general model creation programs"  
"Loading interactive flowsheet building routines"  
"Loading batch input file reader"

## 9.2. Running a Model Building Session

The interactive MIDAS Process Flowsheet Model Building Utility can be started immediately after loading files in BUILDER. If process SDG objects are currently in the

---

<sup>3</sup>The double slash format is used when entering pathnames from TOP LEVEL LISP.

<sup>4</sup>Pathname is a Goldworks pathname. Examples are "C:\\GW\\KBS\\MIDAS\\" for directories and "C:\\GW\\KBS\\MIDAS\\FOO.LSP" for files.

GOLDWORKS environment, these files should not be loaded again.<sup>5</sup> The utility guides the user in creating the SDG of a process as follows:

- 1) selecting user-defined unit models and/or predefined SDG unit models from the model library,
- 2) creating instances of SDG objects for these unit, and
- 3) connecting the units according to the flowsheet.

To start a model building session enter: **(BUILD-MODEL)**<sup>6</sup>

At this point, a banner indicating the MIDAS Process Flowsheet Model Building Utility is displayed. The response to each question asked by the utility is echoed and then confirmed by the user. Finally, each response is checked by the program, ensuring that it is valid. If an invalid response is entered, the utility prompts for a valid response. Details of a typical model building session are described in the remainder of this chapter.

### **9.2.1. Starting with a Partially Complete Process Model**

After the banner is displayed, the user can choose to save, delete or load a partially completed process SDG model. These actions are described below. By a combination of these actions, model building may be interrupted and resumed during another session.

If the current GOLDWORKS environment contains no object instances, to build a completely new SDG model from a flowsheet:

- 1) Enter NO in response to all questions in this section.
- 2) Proceed to Section 9.2.2.

---

<sup>5</sup>If instances of SDG objects exist in the GOLDWORKS environment, loading these files will result in error messages.

<sup>6</sup>This program is in the file, MPROC10B.

## *Save Process Model*

This action saves the model in the GOLDWORKS environment in a file. The action is recommended if the current model has not been saved previously.<sup>7</sup>

The program asks the following.

1) Enter [yes/no] if you want to save the existing model >

If instances of SDG objects are to be saved, enter YES. Otherwise, enter NO. If entered YES, see (2) below. If entered NO, (2) is bypassed.

2) Enter filename to save process model >

Enter the **pathname** of the file to which the current model in the GOLDWORKS environment should be saved.

All contents of the named file are deleted and all instances of SDG objects are saved in the named file. If the directory containing the indicated file does not exist an error message is indicated. The program can be aborted<sup>8</sup> by using the **Ctrl-C** command at the >1 prompt, and the utility started again by entering: (**BUILD-MODEL**).

## *Delete Process Model*

This action deletes all instances of SDG objects in the GOLDWORKS environment. It is impossible to reclaim these objects once deleted, and it is advised that SDG objects in the current GOLDWORKS environment should be saved just before deleting the model.

---

<sup>7</sup>If instances of SDG objects for a unit have been erroneously instantiated, it is more convenient to resume building the model using a previous file containing a partial model than to edit the file containing the current model.

<sup>8</sup>Aborting the utility does not delete instances of SDG objects in the GOLDWORKS environment.

The action is recommended if SDG objects for a process flowsheet are present in the GOLDWORKS environment and if the user intends to

- 1) begin building a new model,
- 2) continue building a model from a different flowsheet, or
- 3) resume building the model from a partially complete model obtained in a previous session, if SDG objects have been erroneously instantiated.

The program asks the following.

- 1) Enter [yes/no] if you want to delete the existing model >

If entered YES, all instances of SDG objects are deleted from the GOLDWORKS environment. If entered NO, the GOLDWORKS environment is not modified.

### *Load Process Model*

This action loads instances of SDG objects created in a previous session. It is recommended that all SDG objects in the GOLDWORKS environment are deleted just prior to loading the partially complete model.

The program asks the following.

- 1) Enter [yes/no] if you want to load the existing model >

If instances of SDG objects are to be loaded, enter YES. Otherwise, enter NO. If entered YES, see (2) below. If entered NO, (2) is bypassed.

- 2) Enter filename to load process model >

Enter the pathname of the file from which the partially completed model should be loaded.

All instances of SDG objects in the file are loaded into the GOLDWORKS environment. If the indicated file does not exist an error message is displayed. The program can be aborted by using the **Ctrl-C** command at the >1 prompt, and the utility started again by entering (**BUILD-MODEL**).

## 9.2.2. Resume Model Building

After completing the actions described in the preceding section, the user may continue building a partially complete model for the flowsheet or begin building a new model for a different flowsheet. Building the model for a flowsheet involves three (3) major steps.

- 1) Instantiate SDG objects for process units (except sensors).
- 2) Instantiate SDG objects for sensors.
- 3) Connect all units and sensors in the flowsheet.

Each step outlined above must be completed before starting the next step<sup>9</sup>. Otherwise, the SDG of the process may be inconsistent and may cause subsequent errors during model translation (Chapter 10).

### 9.2.2.1. ADDING UNITS TO THE PROCESS MODEL

The first step in building the process SDG is to sequentially instantiate SDG object instances for units in the flowsheet and add these objects to the preexisting process model.

The program asks the following.

- 1) Enter [yes/no] if you want to add units >

If SDG object instances for process units are to be added to the partially completed SDG model, enter **YES**. Otherwise, enter **NO**. If entered **YES**, a message

---

<sup>9</sup>For example, connecting units in a previous session then adding units in a subsequent session will lead to errors.

indicating that units are to be added to the process model is displayed, then proceed to Questions 2) - 4) below. If entered NO, 2) - 4) are bypassed.

2) Enter pathname of unit model directory >

Enter the **pathname** of the directory<sup>10</sup> containing the library programs for building unit models.

Programs for building feed stream and effluent stream models are loaded from the model library as indicated by messages. If the directory does not exist or does not contain the routines for building feed or effluent streams an error message is displayed. The program can be aborted by using the **Ctrl-C** command at the >1 prompt, and the utility started again by entering (**BUILD-MODEL**).

3) Enter as a list, the chemical species present in the process >

A list<sup>11</sup> of symbols, denoting chemical species in the process knowledge base is displayed. Elements of this list do not represent names or formulae of chemical species but allow the symbols to be mapped into actual chemical species. Enter a **subset** of the list. Subsequently, only members of the subset can be entered for individual process units. To increase the number of species in the displayed list, see Section 9.3.3.1.

The utility provides two means of adding instances of SDG objects for process units to the existing process. One method involves executing interactive lisp programs in the unit model library (Section 9.3), and the other is to execute a program that allows the unit model to be specified in batch files using a specialized input language (Section 9.4).

4) Add another unit to the existing process >

A list of unit types is displayed. This list can be displayed again before selecting a unit. To end the current model building session or after all units have been added to

---

<sup>10</sup>All files containing programs that create feed streams, effluent streams and other unit models must be in the same directory. For example, "C:\GW\KBS\MIDAS\BUILDER\MODELLIB\".

<sup>11</sup>A LISP list of elements E1,E2, ... En is entered as: (E1 E2 ... En). An empty list may be entered as: (), or: NIL.

complete the model, enter **0**. To specify the unit model using the batch input file reader, enter **1**. To add the desired unit using programs in the model library, enter the **number**<sup>12</sup> associated with the unit. If entered 0, proceed to Section 9.2.2.2. If entered 1, proceed to Section 9.4. Otherwise, proceed to Section 9.3. After answering the relevant questions in Sections 9.3 or 9.4, instances of SDG objects for the unit are created and another unit can be added to the process.

### 9.2.2.2. ADDING SENSORS TO THE PROCESS MODEL

After instances of SDG objects for all process units (excluding sensors) have been created, sensors can be added to the process model. It is crucial that instances of SDG objects for a process unit are created before creating instances of SDG objects for a sensor measuring a variable in the unit. Thus, it is advisable to create instances of objects for all process units before creating any sensor model.

The program asks the following.

1) Enter [yes/no] if you want to add sensors to the existing process >

If sensors are to be added to the process model, enter **YES**. Otherwise, enter **NO**. If entered **YES**, a message indicating that sensors are to be added is displayed, then respond to Questions 2) and 3) below. If answered **NO**, 2) and 3) are bypassed.

2) Enter pathname of sensor model directory >

Enter the **pathname** of the directory containing the program for building sensor models.

The program for building sensor models is loaded from the library as indicated by messages. If the directory does not exist or does not contain the program an error message is displayed. The program can be aborted by using the **Ctrl-C** command at the >1 prompt, and the utility started again by entering **(BUILD-MODEL)**.

---

<sup>12</sup>The program for creating instances of SDG objects for a particular type of unit are loaded into the GOLDWORKS environment, the first time the unit is selected in the current model building session.

3) Enter [yes/no] to add another sensor to the process >

To end the current model building session, or after all sensors have been added to the process, enter **NO**. If entered **NO**, proceed to Section 9.2.2.3. If entered **YES**, answer questions (see Section 9.3) relating to the next sensor. After answering the questions, SDG object instances for the sensor model are created and another sensor can be added to the process.

### 9.2.2.3. CONNECTING UNITS AND SENSORS IN THE FLOWSHEET

The last stage when building the process SDG is to connect instances of objects for units and sensors in the flowsheet. This action deletes all instances of redundant nodes<sup>13</sup> and causal arcs connecting these nodes from the process SDG. Connecting units before all instances of SDG objects for all process units and sensors have been instantiated may lead to subsequent errors during model building or translation.

This step must be performed prior to model translation, otherwise the Extended Signed Directed Graph (ESDG) and subsequently the Process Model (PM) produced by the model translator will be incorrect.

The program asks the following question.

1) Enter [yes/no] to connect units and sensors in the process >

To connect units, enter **YES**. Otherwise, to end the current session without connecting units, enter **NO**. If entered **YES**, a message indicating that the units are being connected is displayed and the appropriate instances are deleted. If answered **NO**, the GOLDWORKS environment is not modified.

### 9.2.2.4. ENDING THE SESSION

Just before ending the current session, the model in the current GOLDWORKS environment may be saved. This action is recommended especially if it is intended to delete

---

<sup>13</sup>Nodes representing identically equal variables in adjacent units.

instances of SDG objects or it is intended to exit the GOLDWORKS environment and go back to the DOS operating system.

The program asks the following.

- 1) Enter [yes/no] if you want to save the existing model >

If instances of SDG objects are to be saved enter YES. Otherwise, enter NO. If entered YES, see (2) below. If entered NO, (2) is bypassed. The current model building session is ended as indicated by the displayed message, and the program returns execution to TOP LEVEL LISP.

- 2) Enter filename to save process model >

Enter the **pathname** of the file in which the current model in the GOLDWORKS environment should be saved.

All contents of the named file are deleted and all instances of SDG objects are saved in the file. The current model building session is ended as indicated by a displayed message, and the program returns execution to TOP LEVEL LISP.

If the directory containing the indicated file does not exist an error message is indicated. To save the objects:

- 1) Abort the program by using the **Ctrl-C** command at the >1 prompt.
- (2) Enter: **(gw-save :instance pathname)**

### **9.3. Using the Model Library to Create SDG Object Instances**

A modular approach is used to instantiate the SDG of a process from SDG models of its constituent units. Guidelines for deriving SDG objects (and their relationships) for process units are discussed in Chapter 4. One means of creating instances of SDG objects

instances for a particular unit is to execute an interactive lisp program in the unit model library. The pertinent program is executed when:

- 1) the unit type is selected in response to Question 4 in Section 9.2.2.1., or
- 2) YES is entered in response to Question 3 in Section 9.2.2.2.

Before SDG objects for the unit or sensor are instantiated, a series of questions is asked. It is not convenient to recover<sup>14</sup> from an incorrect response once instances of SDG objects for the unit have been created (Section 9.2.1.1.), thus, the programs provide three (3) levels to verify user responses before instances of SDG objects are created:

- 1) The response to individual questions are echoed and confirmed by the user. If an invalid response is entered, the program prompts for the response to be entered again.
- 2) After all questions for the unit have been answered, the set of all responses for the unit are echoed again. Once again, the user has another opportunity to confirm the set of responses before instances are created.
- 3) Finally, QUIT may be entered in response to any question at any time before the set of malfunctions affecting the unit is selected. The response to all questions asked (for the unit) are canceled and the program is restarted, asking the first question in the sequence.

In general, the questions asked for each unit can be classified as those that:

- 1) are required to uniquely identify the unit,
- 2) are used to verify responses and instantiate SDG objects,
- 3) are related to connectivity in the flowsheet,
- 4) determine causal relationships for the given unit and
- 5) select a subset of root causes (or malfunctions) affecting the unit from a displayed set of root causes.

---

<sup>14</sup>The effect of an incorrect response on the resulting SDG is not always so straightforward and requires understanding the SDG creation guidelines and model creation programs in some detail.

Questions required to identify units, verify responses, create instances of SDG objects and to select root causes are common to all units. These are described in Section 9.3.1. Other questions depend on the unit type, and are described in the appropriate subsection of Section 9.3.2.

### **9.3.1. Questions Common to all Units**

The unit model creation programs ask the following questions for all types of units.

#### *Unit ID*

1) Enter a unique ID for the unit >

A unique ID of the form U-# (where # is a positive integer) not previously assigned to an existing unit in the current GOLDWORKS environment is entered. The program checks that the ID just selected is unique. If the ID just selected is not unique, the program prompts for another ID to be entered. The program ensures the ID is of the required form and prompts the user to enter the ID again if an incorrect form is entered.

#### *Unit Name*

2) Enter a unique name to identify the unit >

A unique **name** not previously assigned to an existing unit in the current GOLDWORKS environment is entered. The program checks that the name just selected is unique. If the name just selected is not unique, the program prompts for another name to be entered. The name may be any continuous sequence of letters and numbers separated by hyphens.

#### *Unit Documentation String*

3) Enter the documentation string for the unit >

A string<sup>15</sup> is entered. The documentation string is intended to be more descriptive than the name of the unit. The string need not be unique. Unlike most other responses, the string just entered is not echoed and confirmed by the user.

After the documentation string is entered, the program proceeds and asks other questions pertaining to the type of unit (Section 9.3.2.), after which the set of responses for the unit is displayed. Then, the program asks the following.

### *Display Unit Responses*

4) Enter [yes/no] to redisplay all responses for this unit >

To display all the responses<sup>16</sup> for the unit, enter YES. Otherwise, enter NO. If entered YES, the set of responses is displayed again and the program repeats the question. If entered NO, the program proceeds to the next question.

### *Verify Responses*

5) Enter [yes/no] if the above responses are correct >

If the displayed responses are correct, enter YES, otherwise, NO. If entered YES, the program asks proceeds to the next question. Entering NO, has the same effect as entering QUIT. Caution should be exercised if NO is entered. If NO is entered, all responses for the unit are canceled, then the program proceeds to Question 7.

### *Create Unit SDG Instances*

6) Enter [yes/no] to create instances of SDG objects for the unit >

---

<sup>15</sup>A LISP string is a sequence of characters enclosed by quotation marks, e.g. "This is a string".

<sup>16</sup>Some responses are derived from earlier responses to other questions.

To create instances of SDG objects, enter YES. Otherwise, enter NO. Once instances are created, deleting the instances is not easily affected. Therefore, care should be exercised when responding to this question. If entered YES, the program creates instances of SDG objects for the unit and for any FEED stream or EFFLUENT stream associated with the unit. Root causes are selected for feed and effluent streams immediately after object instances are created for the particular unit. The program continues with Question 4, Section 9.2.2.1. or Question 3, Section 9.2.2.2. and the next unit or sensor can be added to the process SDG. Entering NO, has the same effect as entering QUIT. If NO is entered, all responses for the unit are deleted, then the program proceeds to Question 8.

### *Selecting Root Causes that Affect the Unit*

A prespecified set of root causes are created for each unit during instantiation of SDG objects for the unit. The user may select the actual root causes affecting the particular instance of the unit from among this set. The program asks,

7) Enter [ALL] or the list of root causes affecting unit-name >

A list of all root causes associated with unit-type is displayed. This list can be redisplayed before selecting desired root causes. To select the desired root causes, enter the list of numbers associated with the root causes. All other root cause instances for the unit are deleted from the GOLDWORKS environment as indicated by messages. If all root causes are to be selected, enter ALL. Note: QUIT is an invalid response to this question.

### *When Quit is Entered*

**QUIT** can be entered in response to any question before selecting root causes. The program does not echo this response for confirmation, thus, QUIT should be entered with caution. Once QUIT is entered, previous responses for the unit are canceled. Then the program proceeds by asking,

8) Enter [yes/no] if you want to quit >

To restart building a model of the same type, enter **NO**. Otherwise, enter **YES**. If entered **NO**, the program continues by asking for the unit name (see Question 1). If entered **YES**, execution is returned to Question 4 in Section 9.2.2.1. or Question 3 in Section 9.2.2.2. and another type of unit or another sensor can be added to the process.

Using **QUIT** is recommended if it is discovered that the response to a previous question pertaining to the unit is incorrect. Entering **QUIT** avoids responding to subsequent questions for the unit.

### **9.3.2. Questions Specific to a Particular Type of Unit**

Questions asked in the preceding section are common to all units. The questions are used mainly for bookkeeping purposes and to assist the user in recovering from incorrect responses.

Other questions pertaining to a unit are asked after Question 3 but before Question 4 of Section 9.3.1. The nature of the questions depend on the particular type of unit and are described in this section. These additional questions specify the structure of the SDG of the unit and are derived from the guidelines in Chapter 4.

Table 9.3 Unit Library Connectivity Matrix

Process Unit	Flow in	Flow out	Inf. in	Inf. out
Controller	0	0	1/2	1
Centrifugal pump	1	1	0	V
CSTR	V	V	0	V
Control valve	1	1	1	V
Effluent stream	1	0	0	V
Feed stream	0	1	0	V
CSTR with jacket	V	V	0	V
Junction	V	V	0	V
Manual valve	1	1	0	V
Pipe	1	1	0	V
Sensor	0	0	1	0/1
S & T heat exchanger	2	2	0	V
Tank	V	V	0	V

The SDG of a process unit is defined by the:

- 1) connectivity of the unit to adjacent units,
- 2) assumptions pertaining to the nature of the processes occurring in the unit,<sup>17</sup> and
- 3) ordinal relationships between values of variables in the unit and its adjacent units.

---

<sup>17</sup> Examples of these processes are computation, chemical reaction, momentum, heat and mass transfer.

The interactive programs in the unit model library are designed to be quite general and produce the SDG for a unit in different contexts. A different combination of responses to the questions results in a different SDG for the unit.

One class of questions asked by the library programs relate to the connectivity of the unit in the process. The number of units from/to which material flows to/from each unit and from/to which information signals are transmitted to/from each unit in the library is summarized in Table 9.3. In the table, V, refers to a variable number of units.

Other questions relate to assumptions about the processes occurring in the unit and to the relationship between values of variables in the unit and its adjacent units.

The assumptions relating to unit SDG models created by the programs and questions asked for each unit in the library are described in the following subsections.

#### 9.3.2.1. CONTROLLER

The program creating an instance of a controller creates a computing device with one (1) output port and either one (1) or two (2) input ports. The second input port is required if the controller receives the value for its setpoint from another unit. The controller output and setpoint may have signal indicators. Under nominal (fault-free) conditions, assumptions governing controller models created by the program are:

- 1) The controller includes the integral mode of action.
- 2) The controller is used in a SISO feedback or cascade configuration.

The user can select from among the following list<sup>18</sup> of root causes for the particular instance of a controller:

- 1) Controller output signal failed high.
- 2) Controller output signal failed low.
- 3) Controller output signal biased high.
- 4) Controller output signal biased low.

---

<sup>18</sup>Root causes 5 & 6 cannot be selected for secondary controllers in cascade loops.

- 5) Controller setpoint high.
- 6) Controller setpoint low.

The questions asked for a controller are described below.

- 1) Enter [yes/no] if increasing the setpoint of cntl-name increases the controller output signal >

If increasing the setpoint increases the output signal, enter YES. Otherwise, enter NO.

- 2) Enter [yes/no] if the setpoint of cntl-name is computed by another unit >

If the controller is a "slave" (secondary) controller in a cascade loop, enter YES. Otherwise, enter NO.

- 3) Enter the ID of the unit that computes the setpoint of cntl-name >

Enter a unit ID number, U-#. Note: If NO was entered in response to the previous question, this question is not asked.

- 4) Enter [yes/no] if the setpoint of cntl-name has an indicator signal >

If the setpoint has an indicator signal, enter YES. Otherwise, enter NO. Note: If NO was entered in response Question 2, this question is not asked.

- 5) Enter the ID of the unit sending the controlled variable signal to cntl-name >

Enter a unit ID number, U-#.

- 6) Enter the ID of the unit receiving the output signal from cntl-name >

Enter a unit ID number, U-#.

- 7) Enter [yes/no] if the output of cntl-name has an indicator signal >

If the controller output has an indicator signal, enter YES. Otherwise, enter NO.

After the response to this question is answered the program displays the complete set of responses for the controller and proceeds to Question 4 in Section 9.3.1.

### 9.3.2.2. CENTRIFUGAL PUMP

The program creating an instance of a centrifugal pump creates a two (2) port device with connections to one (1) upstream and one (1) downstream unit. The pump is electrically driven and the motor may have an electric current/power indicator. Additionally, there can be as many sensors connected to the pump as there are variables in the pump. Under nominal conditions, assumptions governing centrifugal pump models created by the program are:

- 1) The pump contains a subset of the chemical species present in the process.
- 2) No chemical reaction occurs in the pump.
- 3) The fluid in the pump is incompressible with constant density.
- 4) The pump is adiabatic.
- 5) Pump fluid temperature and pressure are greater than respective ambient conditions.

The user can select from among the following list of root causes for the particular instance of a centrifugal pump:

- 1) Pump blockage.
- 2) Pump outlet leak.
- 3) Loss of insulation.
- 4) Fire.
- 5) Pump overheating.
- 6) Pump cavitation.
- 7) Electric current to pump motor high.
- 8) Electric current to pump motor low.
- 9) Pump motor failure.
- 10) Pump shaft failure.

The questions asked for a centrifugal pump are described below.

1) Enter [yes/no] if cpump-name has an electric current/power indicator >

If the electric motor driving the pump has an electric current or power indicator, enter YES. Otherwise, enter NO.

2) Enter the chemical species present in cpump-name >

The list of species present in the process may be displayed. A list of the species present in the pump is entered. The program verifies that each specie just entered is present in the process. Note: if NIL was entered in response to Question 3 in Section 9.2.2.1., this question is not asked.

3) Enter the ID of the unit upstream of cpump-name >

Enter NONE or a unit ID number, U-#. If no unit exists upstream of the pump, NONE is entered and the program subsequently creates a FEED stream upstream of the pump. Otherwise, the ID of the unit upstream of the pump is entered.

4) Enter [yes/no] if {ID # of upstream unit} has multiple<sup>19</sup> outlets >

If the unit upstream of the pump has more than one outlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous question indicating that the upstream unit is a FEED stream, this question is not asked and its answer defaults to NO.

5) Enter the ID of the unit downstream of cpump-name >

Enter NONE or a unit ID number, U-#. If no unit exists downstream of the pump, NONE is entered and the program subsequently creates an EFFLUENT stream downstream of the pump. Otherwise, the ID of the unit downstream of the pump is entered.

6) Enter [yes/no] if {ID # of downstream unit} has multiple inlets >

---

<sup>19</sup>The response determines the name of instances of SDG objects (see Section 9.6).

If the unit downstream of the pump has more than one inlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous question indicating that the downstream unit is an EFFLUENT stream, this question is not asked and its answer defaults to NO.

After the response to this question is answered the program displays the complete set of responses for the pump and proceeds to Question 4 in Section 9.3.1.

### 9.3.2.3. CONTINUOUS STIRRED TANK REACTOR

The program creating an instance of a continuous stirred tank reactor (CSTR) creates a multi-port device with storage capacity and possible connections to multiple upstream and downstream units. A generalized set of catalyzed liquid phase chemical reactions<sup>20</sup> takes place in the CSTR. Currently the maximum number of both catalysts and independent reactions in the network is 5. To increase these limits, see Section 9.3.3.1 There can be as many sensors connected to the CSTR as there are variables in the CSTR. Under nominal conditions, assumptions governing CSTR models created by the program are:

- 1) The CSTR contains a subset of the chemical species present in the process.
- 2) The fluid the CSTR is incompressible with constant density.
- 3) The contents of the reactor are well mixed and the concentration of all species in each outlet stream are equal to their concentrations in the reactor.
- 4) Liquid (fluid) temperatures in each outlet stream are equal to the reactor temperature. Reactor temperature is greater than ambient temperature.
- 5) The CSTR is adiabatic.
- 6) In sealed (non-vented) CSTR's, compression (expansion) in the vapor space above the liquid is isothermal.
- 7) Catalyst activity is independent of specie concentrations.

The user can select from among the following list of root causes for the particular instance of a CSTR:

---

<sup>20</sup>Each reaction may be (ir)reversible, exo/endothermic, of positive/negative order with respect to reactants/products and may be catalyzed by more than one catalyst. By convention, the steady state concentration of a reactant/product is greater/less than its equilibrium composition.

- 1) Blockage in each inlet port.
- 2) Blockage in each outlet port.
- 3) Leak in each outlet port.
- 4) Leak in CSTR.
- 5) Loss of insulation.
- 6) Fire.
- 7) Loss of activity for each catalyst.
- 8) Inadequate mixing in reactor.

The questions asked for a CSTR are described below.

- 1) Enter [yes/no] if cstr-name is vented >

If the CSTR is open to the surroundings, enter YES. Otherwise, enter NO.

- 2) Enter the chemical species at outlets of cstr-name >

If no chemical species are present in the process a warning message is displayed. The list of species present in the process may be displayed. A list of the species present in all outlet ports is entered. The program verifies that each species just entered is present in the process. Note: if NIL was entered in response to Question 3 in Section 9.2.2.1., this question is not asked.

- 3) Enter the number of units upstream of cstr-name >

The number of units upstream of the CSTR (including feed streams) is entered. The program verifies that the response is a positive integer.

Questions 4-9 are asked for each upstream unit.

- 4) Enter the ID of the next unit upstream of cstr-name >

Enter NONE or a unit ID number, U-#. If no unit exists upstream of the CSTR, NONE is entered and the program subsequently creates a FEED stream upstream of the CSTR. Otherwise, the ID of the next unit upstream of the CSTR is entered.

Note: The ID's of all other upstream units must be entered in sequence before specifying FEED streams.

- 5) Enter [yes/no] if the inlet from {ID # of this upstream unit} is below the liquid level of cstr-name >

If the inlet is below the liquid level of cstr-name, enter YES. Otherwise, enter NO.

- 6) Enter [yes/no] if {ID # of this upstream unit} has multiple outlets >

If the next unit upstream of the CSTR has more than one outlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous question indicating that the upstream unit is a FEED stream, this question is not asked and the answer defaults to NO.

- 7) Enter the relationship of the outlet temperature of {ID # of this upstream unit} to the outlet temperature of cstr-name [>, = or <] >

If the temperature of the inlet stream from the upstream unit is greater than the temperature of the outlet streams of the CSTR, enter >. If the temperatures are equal, enter =. Else, enter <.

Question 8 and 9 are asked for each species in cstr-name.

- 8) Enter [yes/no] if the {next species in cstr-name} is present in {ID # of this upstream unit} >

If the displayed chemical species is present in the upstream unit, enter YES. Otherwise, enter NO.

- 9) Enter the relationship of the outlet concentration of {the next species} in {ID # of this upstream unit} to its outlet concentration in cstr-name [>, = or <] >

If the concentration of the displayed specie in the inlet stream from the upstream unit is greater than its concentration in the outlet streams of the CSTR, enter >. If

the concentrations are equal, enter =. Else, enter <. Note: If NO was entered in response to Question 8, the question is not asked and the answer defaults to <.

10) Enter the number of units downstream of cstr-name >

The **number** of units downstream of the CSTR (including effluent streams) is entered. The program verifies that the response is a positive integer.

Questions 11 and 12 are asked for each downstream unit.

11) Enter the ID of the next unit downstream of cstr-name >

Enter **NONE** or a unit ID number, **U-#**. If no unit exists downstream of the CSTR, NONE is entered and the program subsequently creates an EFFLUENT stream downstream of the CSTR. Otherwise, the ID of the next unit downstream of the CSTR is entered. Note: The ID of all other downstream units must be entered in the sequence before specifying EFFLUENT streams.

12) Enter [yes/no] if {ID # of this downstream unit} has multiple inlets >

If the next unit downstream of the CSTR has more than one inlet flow port, enter **YES**. Otherwise, enter **NO**. Note: if NONE was entered in response to the previous question indicating that the downstream unit is an EFFLUENT stream, this question is not asked and the answer defaults to **NO**.

13) Enter [endo/exo/nil] if the net heat effect of all reactions in cstr-name is endothermic, exothermic or athermal >

If the net effect of all reactions taking place in cstr-name is the consumption of heat, enter **ENDO**. If heat is released, enter **EXO**. Otherwise, enter **NIL**.

Question 14 is asked for each species in the cstr-name.

14) Enter [react/prod/nil] if the {next specie in cstr-name} is a net product, reactant or neither >

If the net effect of the reaction network is to consume the displayed specie, enter **REACT**. If the specie is produced, enter **PROD**. Otherwise, enter **NIL**.

15) Enter the number of independent reactions taking place in cstr-name >

The total **number** of independent reactions in the network taking place in cstr-name is entered. The response must be a positive integer not exceeding the maximum number of possible reactions.

16) Enter the number of catalysts catalyzing the reaction network in cstr-name >

The **number** of catalysts catalyzing the reaction network in cstr-name is entered. The response must be a positive integer not exceeding the maximum number of possible catalysts.

Questions 17 - 21 are asked for each reaction in the network. Question 17 is also asked for each catalyst.

17) Enter [yes/no] if {the next catalyst in cstr-name} catalyzes {the next reaction in the network} >

If the displayed catalyst catalyzes the reaction, enter **YES**. Otherwise, enter **NO**.

18) Enter [rev/irrev] if {the next reaction} is reversible or irreversible >

If the reaction is reversible, enter **REV**. If the reaction is irreversible, enter **IRREV**.

19) Enter [endo/exo/nil] if {the next reaction} is endothermic, exothermic or athermal >

If the reaction consumes heat, enter **ENDO**. If heat is released, enter **EXO**. Otherwise, enter **NIL**.

Questions 20 - 21 are asked for each specie in cstr-name.

- 20) Enter [react/prod-nil] if {the next specie in cstr-name} in {the next reaction} is a reactant, product or neither >

If the displayed specie is consumed by the reaction, enter **REACT**. If the specie is produced, enter **PROD**. If the specie is not involved in the reaction, enter **NIL**.

- 21) Enter [inc/dec-nil] if the rate of {the next reaction} increases, decreases or is unaffected by increasing the concentration of {the next specie in cstr-name} >

If increasing the concentration of the displayed specie increases the rate of the reaction, enter **INC**. If the rate of the reaction is decreased, enter **DEC**. Otherwise, enter **NIL**. Note: If **NIL** was entered in response to the previous question, this question is not asked, and the answer defaults to **NIL**. If **IRREV** was entered in response to Question 18, and **PROD** in response to Question 20, the program issues a warning and prompts for the response to be confirmed.

After the response to this question is answered the program displays the complete set of responses for the CSTR and proceeds to Question 4 in Section 9.3.1.

#### 9.3.2.4. CONTROL VALVE

The program creating an instance of a control valve creates a three (3) port device with connections to one (1) upstream and one (1) downstream unit. The valve receives an input signal from a controller. There can be as many sensors connected to the valve as there are variables in the valve. Under nominal conditions, assumptions governing control valve models created by the program are:

- 1) The valve contains a subset of the chemical species present in the process.
- 2) No chemical reaction occurs in the valve
- 3) The fluid in the valve is incompressible with constant density.
- 4) The valve is adiabatic.
- 5) Valve fluid temperature and pressure are greater than respective ambient conditions.

The user can select from among the following list of root causes for the particular instance of a control valve:

- 1) Valve blockage.
- 2) Valve stuck open.
- 3) Valve stuck closed.
- 4) Valve outlet leak.
- 5) Loss of insulation.
- 6) Fire.

The questions asked for a control valve are described below.

1) Enter the chemical species present in cval-name >

The list of species present in the process may be displayed. A list of the species present in the valve is entered. The program verifies that each species just entered is present in the process. Note: if NIL was entered in response to Question 3 in Section 9.2.2.1., this question is not asked.

2) Enter the ID of the controller actuating cval-name >

Enter a unit ID number, U-#.

3) Enter [open/close] if increasing the signal from controller (ID of actuating controller) causes the valve to open or close >

If the control valve opens (hence increasing the flow) on increasing the value of the controller output signal, enter OPEN. If the valve closes, enter CLOSE.

4) Enter the ID of the unit upstream of cval-name >

Enter NONE or a unit ID number, U-#. If no unit exists upstream of the valve, NONE is entered and the program subsequently creates a FEED stream upstream of the valve. Otherwise, the ID of the unit upstream of the valve is entered.

5) Enter [yes/no] if {ID # of upstream unit} has multiple outlets >

If the unit upstream of the valve has more than one outlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous

question indicating that the upstream unit is a FEED stream, this question is not asked and its answer defaults to NO.

6) Enter the ID of the unit downstream of cval-name >

Enter **NONE** or a unit ID number, U-#. If no unit exists downstream of the valve, NONE is entered and the program subsequently creates an EFFLUENT stream downstream of the valve. Otherwise, the ID of the unit downstream of the valve is entered.

7) Enter [yes/no] if {ID # of downstream unit} has multiple inlets >

If the unit downstream of the valve has more than one inlet flow port, enter **YES**. Otherwise, enter **NO**. Note: if NONE was entered in response to the previous question indicating that the downstream unit is an EFFLUENT stream, this question is not asked and its answer defaults to NO.

After the response to this question is answered the program displays the complete set of responses for the control valve and proceeds to Question 4 in Section 9.3.1.

### 9.3.2.5. EFFLUENT STREAM

An effluent stream represents a sink for material flowing from the process. The program creating an instance of an effluent stream creates a one (1) port device with connections to one (1) upstream unit. There can be as many sensors connected to the effluent stream as there are variables in the stream. Unlike other units, the only question asked for effluent streams pertain to selecting its root causes. Other instances of SDG objects for the stream are determined automatically by the program and depend on earlier responses entered for the stream's upstream unit. Under nominal conditions, assumptions governing effluent stream models created by the program are:

- 1) The effluent stream contains the set of chemical species present in its upstream unit.
- 2) A constant density incompressible fluid is leaves the process via the effluent stream.
- 3) The pressure in the effluent stream is less than in its upstream unit.

The user can select from among the following list of root causes for the particular instance of an effluent stream:

- 1) High effluent sink pressure.
- 2) Low effluent sink pressure.

#### 9.3.2.6. FEED STREAM

A feed stream represents a source of material flowing from the process. The program creating an instance of a feed stream creates a one (1) port device with connections to one (1) downstream unit. There can be as many sensors connected to the feed stream as there are variables in the stream. As in effluent streams, the only question asked for feed streams pertain to selecting its root causes. Other instances of SDG objects for the stream are determined automatically by the program and depend on earlier responses entered for the stream's downstream unit. Under nominal fault-free conditions, assumptions governing feed stream models created by the program are:

- 1) The feed stream contains the set of chemical species present in its downstream unit.
- 2) A constant density incompressible fluid is enters the process via the feed stream.
- 3) The pressure in the feed stream is greater than in its downstream unit.

The user can select from among the following list of root causes for the particular instance of a feed stream:

- 1) High concentration of each chemical specie in the stream.
- 2) Low concentration of each chemical specie in the stream.
- 3) High temperature of the fluid in the stream.
- 4) Low temperature of the fluid in the stream.
- 5) High feed source pressure.
- 6) Low feed source pressure.

### 9.3.2.7. CONTINUOUS STIRRED TANK REACTOR WITH JACKET

The program creating an instance of a jacketed continuous stirred tank reactor creates a multi-port device. The jacket is connected to one (1) upstream and one (1) downstream unit. The reactor has a capacity for material storage and can also have possible connections to multiple upstream and downstream units. A generalized set of catalyzed liquid phase chemical reactions take place in the reactor. Currently the maximum number of both catalysts and independent reactions in the network is 5. To increase these limits, see Section 9.3.3.1. There can be as many sensors connected to the jacketed CSTR as there are variables. Under nominal conditions, assumptions governing jacketed CSTR models created by the program are:

- 1) The reactor and the jacket each contain a subset of the chemical species present in the process.
- 2) No chemical reaction occurs in the jacket.
- 3) The fluids in the reactor and jacket are incompressible with constant density.
- 4) The contents of the reactor are well mixed and the concentration of all species in each outlet stream are equal to their concentrations in the reactor.
- 5) Liquid (fluid) temperatures in each outlet stream of the reactor are equal to the reactor temperature.
- 6) Both the reactor and jacket temperatures and pressures are greater than respective ambient conditions.
- 7) The jacketed CSTR is adiabatic.
- 8) The pressure of the fluid in the jacket is greater than the pressure of the reactor fluid
- 9) In sealed (non-vented) jacketed CSTR's, compression (expansion) in the vapor space above the reaction liquid is isothermal.
- 10) Catalyst activity in the reactor is independent of specie concentrations.

The user can select from among the following list of root causes for the particular instance of a CSTR:

- 1) Blockage in each inlet port of the reactor.
- 2) Blockage in each outlet port of the reactor.
- 3) Leak in each outlet port of the reactor.

- 4) Reactor leak.
- 5) Loss of activity for each catalyst.
- 6) Inadequate mixing in reactor.
- 7) Jacket fouling.
- 8) Jacket leak to reactor.
- 9) Jacket leak to environment
- 9) Jacketed CSTR loss of insulation.
- 10) Jacketed CSTR fire.

The questions asked for a jacketed CSTR are described below.

Questions 1 - 5 are asked for the reactor jacket.

1) Enter the chemical species present in the jacket of jcstr-name >

The list of species present in the process may be displayed. A list of the species present in the jacket is entered. The program verifies that each species just entered is present in the process. Note: if NIL was entered in response to Question 3 in Section 9.2.2.1., this question is not asked.

2) Enter the ID of the unit upstream of the jacket of jcstr-name >

Enter **NONE** or a unit ID number, **U-#**. If no unit exists upstream of the tubes, **NONE** is entered and the program subsequently creates a FEED stream upstream of the jacket. Otherwise, the ID of the unit upstream of the jacket is entered.

3) Enter [yes/no] if {ID # of unit upstream of jacket} has multiple outlets >

If the unit upstream of the jacket has more than one outlet flow port, enter **YES**. Otherwise, enter **NO**. Note: if **NONE** was entered in response to the previous question indicating that the upstream unit is a FEED stream, this question is not asked and its answer defaults to **NO**.

4) Enter the ID of the unit downstream of the jacket of jcstr-name >

Enter **NONE** or a unit ID number, **U-#**. If no unit exists downstream of the jacket, **NONE** is entered and the program subsequently creates an **EFFLUENT** stream downstream of the jacket. Otherwise, the ID of the unit downstream of the tubes is entered.

5) Enter [yes/no] if {ID # of unit downstream of jacket} has multiple inlets >

If the unit downstream of the jacket has more than one inlet flow port, enter **YES**. Otherwise, enter **NO**. Note: if **NONE** was entered in response to the previous question indicating that the downstream unit is an **EFFLUENT** stream, this question is not asked and its answer defaults to **NO**.

Questions 6-26 are asked for the reactor.

6) Enter [yes/no] if the reactor of jcstr-name is vented >

If the reactor is open to the surroundings, enter **YES**. Otherwise, enter **NO**.

7) Enter the chemical species at the outlets of the reactor of jcstr-name >

If no chemical species are present in the process a warning message is displayed. The list of species present in the process may be displayed. A list of the species present in all outlet ports of the reactor is entered. The program verifies that each species just entered is present in the process. Note: if **NIL** was entered in response to Question 3 in Section 9.2.2.1., this question is not asked.

8) Enter the number of units upstream of the reactor of jcstr-name >

The **number** of units upstream of the reactor (including feed streams) is entered. The program verifies that the response is a positive integer.

Questions 9-14 are asked for each upstream unit of the reactor.

9) Enter the ID of the next unit upstream of the reactor of jcstr-name >

Enter **NONE** or a unit ID number, **U-#**. If no unit exists upstream of the reactor, **NONE** is entered and the program subsequently creates a FEED stream upstream of the reactor. Otherwise, the ID of the next unit upstream of the reactor is entered. Note: The ID's of all other upstream units must be entered in sequence before specifying FEED streams.

- 10) Enter [yes/no] if the inlet from {ID # of this upstream unit} is below the reactor liquid level of jcstr-name >

If the inlet is below the reactor liquid level of jcstr-name, enter **YES**. Otherwise, enter **NO**.

- 11) Enter [yes/no] if {ID # of this upstream unit} has multiple outlets >

If the next unit upstream of the reactor has more than one outlet flow port, enter **YES**. Otherwise, enter **NO**. Note: if **NONE** was entered in response to the previous question indicating that the upstream unit is a FEED stream, this question is not asked and the answer defaults to **NO**.

- 12) Enter the relationship of the outlet temperature of {ID # of this upstream unit} to the outlet reactor temperature of the jcstr-name [>, = or <] >

If the temperature of the inlet stream from the upstream unit is greater than the temperature of the reactor outlet streams, enter **>**. If the temperatures are equal, enter **=**. Else, enter **<**.

Question 13 and 14 are asked for each species in the reactor of jcstr-name.

- 13) Enter [yes/no] if the {next species in the reactor of jcstr-name} is present in {ID # of this upstream unit} >

If the displayed chemical species is present in the upstream unit, enter **YES**. Otherwise, enter **NO**.

- 14) Enter the relationship of the outlet concentration of {the next species} in {ID # of this upstream unit} to its reactor outlet concentration in jcstr-name [>, = or <] >

If the concentration of the displayed specie in the inlet stream from the upstream unit is greater than its concentration in the reactor outlet streams, enter >. If the concentrations are equal, enter =. Else, enter <. Note: If NO was entered in response to Question 13, the question is not asked and the answer defaults to <.

- 15) Enter the number of units downstream of the reactor of jcstr-name >

The **number** of units downstream of the the reactor (including effluent streams) is entered. The program verifies that the response is a positive integer.

Questions 16 and 17 are asked for each downstream unit of the reactor.

- 16) Enter the ID of the next unit downstream of the reactor of jcstr-name >

Enter **NONE** or a unit ID number, **U-#**. If no unit exists downstream of the reactor, NONE is entered and the program subsequently creates an **EFFLUENT** stream downstream of the reactor. Otherwise, the ID of the next unit downstream of the reactor is entered. Note: The ID of all other downstream units must be entered in the sequence before specifying **EFFLUENT** streams.

- 17) Enter [yes/no] if {ID # of this downstream unit} has multiple inlets >

If the next unit downstream of the reactor has more than one inlet flow port, enter **YES**. Otherwise, enter **NO**. Note: if NONE was entered in response to the previous question indicating that the downstream unit is an **EFFLUENT** stream, this question is not asked and the answer defaults to **NO**.

- 18) Enter the relationship of the net heat released<sup>21</sup> by all reactions in the reactor of jcstr-name to the net heat removed via the jacket [>, = or <] >

If the net heat released by all reactions in the reactor is greater than the heat removed via the jacket, enter >. If equal, enter =. If less than, enter <.

---

<sup>21</sup>If the net effect of the reactions is to consume heat and the heat supplied by the jacket is greater than the heat consumption, enter >.

Question 19 is asked for each species in the reactor of jcstr-name.

- 19) Enter [react/prod/nil] if the {next specie in the reactor of jcstr-name} is a net product, reactant or neither >

If the net effect of the reaction network is to consume the displayed specie, enter **REACT**. If the specie is produced, enter **PROD**. Otherwise, enter **NIL**.

- 20) Enter the number of independent reactions taking place in the reactor jcstr-name >

The total **number** of independent reactions in the network taking place in the reactor of jcstr-name is entered. The response must be a positive integer not exceeding the maximum number of possible reactions.

- 21) Enter the number of catalysts catalyzing the reaction network in reactor of jcstr-name >

The **number** of catalysts catalyzing the reaction network in the reactor of jcstr-name is entered. The response must be a positive integer not exceeding the maximum number of possible catalysts.

Questions 22 - 25 are asked for each reaction in the network. Question 22 is also asked for each catalyst.

- 22) Enter [yes/no] if {the next catalyst in the reactor of jcstr-name} catalyzes {the next reaction in the network} >

If the displayed catalyst catalyzes the reaction, enter **YES**. Otherwise, enter **NO**.

- 23) Enter [rev/irrev] if {the next reaction} is reversible or irreversible >

If the reaction is reversible, enter **REV**. If the reaction is irreversible, enter **IRREV**.

- 24) Enter [endo/exo/nil] if {the next reaction} is endothermic, exothermic or athermal >

If the reaction consumes heat, enter ENDO. If heat is released, enter EXO. Otherwise, enter NIL.

Questions 25 - 26 are asked for each specie in the reactor of jcstr-name.

- 25) Enter [react/prod/nil] if {the next specie in the reactor of jcstr-name} in {the next reaction} is a reactant, product or neither >

If the displayed specie is consumed by the reaction, enter REACT. If the specie is produced, enter PROD. If the specie is not involved in the reaction, enter NIL.

- 26) Enter [inc/dec/nil] if the rate of {the next reaction} increases, decreases or is unaffected by increasing the concentration of {the next specie in the reactor of jcstr-name} >

If increasing the concentration of the displayed specie increases the rate of the reaction, enter INC. If the rate of the reaction is decreased, enter DEC. Otherwise, enter NIL. Note: If NIL was entered in response to the previous question, this question is not asked, and the answer defaults to NIL. If IRREV was entered in response to Question 23, and PROD in response to Question 25, the program issues a warning and prompts for the response to be confirmed.

- 27) Enter the side of the jacketed CSTR in which the temperature of the fluid is greater [jack/react] >

If the temperature of the reactor fluid is greater than the jacket temperature, enter REACT. Otherwise, enter JACK.

Question 28 is asked for each specie in the reactor.

- 12) Enter the relationship of the reactor concentration {of the next specie} to its jacket concentration [>, = or <] >

If the concentration of the displayed specie is higher in the reactor than in the jacket, enter >. If the reactor concentration is less, enter <. Otherwise, enter =. Note: If the specie is not in the jacket as determined, by the response to Question 1, this question is not asked and the answer defaults to >.

After the response to this question is answered the program displays the complete set of responses for the jacketed CSTR and proceeds to Question 4 in Section 9.3.1.

### 9.3.2.8. JUNCTION

The program creating an instance of a junction creates a multi-port device with possible connections to multiple upstream and downstream units. The model is a generalization of a splitter or mixing junction. There can be as many sensors connected to the junction as there are variables in the junction. Under nominal conditions, assumptions governing junction models created by the program are:

- 1) The junction contains a subset of the chemical species present in the process.
- 2) No chemical reaction occurs in the junction and all species in the inlet streams are in each outlet stream.
- 3) The concentration of all species in each outlet stream are equal.
- 4) The fluid in the junction is incompressible with constant density.
- 5) The junction is adiabatic and fluid temperatures in each outlet stream are equal.
- 6) Junction fluid temperature and pressure are greater than respective ambient conditions.

The user can select from among the following list of root causes for the particular instance of a junction:

- 1) Blockage in each inlet port.
- 2) Blockage in each outlet port.
- 3) Leak in each outlet port.
- 4) Leak at junction.
- 5) Loss of insulation.
- 6) Fire.

The questions asked for a junction are described below.

- 1) Enter the chemical species at outlets of junc-name >

The list of species present in the process can be redisplayed. A list of the species present in all outlet ports is entered. The program verifies that each specie just entered is present in the process. Note: if NIL was entered in response to Question 3 in Section 9.2.2.1., this question is not asked.

2) Enter the number of units upstream of junc-name >

The **number** of units upstream of the junction (including feed streams) is entered. The program verifies that the response is a positive integer.

Questions 3-7 are asked for each upstream unit.

3) Enter the ID of the next unit upstream of junc-name >

Enter NONE or a unit ID number, **U-#**. If no unit exists upstream of the junction, **NONE** is entered and the program subsequently creates a FEED stream upstream of the junction. Otherwise, the ID of the next unit upstream of the junction is entered. Note: The ID of all other upstream units must be entered in the sequence before specifying FEED streams.

4) Enter [yes/no] if {ID # of this upstream unit} has multiple outlets >

If the next unit upstream of the junction has more than one outlet flow port, enter **YES**. Otherwise, enter **NO**. Note: if NONE was entered in response to the previous question indicating that the upstream unit is a FEED stream, this question is not asked and the answer defaults to **NO**.

5) Enter the relationship of the outlet temperature of {ID # of this upstream unit} to the outlet temperature of junc-name [>, = or <] >

If the temperature of the inlet stream from the upstream unit is greater than the temperature of the outlet streams of the junction, enter **>**. If the temperatures are equal, enter **=**. Else, enter **<**. Note: If 1 was entered in response to Question 2, this question is not asked and the answer defaults to **=**.

Question 6 and 7 are asked for each species in the outlets of junc-name.

- 6) Enter [yes/no] if the {next species in junc-name} is present in {ID # of this upstream unit} >

If the displayed chemical species is present in the upstream unit, enter YES. Otherwise, enter NO. Note: If 1 was entered in response to Question 2, this question is not asked and the answer defaults to YES.

- 7) Enter the relationship of the outlet concentration of {the next species} in {ID # of this upstream unit} to its outlet concentration in junc-name [>, = or <] >

If the concentration of the displayed specie in the inlet stream from the upstream unit is greater than its concentration in the outlet streams of the junction, enter >. If the concentrations are equal, enter =. Else, enter <. Note: If 1 was entered in response to Question 2, this question is not asked and the answer defaults to =. If NO was entered in response to Question 6, the question is not asked and the answer defaults to <.

- 8) Enter the number of units downstream of junc-name >

The **number** of units downstream of the junction (including effluent streams) is entered. The program verifies that the response is a positive integer.

Questions 9 and 10 are asked for each downstream unit.

- 9) Enter the ID of the next unit downstream of junc-name >

Enter **NONE** or a unit ID number, **U-#**. If no unit exists downstream of the junction, NONE is entered and the program subsequently creates an EFFLUENT stream downstream of the junction. Otherwise, the ID of the next unit downstream of the junction is entered. Note: The ID of all other downstream units must be entered in the sequence before specifying EFFLUENT streams.

- 10) Enter [yes/no] if {ID # of this downstream unit} has multiple inlets >

If the next unit downstream of the junction has more than one inlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the

previous question indicating that the downstream unit is an EFFLUENT stream, this question is not asked and the answer defaults to NO.

After the response to this question is answered the program displays the complete set of responses for the junction and proceeds to Question 4 in Section 9.3.1.

### 9.3.2.9. MANUAL VALVE

The program creating an instance of a manual valve creates a two (2) port device with connections to one (1) upstream and one (1) downstream unit. There can be as many sensors connected to the valve as there are variables in the valve. Under nominal conditions, assumptions governing manual valve models created by the program are:

- 1) The valve contains a subset of the chemical species present in the process.
- 2) No chemical reaction occurs in the valve
- 3) The fluid in the valve is incompressible with constant density.
- 4) The valve is adiabatic.
- 5) Valve fluid temperature and pressure are greater than respective ambient conditions.

The user can select from among the following list of root causes for the particular instance of a manual valve:

- 1) Valve blockage.
- 2) Valve open.
- 3) Valve closed.
- 4) Valve outlet leak.
- 5) Loss of insulation.
- 6) Fire.

The questions asked for a manual valve are described below.

- 1) Enter the chemical species present in mval-name >

The list of species present in the process may be displayed. A list of the species present in the valve is entered. The program verifies that each specie just entered is

present in the process. Note: if NIL was entered in response to Question 3 in Section 9.2.2.1., this question is not asked.

2) Enter the ID of the unit upstream of mval-name >

Enter NONE or a unit ID number, U-#. If no unit exists upstream of the valve, NONE is entered and the program subsequently creates a FEED stream upstream of the valve. Otherwise, the ID of the unit upstream of the valve is entered.

3) Enter [yes/no] if {ID # of upstream unit} has multiple outlets >

If the unit upstream of the valve has more than one outlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous question indicating that the upstream unit is a FEED stream, this question is not asked and its answer defaults to NO.

4) Enter the ID of the unit downstream of mval-name >

Enter NONE or a unit ID number, U-#. If no unit exists downstream of the valve, NONE is entered and the program subsequently creates an EFFLUENT stream downstream of the valve. Otherwise, the ID of the unit downstream of the valve is entered.

5) Enter [yes/no] if {ID # of downstream unit} has multiple inlets >

If the unit downstream of the valve has more than one inlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous question indicating that the downstream unit is an EFFLUENT stream, this question is not asked and its answer defaults to NO.

After the response to this question is answered, the program displays the complete set of responses for the manual valve and proceeds to Question 4 in Section 9.3.1.

### 9.3.2.10. PIPE

The program creating an instance of a pipe creates a two (2) port device with connections to one (1) upstream and one (1) downstream unit. There can be as many sensors connected to the pipe as there are variables in the pipe. Under nominal conditions, assumptions governing pipe models created by the program are:

- 1) The pipe contains a subset of the chemical species present in the process.
- 2) No chemical reaction occurs in the pipe.
- 3) The fluid in the pipe is incompressible with constant density.
- 4) The pipe is adiabatic.
- 5) Pipe fluid temperature and pressure are greater than respective ambient conditions.

The user can select from among the following list of root causes for the particular instance of a pipe:

- 1) Pipe blockage.
- 2) Pipe outlet leak.
- 3) Loss of insulation.
- 4) Fire.

The questions asked for a pipe are described below.

1) Enter the chemical species present in pipe-name >

The list of species present in the process may be displayed. A list of the species present in the pipe is entered. The program verifies that each specie just entered is present in the process. Note: if NIL was entered in response to Question 3 in Section 9.2.2.1., this question is not asked.

2) Enter the ID of the unit upstream of pipe-name >

Enter **NONE** or a unit ID number, **U-#**. If no unit exists upstream of the pipe, **NONE** is entered and the program subsequently creates a FEED stream upstream of the pipe. Otherwise, the ID of the unit upstream of the pipe is entered.

3) Enter [yes/no] if {ID # of upstream unit} has multiple outlets >

If the unit upstream of the pipe has more than one outlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous question indicating that the upstream unit is a FEED stream, this question is not asked and its answer defaults to NO.

4) Enter the ID of the unit downstream of pipe-name >

Enter NONE or a unit ID number, U-#. If no unit exists downstream of the pipe, NONE is entered and the program subsequently creates an EFFLUENT stream downstream of the pipe. Otherwise, the ID of the unit downstream of the pipe is entered.

5) Enter [yes/no] if {ID # of downstream unit} has multiple inlets >

If the unit downstream of the pipe has more than one inlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous question indicating that the downstream unit is an EFFLUENT stream, this question is not asked and its answer defaults to NO.

After the response to this question is answered the program displays the complete set of responses for the pipe and proceeds to Question 4 in Section 9.3.1.

#### 9.3.2.11. SENSOR

The program creating a sensor instance creates a device with one (1) input port and either none (0) or one (1) output port. The output port is required if the sensor is part of a control loop.

The user can select from among the following list of root causes for the particular instance of a sensor:

- 1) Sensor output signal failed high.
- 2) Sensor output signal failed low.

- 3) Sensor output signal biased high.
- 4) Sensor output signal biased low.

The questions asked for a sensor are described below.

- 1) Enter [yes/no] if the sensor is part of a control loop >

If the sensor is a unit in a control loop, enter YES. Otherwise, enter NO.

- 2) Enter the ID of the controller receiving the output signal from sensor-name >

Enter a unit ID number, U-#. Note: If NO was entered in response to the previous question, this question is not asked.

- 3) Enter [yes/no] if the setpoint of {ID of controller receiving input signal from sensor-name} is computed by another unit >

If the controller is a "slave" (secondary) controller in a cascade loop, enter YES. Otherwise, enter NO. Note: If NO was entered in response to the Question 1, this question is not asked.

- 4) Enter the ID of the unit whose variable is monitored by sensor-name >

The list of all units currently instantiated in the GOLDWORKS environment is displayed. This list can be redisplayed before selecting a unit. To select the desired unit, enter the **number** associated with the unit.

- 5) Enter the variable monitored by the sensor-name >

The list of all variables of the unit monitored by sensor-name is displayed. This list can be redisplayed before selecting a variable. To select the desired variable, enter the **number** associated with the variable.

- 6) Enter [yes/no] if the measurement of {variable monitored by sensor-name} is instantaneous >

If there is no delay associated with sensor-name, enter YES. Otherwise, enter NO.

After the response to this question is answered the program displays the complete set of responses for the sensor and proceeds to Question 4 in Section 9.3.1.

### 9.3.2.12. SHELL AND TUBE HEAT EXCHANGER

The program creating an instance of a shell and tube heat exchanger creates a four (4) port device with the capability for heat transfer. Both shell and tube sides are connected to one (1) upstream and one (1) downstream unit each. There can be as many sensors connected to the heat exchanger as there are variables in the heat exchanger. Under nominal conditions, assumptions governing shell and tube heat exchanger models created by the program are:

- 1) Each side of the heat exchanger contains a subset of the chemical species present in the process.
- 2) No chemical reaction occurs in the heat exchanger.
- 3) The fluids in both sides of the heat exchanger are incompressible with constant density.
- 4) The heat exchanger is adiabatic and may have a cocurrent or countercurrent configuration.
- 5) Heat exchanger fluid temperatures and pressures are greater than respective ambient conditions.
- 6) The pressure of the fluid in the tubes is greater than the pressure of the fluid in the shell.

The user can select from among the following list of root causes for the particular instance of a shell and tube heat exchanger:

- 1) Tube blockage.
- 2) Shell blockage
- 3) Tube outlet leak.
- 4) Shell outlet leak.
- 5) Heat exchanger fouling.
- 6) Loss of insulation.

7) Fire.

The questions asked for a shell and tube heat exchanger are described below.

Questions 1 - 5 are asked for the heat exchanger tubes.

1) Enter the chemical species present in the tubes of sthtx-name >

The list of species present in the process may be displayed. A list of the species present in the tubes is entered. The program verifies that each species just entered is present in the process. Note: if NIL was entered in response to Question 3 in Section 9.2.2.1., this question is not asked.

2) Enter the ID of the unit upstream of the tubes of sthtx-name >

Enter **NONE** or a unit ID number, **U-#**. If no unit exists upstream of the tubes, **NONE** is entered and the program subsequently creates a FEED stream upstream of the tubes. Otherwise, the ID of the unit upstream of the tubes is entered.

3) Enter [yes/no] if {ID # of unit upstream of tubes} has multiple outlets >

If the unit upstream of the tubes has more than one outlet flow port, enter **YES**. Otherwise, enter **NO**. Note: if **NONE** was entered in response to the previous question indicating that the upstream unit is a FEED stream, this question is not asked and its answer defaults to **NO**.

4) Enter the ID of the unit downstream of the tubes of sthtx-name >

Enter **NONE** or a unit ID number, **U-#**. If no unit exists downstream of the tubes, **NONE** is entered and the program subsequently creates an EFFLUENT stream downstream of the tubes. Otherwise, the ID of the unit downstream of the tubes is entered.

5) Enter [yes/no] if {ID # of unit downstream of tubes} has multiple inlets >

If the unit downstream of the tubes has more than one inlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous question indicating that the downstream unit is an EFFLUENT stream, this question is not asked and its answer defaults to NO.

Questions 6 - 10 are asked for the heat exchanger shell.

6) Enter the chemical species present in the shell of sthtx-name >

The list of species present in the process may be displayed. A list of the species present in the shell is entered. The program verifies that each species just entered is present in the process. Note: if NIL was entered in response to Question 3 in Section 9.2.2.1., this question is not asked.

7) Enter the ID of the unit upstream of the shell of sthtx-name >

Enter NONE or a unit ID number, U-#. If no unit exists upstream of the shell, NONE is entered and the program subsequently creates a FEED stream upstream of the shell. Otherwise, the ID of the unit upstream of the shell is entered.

8) Enter [yes/no] if {ID # of unit upstream of shell} has multiple outlets >

If the unit upstream of the shell has more than one outlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous question indicating that the upstream unit is a FEED stream, this question is not asked and its answer defaults to NO.

9) Enter the ID of the unit downstream of the shell of sthtx-name >

Enter NONE or a unit ID number, U-#. If no unit exists downstream of the shell, NONE is entered and the program subsequently creates an EFFLUENT stream downstream of the shell. Otherwise, the ID of the unit downstream of the shell is entered.

10) Enter [yes/no] if {ID # of unit downstream of shell} has multiple inlets >

If the unit downstream of the shell has more than one inlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous question indicating that the downstream unit is an EFFLUENT stream, this question is not asked and its answer defaults to NO.

11) Enter the side of the exchanger in which the fluid temperature is greater [shell/tube] >

If the temperature of the shell side fluid is greater than the tube side temperature, enter SHELL. Otherwise, enter TUBE.

Question 12 is asked for each specie in the shell side of the heat exchanger.

12) Enter the relationship of the shell side concentration {of the next specie} to its tube side concentration [>, = or <] >

If the concentration of the displayed specie is higher in the shell than in the tubes, enter >. If the shell side concentration is less, enter <. Otherwise, enter =. Note: If the specie is not in the tubes as determined, by the response to Question 1, this question is not asked and the answer defaults to >.

After the response to this question is answered the program displays the complete set of responses for the heat exchanger and proceeds to Question 4 in Section 9.3.1.

### 9.3.2.13. TANK

The program creating an instance of a tank creates a multi-port device with storage capacity and possible connections to multiple upstream and downstream units. There can be as many sensors connected to the tank as there are variables in the tank. Under nominal conditions, assumptions governing tank models created by the program are:

- 1) The tank contains a subset of the chemical species present in the process.
- 2) No chemical reaction occurs in the tank and all species in the inlet streams are in each outlet stream.

- 3) The fluid in the tank is incompressible with constant density.
- 4) The contents of the tank are well mixed and the concentration of all species in each outlet stream are equal to their concentrations in the reactor.
- 5) Liquid (fluid) temperatures in each outlet stream are equal to the tank temperature.  
Tank temperature is greater than ambient temperature.
- 6) The tank is adiabatic.
- 7) In sealed (non-vented) tanks, compression (expansion) in the vapor space above the liquid is isothermal.

The user can select from among the following list of root causes for the particular instance of a tank:

- 1) Blockage in each inlet port.
- 2) Blockage in each outlet port.
- 3) Leak in each outlet port.
- 4) Leak in tank.
- 5) Loss of insulation.
- 6) Fire.

The questions asked for a tank are described below.

1) Enter [yes/no] if tank-name is vented >

If the tank is open to the surroundings, enter YES. Otherwise, enter NO.

2) Enter the chemical species at outlets of tank-name >

The list of species present in the process may be displayed. A list of the species present in all outlet ports is entered. The program verifies that each species entered is present in the process. Note: if NIL was entered in response to Question 3 in Section 9.2.2.1., this question is not asked.

3) Enter the number of units upstream of tank-name >

The number of units upstream of the tank (including feed streams) is entered. The program verifies that the response is a positive integer.

Questions 4-9 are asked for each upstream unit.

4) Enter the ID of the next unit upstream of tank-name >

Enter NONE or a unit ID number, U-#. If no unit exists upstream of the tank, NONE is entered and the program subsequently creates a FEED stream upstream of the tank. Otherwise, the ID of the next unit upstream of the tank is entered. Note: The ID of all other upstream units must be entered in the sequence before specifying FEED streams.

5) Enter [yes/no] if the inlet from {ID # of this upstream unit} is below the liquid level of tank-name >

If the inlet is below the liquid level of tank-name, enter YES. Otherwise, enter NO.

6) Enter [yes/no] if {ID # of this upstream unit} has multiple outlets >

If the next unit upstream of the tank has more than one outlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous question indicating that the upstream unit is a FEED stream, this question is not asked and the answer defaults to NO.

7) Enter the relationship of the outlet temperature of {ID # of this upstream unit} to the outlet temperature of tank-name [>, = or <] >

If the temperature of the inlet stream from the upstream unit is greater than the temperature of the outlet streams of the tank, enter >. If the temperatures are equal, enter =. Else, enter <. Note: If 1 was entered in response to Question 3, this question is not asked and the answer defaults to =.

Questions 8 and 9 are asked for each species in the outlets of tank-name.

8) Enter [yes/no] if the {next species in tank-name} is present in {ID # of this upstream unit} >

If the displayed chemical specie is present in the upstream unit, enter YES. Otherwise, enter NO. Note: If 1 was entered in response to Question 3, this question is not asked and the answer defaults to YES.

- 9) Enter the relationship of the outlet concentration of {the next species} in {ID # of this upstream unit} to its outlet concentration in tank-name [>, = or <] >

If the concentration of the displayed specie in the inlet stream from the upstream unit is greater than its concentration in the outlet streams of the tank, enter >. If the concentrations are equal, enter =. Else, enter <. Note: If 1 was entered in response to Question 3, this question is not asked and the answer defaults to =. If NO was entered in response to Question 8, the question is not asked and the answer defaults to <.

- 10) Enter the number of units downstream of tank-name >

The **number** of units downstream of the tank (including effluent streams) is entered. The program verifies that the response is a positive integer.

Questions 11 and 12 are asked for each downstream unit.

- 11) Enter the ID of the next unit downstream of tank-name >

Enter **NONE** or a unit ID number, U#. If no unit exists downstream of the tank, NONE is entered and the program subsequently creates an EFFLUENT stream downstream of the tank. Otherwise, the ID of the next unit downstream of the tank is entered. Note: The ID of all other downstream units must be entered in the sequence before specifying EFFLUENT streams.

- 12) Enter [yes/no] if {ID # of this downstream unit} has multiple inlets >

If the next unit downstream of the tank has more than one inlet flow port, enter YES. Otherwise, enter NO. Note: if NONE was entered in response to the previous question indicating that the downstream unit is an EFFLUENT stream, this question is not asked and the answer defaults to NO.

After the response to this question is answered the program displays the complete set of responses for the tank and proceeds to Question 4 in Section 9.3.1.

### **9.3.3. Making Changes to the Model Library**

The user may wish to enhance the model library by:

- 1) modifying existing programs, or
- 2) writing programs for units not included in the library.

Modifications to the model library require differing amounts of LISP programming experience and effort. The most likely changes that may be made are discussed in this section.

#### **9.3.3.1. INCREASING THE NUMBER OF CHEMICAL SPECIES, REACTIONS AND CATALYSTS**

Currently, the maximum number of species present in the process and the number of reactions and catalysts in the CSTR and jacketed CSTR models is five (5). These limits can be increased by a user with little Lisp programming experience.

Chemical species in the process are denoted by the letters A,B,D,J, and K. C,F,N,P,R,S, and T are disallowed symbols. To add other species to the process, certain procedures in the file CODES10B.LSP need to be modified. To add a new species, Z, to the process knowledge base, make changes<sup>22</sup> to procedures MAKE-VARIABLE-CODE-TABLE, MAKE-ROOT-CAUSE-CODE-TABLE and MAKE-CHEMICAL-CODE-TABLE.

To increase the number of catalysts and the number of reactions in the CSTR and jacketed CSTR models, procedures in files CODES10B.LSP, CSTR10L.LSP and JCSTR10L.LSP need to be modified. To increase the maximum number of reactions and catalysts to Nr and

---

<sup>22</sup>The changes are indicated in the program listing.

Nc, change the following lines in procedures CREATE-A-CSTR and CREATE-A-JCSTR in CSTR10L.LSP and JCSTR10L.LSP.

```
(setq max-rxn 5) ---> (setq max-rxn Nr)
(setq max-cat 5) ---> (setq max-cat Nr)
```

In addition, procedures MAKE-VARIABLE-CODE-TABLE and MAKE-ROOT-CAUSE-CODE-TABLE in CODES10B.LSP are modified as indicated in CODES10B.LSP.

### 9.3.3.2. MODIFYING PROGRAMS CREATING UNIT MODELS

If some of the assumptions governing the SDG model of a particular type of unit (Section 9.3.2) created by the library program are violated, the experienced Lisp programmer may choose to modify the relevant program. Making these changes also requires understanding guidelines in Chapter 4 as well as a detailed understanding of all procedures used in creating the SDG model. Thus instead of modifying the program, it is suggested that the model for the "specialized" unit is entered by using the input language described in Section 9.4.

In addition to making necessary modifications to the appropriate program in the MODELLIB directory, the procedures MAKE-VARIABLE-CODE-TABLE, MAKE-ROOT-CAUSE-CODE-TABLE, MAKE-FUNCTION-CODE-TABLE<sup>23</sup> and MAKE-FUNCTION-SYSTEM-CODE-TABLE in CODES10B.LSP may need to be modified. These modifications should be consistent with other procedures in FRAME10B.LSP, and MUNIT10B.LSP.

### 9.3.3.3. ADDING NEW UNIT MODELS TO THE LIBRARY

An experienced LISP programmer may wish to write programs creating SDG unit models not included in the model library. In addition to adding the new program to the MODELLIB directory, the unit has to be added as indicated to procedure MAKE-UNIT-CODE-TABLE in CODES10B.LSP. A frame defining the unit's class should be added to

---

<sup>23</sup>Currently, systems and functions are not implemented.

FRAME10B.LSP. Modifications may be needed for procedures, MAKE-VARIABLE-CODE-TABLE, MAKE-ROOT-CAUSE-CODE-TABLE, MAKE-FUNCTION-CODE-TABLE and MAKE-FUNCTION-SYSTEM-CODE-TABLE in CODES10B.LSP. These modifications should be consistent with relevant procedures in MUNIT10B.LSP.

## 9.4. Creating Unit SDG Objects Using Batch Files

The other method for adding instances of SDG objects for specialized process units to the existing process SDG is to specify the SDG in batch files using a specialized input language. This method requires no programming effort and is preferred to the methods described in Sections 9.3.3.2 and 9.3.3.3.

This option is selected if the user enters 1 when asked to add another unit to the process (Section 9.2.2.1). The program in MODEL10B is executed and the following questions are asked.

- 1) Enter unit filename or QUIT to go to the next unit >

The **pathname** of the batch file containing the specialized unit model is entered. If **QUIT** is entered, Question 2 is bypassed and the program returns to Question 4 in Section 9.2.2.1.

- 2) Enter [yes/no] if the unit is a sensor >

If the process unit in the named batch file is a sensor, enter **YES**. Otherwise, enter **NO**. The program inputs information using the input language creating instances of SDG objects for the unit. Execution is returned to Question 4 in Section 9.2.2.1.

## **9.4.1. Batch File Input Language**

Batch files for unit SDG models may be written using the Goldworks GMACS editor. To invoke the editor

- 1) From the Menu Interface, click on the GMACS option under the SYSTEM menu item, or
- 2) From Top Level Lisp, use the **Ctrl-E** command.

The input language for entering unit SDG models consists of a set of paragraphs. A paragraph is a group of input data such as data describing the variables in the unit. The first line in each paragraph contains a keyword and is followed by other lines (sentences) containing the input data. Each sentence is a LISP list. Each entry in the file except character strings can be entered using lowercase or uppercase characters. A description of each paragraph (denoted by its keyword) follows.

### **1) BEGIN**

The input file must start with the "begin" paragraph. The paragraph indicates the beginning of the file and contains no sentences.

### **2) SET-UNIT-IDENTIFICATION**

This paragraph is entered immediately after the "begin" paragraph. The paragraph contains one sentence, a list of three (3) elements. The first element of the list is a unique unit ID of the form U-#. The second element is a unique unit name and can be any sequence of letters and numbers separated by hyphens. The third element is the unit documentation string is a LISP string and is intended to be more descriptive than the unit name.

### **3) SET-UPSTREAM-UNITS**

This paragraph is optional and contains one sentence, a list of the unit IDs of all units from which material flows to the given process unit. If there are no units

upstream of the process unit, the paragraph may be omitted or the sentence may be entered as the empty list, NIL.

#### 4) SET-DOWNSTREAM-UNITS

This paragraph is optional and contains one sentence, a list of the unit IDs of all units to which material flows from the given process unit. If there are no units downstream of the process unit, the paragraph may be omitted or the sentence may be entered as the empty list, NIL.

#### 5) SET-INPUT-UNITS

This paragraph is optional and contains one sentence, a list of the unit IDs of all units from which information signals are sent to the given process unit. If the process unit has no input units, the paragraph may be omitted or the sentence may be entered as the empty list, NIL.

#### 6) SET-OUTPUT-UNITS

This paragraph is optional and contains one sentence, a list of the unit IDs of all units to which information signals are sent from the given process unit. If the process unit has no output units, the paragraph may be omitted or the sentence may be entered as the empty list, NIL.

#### 7) SET-UNIT-MALFUNCTIONS

This paragraph is optional. Each sentence is a list of three (3) elements describing a malfunction (root cause) affecting the unit. The first element of the list is a list of two elements; the first, a Lisp string denoting the character code associated with the malfunction, the second is optional<sup>24</sup> and is the ID of the adjacent unit associated

---

<sup>24</sup>The ID of the adjacent unit is used only when two or more malfunctions are denoted by the same character code. The ID must have been entered in 3,4,5, or 6.

with the malfunction. The second element is a description of the malfunction. The third element is the location<sup>25</sup> of the malfunction in the unit.

## 8) SET-UNIT-VARIABLES

Each sentence in this paragraph is a list of five (5) elements describing a variable (or parameter) in the unit. The first element of the list is a list of two elements; the first, a Lisp string denoting the character code associated with the variable, the second is optional and is the ID of the adjacent unit associated with the variable. The second element is a description of the variable. The third element is the location of the variable in the unit. The fourth element is the sign of the self-cycle associated with the variable and its value must be one of +, 0 or -. The fifth element indicates whether the variable is a sensor signal. If the variable is a sensor signal, this element has the value T, otherwise the value is NIL.

## 9) SET-PRIMARY-DEVIATIONS

This paragraph is optional. Each sentence is a list of three (3) elements describing the primary deviation relationship between a unit malfunction and a variable in the unit. The first element of the list is the malfunction, described by a list of two elements; the first, the character code associated with the malfunction and the second (optional), the ID of the adjacent unit associated with the malfunction. The second element of the list is the malfunction's primary deviation variable,<sup>26</sup> described by a list of two elements; the first, the character code associated with the variable and the second (optional), the ID of the adjacent unit associated with the variable. The third element of the list is the deviation direction of the variable given the malfunction and its value must be one of HIGH or LOW.

## 10) SET-SIGNAL-VARIABLE-MALFUNCTIONS

This paragraph is optional. Each sentence is a list of two (2) elements describing the set of malfunctions that lead to a failure of a signal variable to a fixed position.

---

<sup>25</sup>The location must be one of: effluent, feed, inlet, input, internal, outlet, and output. When the ID of an adjacent unit is entered for the malfunction, the location must be one of inlet, input, outlet or output.

<sup>26</sup>The malfunction and the variable must have been entered in 7 and 8.

The first element of the list is the signal variable, described by a list of two elements; the first, the character code associated with the variable and the second (optional), the ID of the adjacent unit associated with the variable. The second element of the list is a list of malfunctions leading to a fixed failure of the signal variable.<sup>27</sup> Each malfunction in this list is described by a list of two elements; the first, the character code associated with the malfunction and the second (optional), the ID of the adjacent unit associated with the malfunction.

#### 11) SET-UNIT-CAUSAL-ARCS

Each sentence in this paragraph is a list of five (5) elements describing a causal arc<sup>28</sup> between two variables in the process unit. The first element of the list is the variable from which the arc initiates, described by a list of two elements; the first, the character code associated with the variable and the second (optional), the ID of the adjacent unit associated with the variable. The second element of the list is the variable at which the arc terminates, described by a list of two elements; the first, the character code and the second (optional), the ID of the adjacent unit. The third element of the list is the sign of the causal arc and its value must be one of + or -. The fourth element is the time delay associated with arc and its value must be either 0 or 1. The fifth element is a list of malfunctions that disable the causal arc (i.e. cause the arc to have a 0 magnitude). Each malfunction in this list is described by a list of two elements; the first, the character code and the second (optional), the ID of the adjacent unit associated with the malfunction.

#### 12) SET-ADJACENT-UNIT-VARIABLES

Each sentence in this paragraph is a list of three (3) elements describing identically equal variables<sup>29</sup> in adjacent units. The first element is a Lisp string denoting the character code associated with the variable. The second is the ID of the adjacent

---

<sup>27</sup>The variable must have been entered as a signal variable in 8, and each malfunction entered in 7.

<sup>28</sup>The initiating variable, terminating variable and disabling malfunctions associated with the arc must have been entered in 7 and 8.

<sup>29</sup>For example, the outlet pressure in an upstream unit is identically equal to the inlet pressure in a downstream unit.

unit. The third (optional) is the ID of the process unit and is used only when two or more variables in the adjacent unit are denoted by the same character code.

### 13) SET-ADJACENT-CAUSAL-ARCS

Each sentence in this paragraph is a list of four (4) elements describing causal arcs<sup>30</sup> that initiate from an identically equal variable in an adjacent unit to a variable in the process unit. The first element in the list is the variable in the adjacent unit from which the arc initiates, described by a list of three elements; the first, the character code, the second, the ID of the adjacent unit and the third (optional), the ID of the process unit. The second element in the list is the variable in the process unit at which the arc terminates, described by a list of two elements; the first, the character code, the second (optional) the ID of the adjacent unit associated with the variable. The third element is a list of malfunctions that disable the causal arc. Each malfunction in this list is described by a list of two elements; the first, the character code and the second (optional), the ID of the adjacent unit associated with the malfunction. The fourth element is optional and is used only in sensors. This element signifies the delay on the arc from the variable in the sensed unit to the variable denoting the sensor value in the sensor. The value is either 0 or 1.

### 14) END

The input file must end with the "end" paragraph. The paragraph indicates the end of the file and contains no sentences.

---

<sup>30</sup>The initiating variable, terminating variable and disabling malfunctions associated with the arc must have been entered in 7, 8 and 12.

#### 9.4.2. Batch Input Files for a Gravity Flow Mixing Tank Process

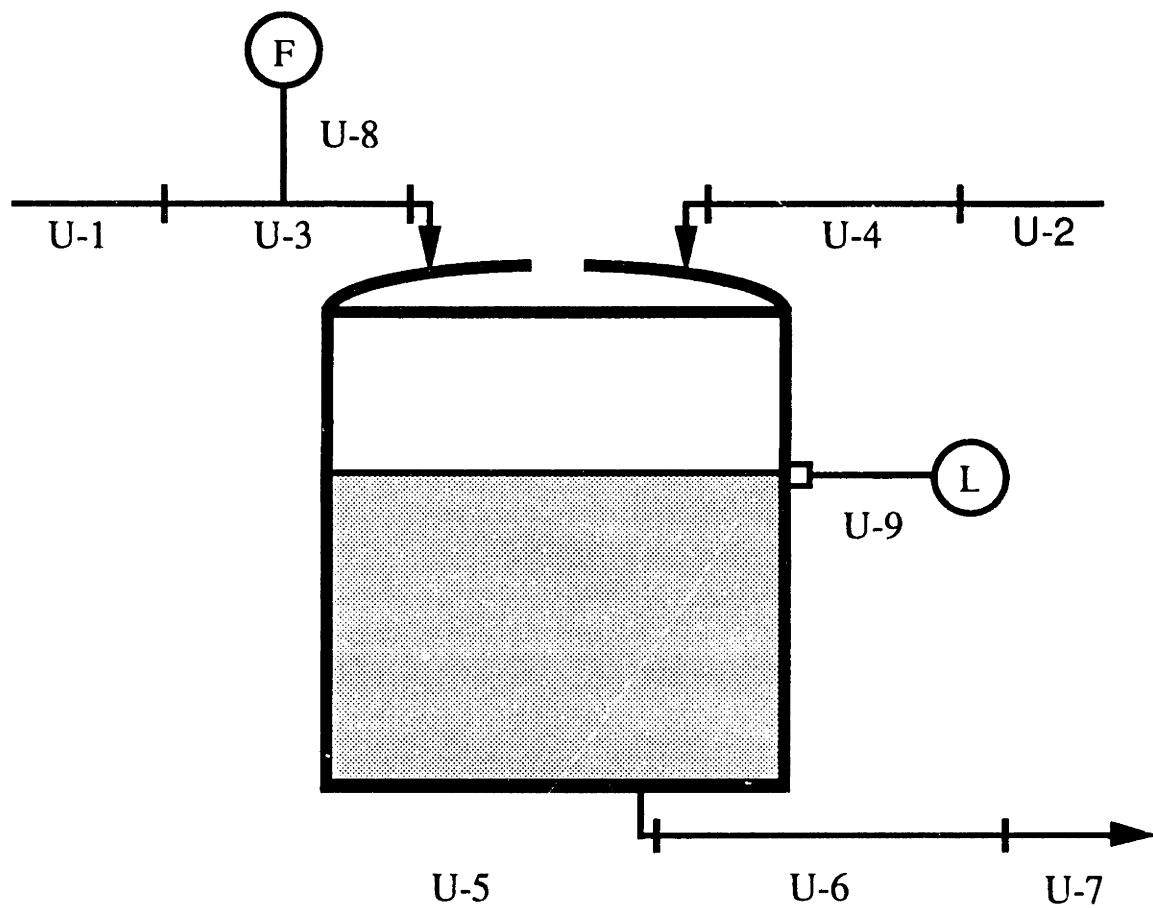


Fig. 9.1. Gravity Flow Mixing Tank

Figure 9.1 shows the process schematic for a gravity flow mixing tank process. IDs and names of the process units are displayed in Table 9.4.

Table 9.4 Units in Gravity Tank Mixing Process

Unit ID	Unit Name
U-1	FEED-TO-PIPE-A
U-2	FEED-TO-PIPE-B
U-3	PIPE-A
U-4	PIPE-B
U-5	TANK
U-6	PIPE-C
U-7	EFFLUENT-FROM-PIPE-C
U-8	F-SENSOR
U-9	L-SENSOR

The SDG of these units and a legend describing the SDG models are shown in Figures 9.2, 9.3, 9.4 and 9.5. Only flow-pressure phenomena are modeled.

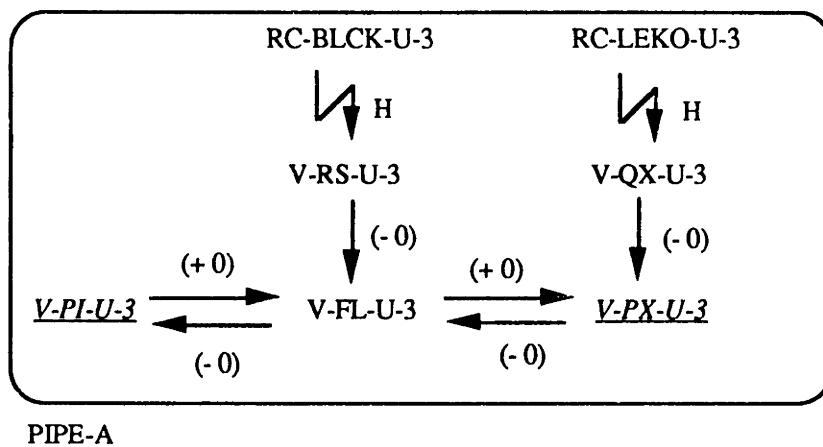


Fig. 9.2. SDG for PIPE-A

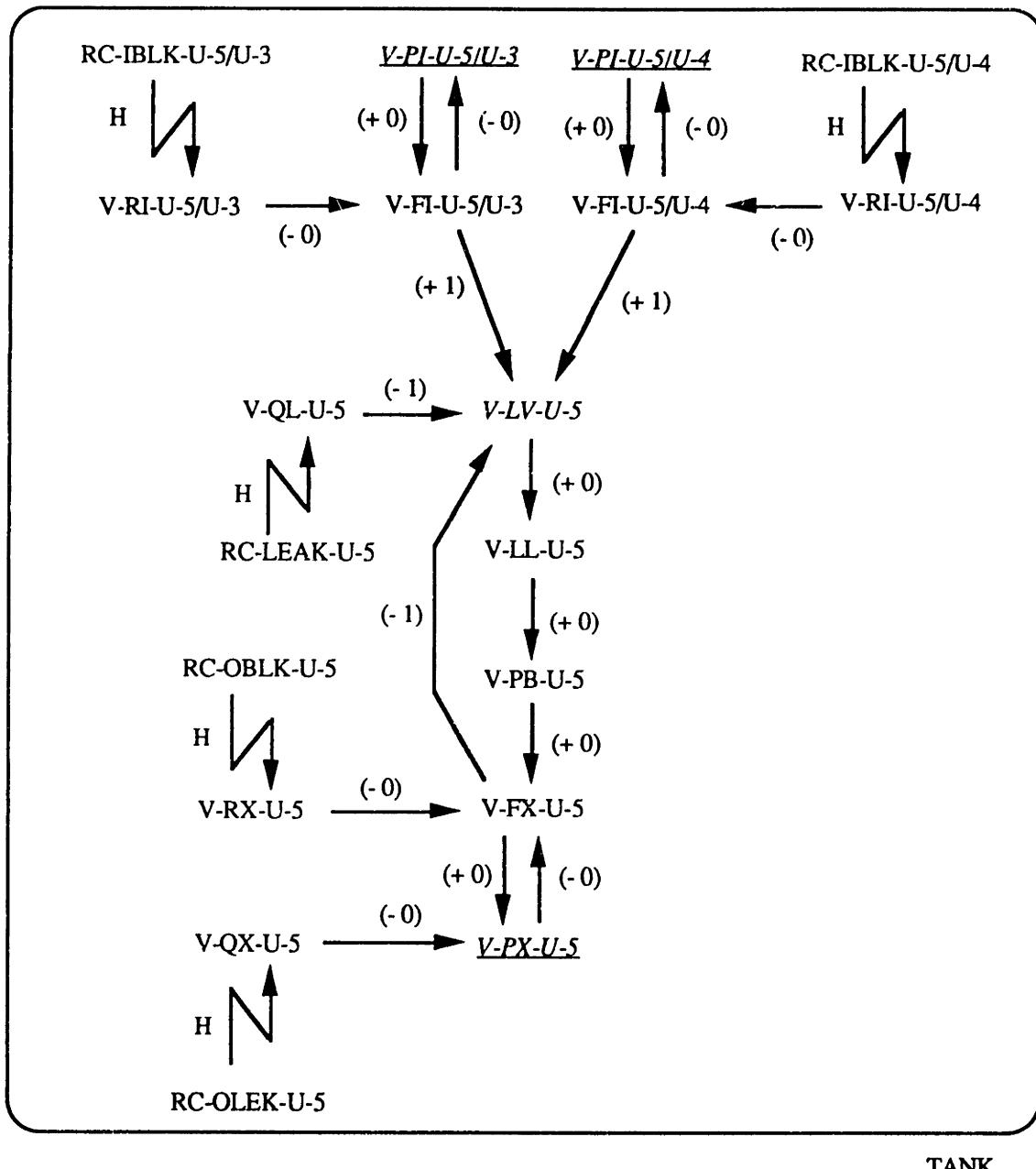


Fig. 9.3. SDG for TANK

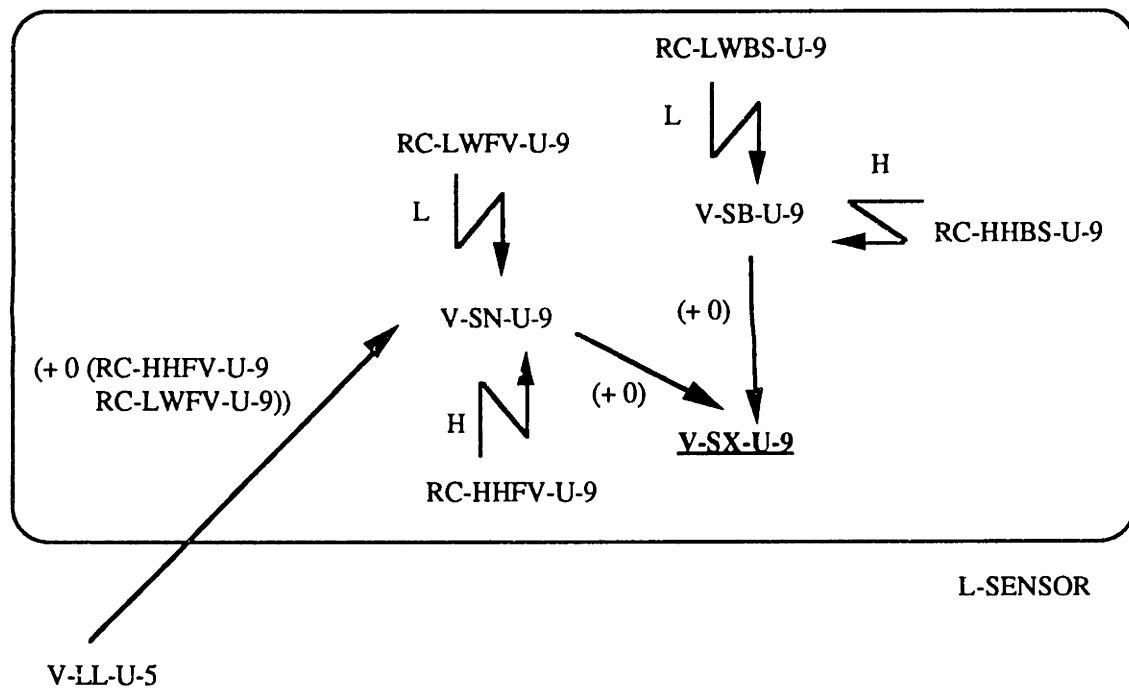


Fig. 9.4. SDG for L-SENSOR

## LEGEND

### CAUSAL ARCS

TRIPLET: 1ST ELEMENT: ARC SIGN  
2ND ELEMENT: TIME DELAY  
3RD ELEMENT (IF EXISTS): DISABLING CONDITIONS

### VARIABLES

PLAIN TEXT (DEFAULT): NEGATIVE SELF-CYCLE  
NON-SIGNAL VARIABLE  
INTERNAL VARIABLE

BOLD: SIGNAL-VARIABLE  
ITALICS: ZERO-SELF-CYCLE  
UNDERLINE: BOUNDARY VARIABLE

### PRIMARY DEVIATION DIRECTION

H: HIGH  
L: LOW

Fig. 9.5. Legend for SDG models

Input files describing the unit SDG models for L-SENSOR, PIPE-A, and TANK are shown below.

#### 9.4.2.1. INPUT FILE FOR PIPE-A

```
BEGIN
SET-UNIT-IDENTIFICATION
(U-3 PIPE-A "Inlet Pipe A")
SET-UPSTREAM-UNIT-IDS
(U-1)
SET-DOWNSTREAM-UNIT-IDS
(U-5)
SET-OUTPUT-UNIT-IDS
(U-8)
SET-UNIT-MALFUNCTIONS
(("BLCK") BLOCKAGE INTERNAL)
(("LEKO") OUTLET LEAK OUTLET)
SET-UNIT-VARIABLES
(("PI") PRESSURE INLET 0 NIL)
(("FL") FLOW INTERNAL - NIL)
(("RS") FLOW_RESISTANCE INTERNAL - NIL)
(("QX") LEAK_FLOWRATE OUTLET - NIL)
(("PX") PRESSURE OUTLET 0 NIL)
SET-PRIMARY-DEVIATIONS
(("BLCK") ("RS") HIGH)
(("LEKO") ("QX") HIGH)
SET-UNIT-CAUSAL-ARCS
(("PI") ("FL") + 0 NIL)
(("FL") ("PI") - 0 NIL)
(("RS") ("FL") - 0 NIL)
(("FL") ("PX") + 0 NIL)
(("QX") ("PX") - 0 NIL)
SET-ADJACENT-UNIT-VARIABLES
("PX" U-1)
("PI" U-5 U-3)
```

```
("SN" U-8)
SET-ADJACENT-CAUSAL-ARCS
(("PX" U-1) ("PI") NIL)
(("PI" U-5 U-3) ("PX") NIL)
END
```

#### 9.4.2.2. INPUT FILE FOR TANK

```
BEGIN
SET-UNIT-IDENTIFICATION
(U-5 TANK "Process Tank")
SET-UPSTREAM-UNIT-IDS
(U-3 U-4)
SET-DOWNSTREAM-UNIT-IDS
(U-6)
SET-OUTPUT-UNIT-IDS
(U-9)
SET-UNIT-MALFUNCTIONS
(("IBLK" U-3) INLET_BLOCKAGE INLET)
(("IBLK" U-4) INLET_BLOCKAGE INLET)
(("LEAK") LEAK INTERNAL)
(("OBLK") OUTLET_BLOCKAGE OUTLET)
(("LEKO") OUTLET_LEAK OUTLET)
SET-UNIT-VARIABLES
(("PI" U-3) PRESSURE INLET 0 NIL)
(("PI" U-4) PRESSURE INLET 0 NIL)
(("FI" U-3) FLOW INLET - NIL)
(("FI" U-4) FLOW INLET - NIL)
(("RI" U-3) FLOW_RESISTANCE INLET - NIL)
(("RI" U-4) FLOW_RESISTANCE INLET - NIL)
(("LV") LIQUID_VOLUME INTERNAL 0 NIL)
(("QL") LEAK_FLOWRATE INTERNAL - NIL)
(("LL") LIQUID_LEVEL INTERNAL - NIL)
(("PB") BASE_PRESSURE INTERNAL - NIL)
(("FX") FLOW OUTLET - NIL)
```

(("RX") FLOW\_RESISTANCE OUTLET - NIL)  
(("QX") LEAK\_FLOWRATE OUTLET - NIL)  
(("PX") PRESSURE OUTLET 0 NIL)  
**SET-PRIMARY-DEVIATIONS**  
(("IBLK" U-3) ("RI" U-3) HIGH)  
(("IBLK" U-4) ("RI" U-4) HIGH)  
(("LEAK") ("QL") HIGH)  
(("OBLK") ("RX") HIGH)  
(("LEKO") ("QX") HIGH)  
**SET-UNIT-CAUSAJ.-ARCS**  
(("PI" U-3) ("FI" U-3) + 0 NIL)  
(("PI" U-4) ("FI" U-4) + 0 NIL)  
(("FI" U-3) ("PI" U-3) - 0 NIL)  
(("FI" U-4) ("PI" U-4) - 0 NIL)  
(("RI" U-3) ("FI" U-3) - 0 NIL)  
(("RI" U-4) ("FI" U-4) - 0 NIL)  
(("FI" U-3) ("LV") + 1 NIL)  
(("FI" U-4) ("LV") + 1 NIL)  
(("QL") ("LV") - 1 NIL)  
(("FX") ("LV") - 1 NIL)  
(("LV") ("LL") + 0 NIL)  
(("LL") ("PB") + 0 NIL)  
(("PB") ("FX") + 0 NIL)  
(("RX") ("FX") - 0 NIL)  
(("FX") ("PX") + 0 NIL)  
(("PX") ("FX") - 0 NIL)  
(("QX") ("PX") - 0 NIL)  
**SET-ADJACENT-UNIT-VARIABLES**  
("PX" U-3)  
("PX" U-4)  
("PI" U-6)  
("SN" U-9)  
**SET-ADJACENT-CAUSAL-ARCS**  
(("PX" U-3) ("PI" U-3) NIL)  
(("PX" U-4) ("PI" U-4) NIL)  
(("PI" U-6) ("PX") NIL)

END

#### 9.4.2.3. INPUT FILE FOR L-SENSOR

```
BEGIN
SET-UNIT-IDENTIFICATION
(U-9 L-SENSOR "Tank Level Sensor")
SET-INPUT-UNIT-IDS
(U-5)
SET-UNIT-MALFUNCTIONS
(("HHBS") BIASED_HIGH INTERNAL)
(("LWBS") BIASED_LOW INTERNAL)
(("HHFV") FAILED_HIGH INTERNAL)
(("LWFV") FAILED_LOW INTERNAL)
SET-UNIT-VARIABLES
(("SN") SENSOR_VALUE INPUT - NIL)
(("SB") SENSOR_BIAS INTERNAL - NIL)
(("SX") SENSOR_SIGNAL OUTPUT - T)
SET-PRIMARY-DEVIATIONS
(("HHBS") ("SB") HIGH)
(("LWBS") ("SB") LOW)
(("HHFV") ("SN") HIGH)
(("LWFV") ("SN") LOW)
SET-SIGNAL-VARIABLE-MALFUNCTIONS
(("SX") ((("HHFV") ("LWFV"))))
SET-UNIT-CAUSAL-ARCS
(("SB") ("SN") + 0 NIL)
(("SN") ("SX") + 0 NIL)
SET-ADJACENT-UNIT-VARIABLES
("LV" U-5)
SET-ADJACENT-CAUSAL-ARCS
(("LV" U-5) ("SN") ((("HHFV") ("LWFV")) 0))
END
```

## **9.5. Building the Process SDG from Batch Files and the Model Library**

For certain processes, the user may build the SDG of the process by combining models of units from the model library with models entered using the batch input language. In building the processs SDG, it must be ensured that the names for identically equivalent variables in adjacent units in models entered using the input language uses the same convention as those from the model library. The names for variables are derived from the units ID, the variables character code and the ID of the variables adjacent unit (Section 9.5).

By convention (Chapter 4), the identically equivalent (boundary) variables in a unit are inlet (outlet) pressures, temperatures, phase compositions and input (output) signals. The character codes used in the model library for these variables are shown in Table 9.5.

Table 9.5 Model Library Character Codes for Boundary Variables

Variable Type	Location	Code
Concentration_A	Inlet	"AI"
Concentration_A	Outlet	"AX"
Concentration_B	Inlet	"BI"
Concentration_B	Outlet	"BX"
Concentration_D	Inlet	"DI"
Concentration_D	Outlet	"DX"
Concentration_J	Inlet	"JI"
Concentration_J	Outlet	"JX"
Concentration_K	Inlet	"KI"
Concentration_K	Outlet	"KX"
Pressure	Inlet	"PI"
Pressure	Outlet	"PX"
Signal	Input	"SI"
Signal	Output	"SX"
Temperature	Inlet	"TI"
Temperature	Outlet	"TX"

## **9.6. Naming Convention for SDG Objects**

The names of SDG objects are derived from

- 1) responses to questions asked by the interactive model library creation programs and
- 2) batch input language files.

The list of slots associated with the objects are described in Chapter 6.

### *Causal\_Arcs*

Causal arcs are numbered sequentially (in the order of their creation). The name of a causal arc instance is of the form SDG-ARC-## where ## is a positive integer.

### *Process\_Units*

The name of a process unit instance is the unit 's ID.<sup>31</sup>

### *Root\_Causes*

The name of a root cause instance is of the form RC-XX-U-##/U-###. XX is the character code for the root cause, U-##, the ID of unit in which the root cause is located and U-### (optional), the ID of the root cause's adjacent unit.

### *Variables*

The name of a variable instance is of the form V-XX-U-##/U-###. XX is the character code for the variable, U-##, the ID of unit in which the variable is located and U-### (optional), the ID of the variable's adjacent unit.

---

<sup>31</sup>The ID of a feed or effluent stream created using the model library is of the form FEED\_TO\_U-#, FEED\_#\_TO\_U-#, EFFL\_TO\_U-#, or EFFL\_#\_TO\_U-#.

## 10. Midas Model Translator

The model translator utility of MIDAS contains programs that derive the Process Model (PM) from the process SDG. The utility consists of nine (9) files in the directory C:\GW\KBS\MIDAS\TRANSLAT. The files and their descriptions are summarized in Table 10.1.

Table 10.1 Translator Utility Files

Name	Description
DEMON10T	PM instance demons
FRAME10T	PM object definitions
LOADF10T	Top level translator utility loading program
MCNTS10T	Constraint and test batch input file reader
MESDG10T	ESDG creation program
MPDMA10T	Qualitative PM translation program (Part A)
MPDMB10T	Qualitative PM translation program (Part B)
MSCRN10T	Screen interface objects creation program
TRMOD10T	Top level translator utility program

In order to conserve memory these files are loaded into the GOLDWORKS environment by the top level translator utility program when needed. A program in the file LOADF10T (Section 10.1) is provided to load the top level utility program.

## **10.1. Loading Top Level Translator Program**

The LOADF10T file contains a program to load the top level translator utility program. To run the program:

- 1) Load Goldworks. If Goldworks has been previously loaded, omit this step.
- 2) From the Goldworks Menu Interface, click on the TOP LEVEL LISP option under the SYSTEM menu item.
- 3a) If the LOADF10T file is in the TRANSLAT directory in the C: device, at the GOLDWORKS prompt, enter:  
**(load "C:\\GW\\KBS\\MIDAS\\TRANSLAT\\LOADF10T")<sup>1</sup>**

or,

- 3b) Set the default directory to the directory containing the LOADF10T file. This can be done at the GOLDWORKS prompt by entering: **(cd pathname)<sup>2</sup>**

and,

- 3b) Then enter: **(load "LOADF10T")**

The utility program will be loaded into the GOLDWORKS environment as indicated by the following message.

"Loading process model translation utility program"

---

<sup>1</sup>The double slash format is used when entering pathnames from TOP LEVEL LISP.

<sup>2</sup>Pathname is a Goldworks pathname. Examples are "C:\\GW\\KBS\\MIDAS\\" for directories and "C:\\GW\\KBS\\MIDAS\\FOO.LSP" for files.

## 10.2. Running a Model Translation Session

The Process Model Translation Utility can be started immediately after loading LOADF10T. The utility analyzes the structure of the SDG, creating the PM in four major steps:

- 1) Derive the ESDG from the SDG.
- 2) Translate the ESDG to qualitative relationships in the PM.
- 3) Supplement qualitative relationships in the PM with relationships between constraints and root causes, and between tests and root causes.
- 4) Create screen interface objects associated with the PM.

The first two stages are mostly

non-interactive but require user input when ambiguities arise during the analysis. The third stage requires the user to specify relationships using an input specification language. The final stage is non-interactive. Each stage outlined above should be completed before proceeding to the next stage. If a subsequent stage has been performed, attempting to create a prior stage may lead to errors. The second and fourth stage are compulsory, while the first and third are optional.

To start a model translation session enter: **(TRANSLATE-MODEL)<sup>3</sup>**

At this point, a banner indicating the MIDAS Process Model Translation Utility is displayed. The response to each question asked by the utility program is echoed and then confirmed by the user. If an invalid response is entered, the utility prompts for a valid response. Details of a typical model translation session are discussed in the remainder of this chapter.

---

<sup>3</sup>This program is in the file, TRMOD10T.

## **10.2.1. Loading Frame Definitions and Demons**

After the banner is displayed, the utility program loads frame definitions for (E)SDG, Screen Toolkit and PM objects as well as demons for PM objects. These objects are loaded only if they do not already exist in the current GOLDWORKS environment.

If definitions for SDG frames do not exist in the current environment, SDG frames will be loaded as indicated by the message:

"Loading SDG object frame definitions"<sup>4</sup>

If definitions for Screen Toolkit frames do not exist, Screen Toolkit frame objects will be loaded as indicated by the message:

"Loading Screen Toolkit"

If definitions for PM frames and demons do not exist, PM frames and demons will be loaded as indicated by the messages:

"Loading PM object frame definitions"

"Loading PM demons"

## **10.2.2. Starting with a Partially Translated Process Model**

After the frame definitions and demons have been loaded, the user can choose to save, delete or load a partially translated PM. By a combination of these actions, model translation may be interrupted and resumed during another session.

If the current GOLDWORKS environment contains object instances for the desired partially translated model and the user does not intend to save object instances:

---

<sup>4</sup>The file C:\GW\KBS\MIDAS\BUILDER\FRAME10B in the builder directory is loaded.

- 1) Enter **NO** in response to all questions in this section.
- 2) Proceed to Section 10.2.3.

### *Save Process Model*

This action saves the model in the GOLDWORKS environment to a file. The action is recommended if the current model has not been saved previously.<sup>5</sup>

The program asks the following.

- 1) Enter [yes/no] if you want to save the existing model >

If object instances are to be saved, enter **YES**. Otherwise, enter **NO**. If entered **YES**, see (2) below. If entered **NO**, (2) is bypassed.

- 2) Enter filename to save process model >

Enter the **pathname** of the file in which the current model in the GOLDWORKS environment should be saved.

All contents of the named file are deleted and all object instances are saved in the named file. If the directory containing the indicated file does not exist an error message is indicated. The program can be aborted<sup>6</sup> by using the Ctrl-C command at the >1 prompt, and the utility started again by entering (**TRANSLATE-MODEL**).

---

<sup>5</sup>If you have to exit the GOLDWORKS environment, instances of the partially translated model should be saved.

<sup>6</sup>Aborting the utility does not delete object instances in the GOLDWORKS environment.

## *Delete Process Model*

This action deletes all object instances in the GOLDWORKS environment. It is impossible to reclaim these objects once deleted, and it is advised that these objects should be saved just before deleting the model.

The action is recommended if object instances are present in the current GOLDWORKS environment and it is intended to resume translating a partial model from a previous session.

The program asks the following.

- 1) Enter [yes/no] if you want to delete the existing model >

If entered YES, all object instances are deleted from the GOLDWORKS environment. If entered NO, the GOLDWORKS environment is not modified.

## *Load Process Model*

This action loads object instances in a partially translated model from a previous session. It is recommended that all instances in the GOLDWORKS environment be deleted just prior to loading the model.

The program asks the following.

- 1) Enter [yes/no] if you want to load the existing model >

If object instances are to be loaded, enter YES. Otherwise, enter NO. If entered YES, see (2) below. If entered NO, (2) is bypassed.

- 2) Enter filename to load process model >

Enter the pathname of the file from which the partially translated model should be loaded.

All object instances in the file are loaded into the GOLDWORKS environment. If the indicated file does not exist an error message is displayed. The program can be aborted by using the Ctrl-C command at the >1 prompt, and the utility started again by entering (TRANSLATE-MODEL).

### 10.2.3. Deriving the ESDG from the SDG

The SDG of a process describes local causality among process variables and root causes. Certain apparent "non-local" causalities resulting from global interactions in negative feedback loops cannot be explained by restricting disturbance propagation to feedforward paths of the SDG. The apparent global causalities result in variables displaying compensatory or inverse response due to negative feedback effects. The behavior of these variables is explained by adding non-physical arcs to the SDG, thus, creating the ESDG. The program in MESDG10T implements the algorithm for creating the ESDG described in Chapter 6. The program creates the ESDG in the following sequence of steps:

- 1) Reduce the size of the SDG, removing variables<sup>7</sup> and arcs that leave local and apparent global causal effects between process measurements unchanged.
- 2) Decompose the SDG into Strongly Connected Components (SCCs).
- 3) Determine variables that may exhibit inverse or compensatory response due to negative feedback in each SCC.
- 4) Create instances of additional ESDG arcs to explain behavior of variables that exhibit inverse or compensatory response due to feedback in each SCC. Additional ESDG arcs are numbered sequentially (in their order of creation). The name of an ESDG arc instance is of the form ESDG-ARC-## where ## is a positive integer.

---

<sup>7</sup>Maximal variables are variables that cannot be affected by any root causes in the SDG. Linear paths and flow-pressure cycles are described in Section 6.5.

If all SDG object instances for the process are in the GOLDWORKS environment, the user may create the ESDG for the process. The program asks the following question.

1) Enter [yes/no] if you want to create the process ESDG >

If the ESDG is to be created enter, YES. Otherwise, enter NO. If entered NO, proceed to Section 10.2.4.

If yes was answered in response to the preceding question, the following actions are performed.

The program that creates the ESDG is loaded as indicated by the message:

"Loading ESDG creation program"

The user sets values for the programs adjustable parameters.

User input is required to determine variables that may exhibit inverse response<sup>8</sup> to negative feedback effects. As a default, the user may specify that no variable exhibits inverse response to negative feedback. The program asks the following question.

2) Enter [yes/no] if any variable in the process exhibits inverse response to negative feedback >

If a variable may exhibit inverse response to negative feedback, enter YES. Otherwise, enter NO. If entered YES, see (3 and 4) below. If entered NO, (3 and 4) are bypassed and their responses default to 0 and NO, respectively.

The size of the SDG is reduced by removing variables and arcs that leave local and apparent global causal effects between process measurements unchanged. Instances of the variables and arcs are removed as indicated by the following messages:

---

<sup>8</sup>User input is required only for variables that satisfy the necessary conditions described in Chapter 6 for exhibiting inverse response to negative feedback. Sufficient conditions relate to numerical parameter values for the particular system.

"Deleting SDG Variable: {variable name} from process knowledge base"

"Deleting: {SDG arc name} from process knowledge base"

"# maximal variables eliminated from SDG"

"Deleting SDG Variable: {variable name} from process knowledge base"

"Deleting: {SDG arc name} from process knowledge base"

"# variables eliminated from linear paths in SDG"

"Deleting SDG Variable: {variable name} from process knowledge base"

"Deleting: {SDG arc name} from process knowledge base"

"# variables eliminated from flow-pressure cycles in SDG"

The SDG is decomposed into its SCC's. The number of SCCs in which apparent global causal effects may occur is indicated by the message:

"The process was decomposed into # relevant SCCs"

Each SCC is analyzed for the presence of apparent global causal effects. Variables that may exhibit inverse or compensatory response due to negative feedback to perturbations in the SCC's disturbance variables are determined. Additional ESDG arcs are created to account for the behavior of these variables. The following actions are performed, messages displayed and questions asked during the analysis.

"Beginning analysis of #th of {total # of relevant SCCs} SCCs: SCC-##"

"Beginning analysis for #th of {total # of disturbance variables in SCC-##} disturbance variables to #th of {total # of relevant SCCs} SCCs: Disturbance Variable {disturbance variable name} to SCC-##"

Variables that exhibit compensatory response due to negative feedback as a result of a perturbation in the disturbance variable do not require user input. User input is required to specify variables that may exhibit inverse response. Depending on the specifics of the situation, one of the following two questions may be asked.

- 3) Enter the set of variables that exhibit inverse response to {disturbance variable name} >

A list of sets of variables in the SCC that may exhibit inverse response due to negative feedback in response to perturbations in the disturbance is displayed. The list can be redisplayed before selecting the desired set of variables. To select the desired set, enter the **number** associated with the list. If no variable exhibits inverse response, enter **0**. Note: if NO was entered in response to Question 2, this question is not asked and the answer defaults to 0.

- 4) Enter [yes/no] if {SCC variable name} exhibits inverse response to {disturbance variable name} >

A variable in the SCC that may exhibit inverse response due to negative feedback in response to perturbations in the disturbance is displayed. If inverse response is exhibited by the variable, enter **YES**. Otherwise, enter **NO**. Note: if NO was entered in response to Question 2, this question is not asked and the answer defaults to NO.

Instances of ESDG arcs are created after analysis for all disturbance variables to the SCC. The following messages are indicated as the arcs are created.

"Creating an ESDG arc from {name of initiating variable} to {name of terminating variable} with arc sign {sign of arc} and delay 1 and disabling conditions {list of root causes that disable the ESDG arc}"

"Completed creating # ESDG arcs for SCC-##. A total of # ESDG arcs created."

"Completed analysis of #th of {total # of relevant SCCs} SCCs: SCC-##"

The following message is displayed after analysis of all SCCs.

"Completed creating process ESDG"

## 10.2.4. Translating the ESDG to the PM

The (E)SDG consists of causal relationships between process malfunctions and both measured and unmeasured variables. Unmeasured variables are not required for diagnosis and are removed from the (E)SDG, resulting in qualitative relationships expressed in the PM. The programs in MPDMA10T and MPDMB10T implement the algorithm for translating the (E)SDG to the PM described in Chapter 6. The programs create the PM in the following sequence of steps:

- 1) Create instances of potential root causes<sup>9</sup> and measured variables<sup>10</sup>.
- 2) For each potential root cause, determine states of measured variables that may be attributed to the potential root cause.
- 3) Instantiate primary deviation relationships for all potential root causes.
- 4) Instantiate successor links between measured variable events for all sensor signals. Associated diagnostic conditions deriving from local relationships in the (E)SDG are also instantiated.
- 5) Instantiate diagnostic conditions associated with successor links deriving from global relationships in the (E)SDG for all measured variables.
- 6) Instantiate diagnostic conditions associated with intravariable links for all measured variables.
- (7) Instantiate sensor failure links for all measured variables.

---

<sup>9</sup>Potential root causes in the PM correspond to root causes in the (E)SDG.

<sup>10</sup>Measured variables in the PM correspond to sensor signals in the (E)SDG.

If (E)SDG object instances for the process are in the GOLDWORKS environment,<sup>11</sup> the user may translate the (E)SDG to qualitative relationships in the PM. The program asks the following question.

1) Enter [yes/no] if you want to create the PM >

If the PM is to be created enter, YES. Otherwise, enter NO. If entered NO, proceed to Section 10.2.5.

If yes was answered in response to the preceding question, the following actions are performed.

The program that translates the (E)SDG into qualitative relationships in the PM is loaded as indicated by the message:

"Loading PM translation program"

The user then sets values for the programs adjustable parameters.

The search depth is specified. The search depth is the maximum number of SCCs in the (E)SDG containing measured variables over which search (initiating from a potential root cause) takes place when determining possible measured variable states that may be attributed to the potential root cause.

2) Enter the search depth [minimum value 2, suggested value 3]<sup>12</sup> >

The **search depth** is entered. The program verifies that the response is a positive integer greater than one.

---

<sup>11</sup>The algorithm can translate either the SDG or ESDG into qualitative relationships in the PM. However, if the PM is derived from the SDG, behaviors arising from global interactions in negative feedback loops will not be represented in the PM.

<sup>12</sup>The difference between minimum and suggested values is because events may occur out of their expected order. A conservative choice for the search depth is  $2 + M$ , where  $M$  is the maximum number of missing intermediate events in an expected sequence of events.

User input is required to resolve ambiguities<sup>13</sup> that arise from causal paths of opposite signs initiating from root causes to sensor signals. As a default, the user may specify that the ambiguity should not be resolved for all potential root cause/ measured variable pairs. The program asks the following question.

- 3) Enter [yes/no] to resolve ambiguities arising from feedforward paths of opposite sign >

If ambiguities arising from causal paths with opposite sign are to be resolved, enter YES. Otherwise, enter NO. If entered YES see (4) below. If entered NO, (4) is not asked, and the response defaults to DONT-KNOW.

Instances of potential root causes and measured variables are created as indicated by the following messages:

"Completed creating potential root cause instances"

"Completed creating measured variable instances"

" The PM of the process has {total # of potential root causes} potential root causes and {total # of measured variables} measured variables"

States of measured variables that may be attributed to potential root causes are determined. Process specific information is supplied by the user to resolve ambiguities for variables that may attain opposite qualitative states for a potential root cause. The program asks the following question.

- 4) Potential root cause: {potential root cause name} may cause variable: {measured variable name} to attain high and low states. Enter the states the variable may attain [high, low, both or dont-know] >

A measured variable that may attain opposite qualitative states as a result of the potential root cause is displayed. If the named variable can attain only the high state as a result of the malfunction, enter HIGH. If only the low state can be attained, enter LOW. If both high and low states can be attained, enter BOTH. Otherwise,

---

<sup>13</sup>User input is only required for potential root cause/measured variable pairs for which causal paths with opposite signs exist. The maximum length of these paths is determined by the search depth. Process-specific information is required to resolve this ambiguity.

enter DONT-KNOW. Note: If NO was entered in response to Question 3, this question is not asked and the answer defaults to DONT-KNOW.

The program continues by deriving qualitative relationships in the PM. The following messages are displayed indicating progress during derivation of the relationships.

"Completed analysis of primary deviations for #th of {total # of potential root causes} potential root causes: {potential root cause name}"

"Completed analysis of primary deviations for all {total # of potential root causes} potential root causes"

"Completed analysis of diagnostic conditions deriving from local causal relationships for successor links of #th of {total # of measured variables} measured variables: {measured variable name}"

"Completed analysis of diagnostic conditions deriving from local causal relationships for successor links for all {total # of measured variables} measured variables"

"Completed analysis of diagnostic conditions deriving from global causal relationships for successor links of #th of {total # of measured variables} measured variables: {measured variable name}"

"Completed analysis of diagnostic conditions deriving from global causal relationships for successor links for all {total # of measured variables} measured variables"

"Completed analysis of diagnostic conditions associated with intravariable links for all {total # of measured variables} measured variables"

"Completed creating sensor failure links for all {total # of measured variables} measured variables"

"A total of # Compiled Causal Links expressing qualitative relationships created."

"Completed deriving qualitative relationships in the PM"

## **10.2.5. Specifying Measured Constraints and Tests in the PM**

Qualitative relationships in the PM may be supplemented with relationships between (quantitative) measured constraints and potential root causes, and between tests and potential root causes. Constraint relationships improve diagnostic resolution when compared to the diagnosis obtained using qualitative relationships alone. Tests are the results of visual or similar inspections used to confirm or deny the existence of a potential root cause. These relationships are specified using the batch input file reader program in MCNTS10T (Section 10.4).

If qualitative relationships in the PM are in the GOLDWORKS environment, the user may specify relationships between measured constraints and potential root causes, and relationships between tests and potential root causes. The program asks the following question.

1) Enter [yes/no] if you want to specify measured constraints and tests in the PM >

If relationships between measured constraints and potential root causes and/or tests and potential root causes are to be specified, enter YES. Otherwise, enter NO. If entered NO, proceed to Section 10.2.6.

If yes was answered in response to the preceding question, the following action is performed as indicated by the message:

"Loading constraint and test batch input file reader"

The program asks the following question.

2) Enter filename specifying constraints and tests, or QUIT to return to top level program >

The **pathname** of the batch file specifying the relationships<sup>14</sup> between constraints and potential root causes, and between tests and potential root causes is entered. If **QUIT** is entered, the program returns to Question 1 in this section.

After relationships specified in the batch file have been instantiated the following messages are indicated.

"A total of # Compiled Causal Links expressing constraint relationships created."  
"Completed deriving constraint relationships in the PM"

### 10.2.6. Creating Screen Interface Objects

Screen interface objects associated with the PM are displayed as part of the MIDAS user interface. After the PM has been created the user may decide to create these objects. The program asks the following question.

1) Enter [yes/no] if you want to create PM screen interface objects >

If screen interface objects are to be created enter, **YES**. Otherwise, enter **NO**. If entered **NO**, proceed to Section 10.2.7.

If yes was answered in response to the preceding question, the following actions are performed as indicated by messages:

"Loading PM screen interface objects creation program"

"Completed creating instances of screen interface objects"

Note: Currently there is a **program bug** in the GOLDWORKS software and some facets associated with slots of the screen interface objects cannot be saved (using the procedure in Section 10.2.7).

---

<sup>14</sup>The relationships are specified according to the specialized input language described in Section 10.4.

To correct this:

1. Using the Goldworks GMACS editor, load the (user) designated file<sup>15</sup> containing the PM for the process. To invoke the editor
  - a) From the Menu Interface, click on the GMACS option under the SYSTEM menu item, or
  - b) From Top Level Lisp, use the **Ctrl-E** command.
2. Add the line

(ANSWER :WHEN-MODIFIED (ACTIVATE-POPUP))

to the following screen interface instance objects

- a) CHOOSE-TEST
- b) CHOOSE-SYSTEM
- c) CHOOSE-CONSTRAINT
- d) CHOOSE-MONITOR
- e) CHOOSE-POTENTIAL-CAUSE
- f) CHOOSE-LINK
- g) CHOOSE-VARIABLE

3. Add the line

(ANSWER :WHEN-MODIFIED (ACTIVATE-CHOSEN-EVENT))

to the CHOOSE-NEXT-EVENT instance object.

---

<sup>15</sup>The filename may be specified in Section 10.2.7.

## 10.2.7. Saving the PM

### *Delete ESDG Objects*

After screen interface objects have been created (E)SDG object instances may be deleted from the GOLDWORKS environment prior to saving instances of PM objects. It is recommended that these objects are deleted before the file containing the PM is used for diagnosis. Frame object definitions for (E)SDG object instances may not be loaded in the GOLDWORKS environment when loading the PM prior to running a diagnostic session. This results in error messages when loading the PM just prior to diagnosis. Deleting these objects also helps in conserving working memory available for diagnosis. The program asks the following question.

- 1) Enter [yes/no] if you want to delete ESDG objects >

If entered, YES all instances of (E)SDG objects are deleted from the GOLDWORKS environment. If entered NO, the GOLDWORKS environment is not modified.

### *Save Process Model*

This action saves the model in the GOLDWORKS environment to a file. The action is recommended if the current model has not been saved previously.

The program asks the following.

- 1) Enter [yes/no] if you want to save the existing model >

If object instances are to be saved, enter YES. Otherwise, enter NO. If entered YES, see (2) below. If entered NO, (2) is bypassed.

- 2) Enter filename to save process model >

Enter the **pathname** of the file in which the current model in the GOLDWORKS environment should be saved.

All contents of the named file are deleted and all object instances are saved in the named file. If the directory containing the indicated file does not exist an error message is indicated. The program can be aborted by using the Ctrl-C command at the >1 prompt, and the utility started again by entering (**TRANSLATE-MODEL**).

### **10.3. Naming Convention for PM and Screen Interface Objects**

The names of PM and screen interface objects are derived

- 1) from their associated SDG objects, or
- 2) sequentially in the order of creation.

The list of slots associated with the objects are described in Chapter 6.

#### **10.3.1 PM Objects**

The names of PM objects created are

##### *Compiled-Causal-Link*

Complied causal links are numbered sequentially in the order of their creation. The name of a compiled causal link instance is of the form CAUSAL-LINK-## where ## is a positive integer.

##### *Constraint-Monitor*

Constraints monitors are numbered sequentially in the order their associated measured constraint instances are specified by the user.<sup>16</sup> The name of a constraint monitor instance is of the form CONSTRAINT-MONITOR-## where ## is a positive integer.

---

<sup>16</sup>Measured-constraints are specified using the input language described in Section 10.4.

### *Measured-Constraint*

The name of a measured constraint instance is of the form MC-name. The name is specified by using the input specification language described in Section 10.4.

### *Measured-Variable*

The name of a measured variable instance is of the form MV-name. "Name" is the value of the name slot of the measured variable's associated variable in the SDG.

### *Potential-Event*

Potential events are numbered sequentially in the order of their creation. The name of a potential event instance is of the form EV-## where ## is a positive integer.

### *Potential-Root-Cause*

The name of a potential root cause instance is of the form PRC-name. "Name" is the value of the name slot of the potential root cause's associated root-cause in the SDG.

### *Sensor-Monitor*

Sensor monitors are numbered sequentially in the order of their creation. The name of a sensor monitor instance is of the form SENSOR-MONITOR-## where ## is a positive integer.

### *Test*

The name of a test instance is of the form T-name. The name is specified by using the input specification language described in Section 10.4.

## **10.3.2. Screen Interface Objects**

The names of the screen interface objects created are:

Choose-Constraint  
Choose-Link  
Choose-Monitor  
Choose-Next-Event  
Choose-Potential-Cause  
Choose-System  
Choose-Test  
Choose-Variable

## **10.4. Specifying Measured Constraints and Tests Using Batch Files**

Relationships between measured constraints and potential root causes, and tests and potential root causes are specified using a specialized input language. The program in MCNTS10T is executed if the user specifies a batch file name in response to Question 2 in Section 10.2.5.

### **10.4.1. Batch File Input Language**

Batch files for specifying relationships between measured constraints and potential root causes, and tests and potential root causes may be written using the Goldworks GMACS editor. To invoke the editor

- 1) From the Menu Interface, click on the GMACS option under the SYSTEM menu item, or
- 2) From Top Level Lisp, use the **Ctrl-E** command.

The input language for specifying relationships between measured constraints and potential root causes, and tests and potential root causes consists of a set of paragraphs. A

paragraph is a group of input data such as the data specifying all constraints associated with the process. The first line in a paragraph contains a keyword and is followed by other line (sentences) containing the input data. Each sentence is a LISP list.<sup>17</sup> Each entry in the file can be entered using lowercase or uppercase characters. A description of each paragraph (denoted by its keyword) follows.

#### 1) BEGIN

The input file must start with the "begin" paragraph. The paragraph indicates the beginning of the file and contains no sentences.

#### 2) BEGIN-CONSTRAINTS

The section specifying relationships between measured constraints and potential root causes must start with the "begin-constraints" paragraph. The paragraph indicates the beginning of this section and contains no sentences. If no relationships between measured constraints and potential root causes are to be specified, this and other paragraphs specifying constraint relationships may be omitted.

#### 3) SET-NAMES-CONSTRAINTS

This paragraph is optional and if entered, it is entered immediately after the "begin-constraints" paragraph. The paragraph contains one sentence, a list of the "names" of all measured constraints for the process. If no measured constraints are to be specified for the process, the paragraph may be omitted or the sentence may be entered as the empty list, NIL.

#### 4) SET-CONSTRAINT-RELATIONSHIPS

This paragraph is optional. Each sentence is a list of eight (8) elements describing relationships between a measured constraint and potential root causes in the process. The first element of the list is the "name" of the measured constraint.<sup>18</sup>

---

<sup>17</sup>A LISP list of elements E<sub>1</sub>, E<sub>2</sub>, ... E<sub>n</sub> is entered as: (E<sub>1</sub> E<sub>2</sub> ... E<sub>n</sub>).

<sup>18</sup>The measured constraint must have been entered in 3.

The second element is a list of "names" of potential root causes that cause the initial transition associated with the constraint to be from the NORMAL state to HIGH. The third element is a list of "names" of potential root causes mentioned in the second element for which the constraint displays compensatory response. The fourth element is a list of "names" of potential root causes mentioned in the second element for which the constraint displays inverse response. The fifth element is a list of the "names" of potential root causes that cause the initial transition associated with the constraint to be from the NORMAL state to LOW.<sup>19</sup> The sixth element is a list of "names" of potential root causes mentioned in the fifth element for which the constraint displays compensatory response. The seventh element is a list of "names" of potential root causes mentioned in the fifth element for which the constraint displays inverse response. The eighth element is a list of "names" of potential root causes that may result in status transitions of measured variables associated with the constraint. Any of the lists of "names" of potential root causes associated with the constraint could be the empty list, NIL.

## 5) END-CONSTRAINTS

The section of the input file specifying relationships between measured constraints and potential root causes must end with the "end-constraints" paragraph. The paragraph indicates the end of the section and contains no sentences.

## 6) BEGIN-TESTS

The section specifying relationships between tests and potential root causes must start with the "begin-tests" paragraph. The paragraph indicates the beginning of this section and contains no sentences. If no relationships between tests and potential root causes are to be specified, this and other paragraphs specifying test relationships may be omitted.

---

<sup>19</sup>The second and fifth elements must have no potential root causes in common.

## **7) SET-NAMES-TESTS**

This paragraph is optional and if entered, it is entered immediately after the "begin-tests" paragraph. The paragraph contains one sentence, a list of the "names" of all tests for the process. If no tests are to be specified for the process, the paragraph may be omitted or the sentence may be entered as the empty list, NIL.

## **8) SET-TEST-RELATIONSHIPS**

This paragraph is optional. Each sentence is a list of four (4) elements describing relationships between a test and potential root causes in the process. The first element of the list is the "name" of the test.<sup>20</sup> The second element is a list of the "names" of potential root causes whose existence is confirmed by the test. The third element is a list of "names" of potential root causes whose existence is contraindicated by the test.<sup>21</sup> The fourth element is a list of the "names" of other tests that are excluded by the test. Any of the lists of "names" potential root causes or other tests associated with the test could be the empty list, NIL.

## **9) END-TESTS**

The section of the input file specifying relationships between tests and potential root causes must end with the "end-tests" paragraph. The paragraph indicates the end of the section and contains no sentences.

## **10) END**

The input file must end with the "end" paragraph. The paragraph indicates the end of the file and contains no sentences.

---

<sup>20</sup>The test must have been entered in 7.

<sup>21</sup>The second and third elements must have no potential root causes in common.

## 10.4.2. Batch Input File for PM of a Hypothetical Process

Figure 10.1. shows the portion of a PM representing relationships between measured constraints and potential root causes, and between tests and potential root causes. Measured constraints are specified in Fig. 10.1a and tests are specified in Fig. 10.1b.

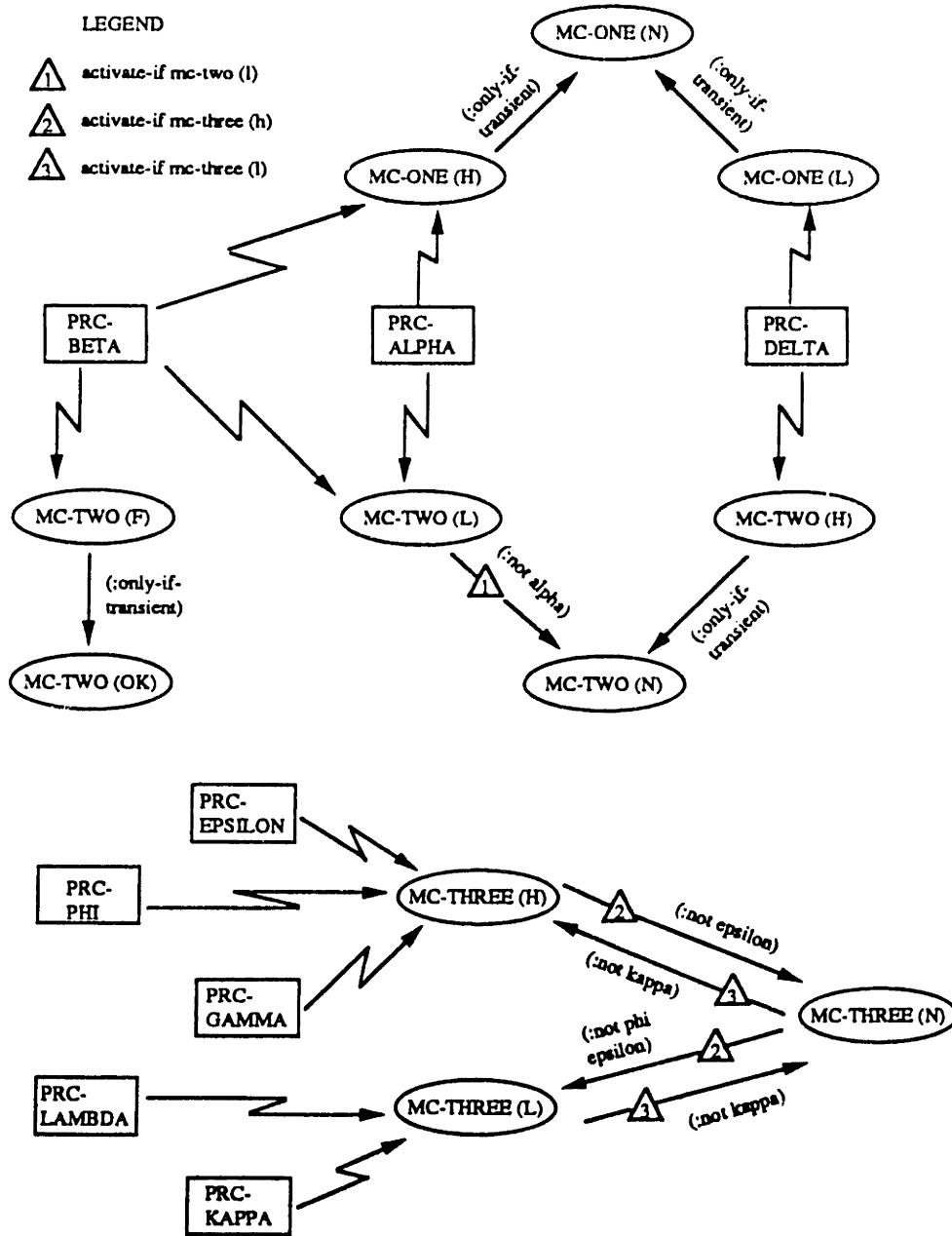


Fig. 10.1a. PM Showing Measured Constraints

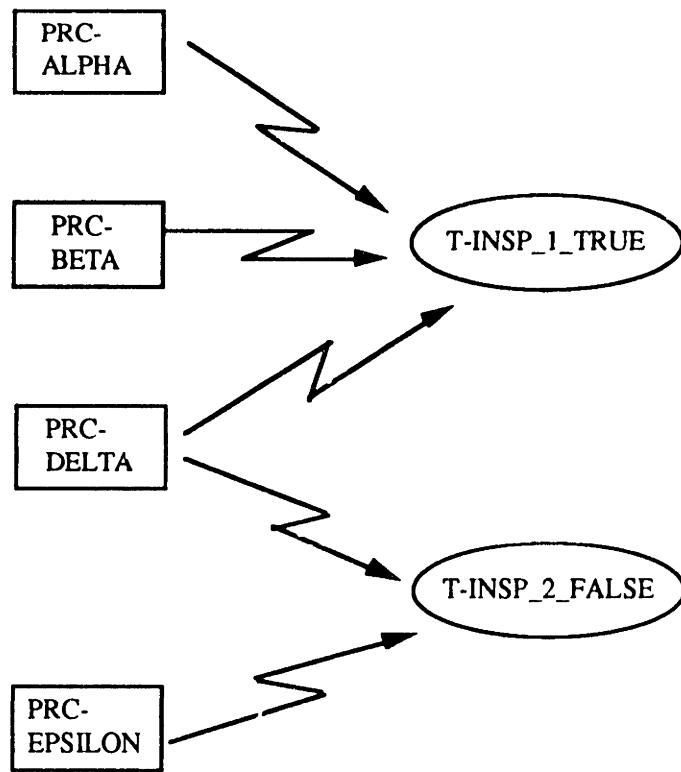


Fig. 10.1b. PM Showing Tests

The input file describing the PM is shown in the next section.

#### 10.4.2.1. INPUT FILE FOR PROCESS PM

```

BEGIN
BEGIN-CONSTRAINTS
SET-NAMES-CONSTRAINTS
(ONE TWO THREE)
SET-CONSTRAINT-RELATIONSHIPS
(ONE (ALPHA BETA) NIL NIL (DELTA) NIL NIL NIL)
(TWO (DELTA) NIL NIL (ALPHA BETA) (BETA) NIL (BETA))
(THREE (EPSILON PHI GAMMA) (PHI) (GAMMA) (KAPPA LAMBDA) NIL
(LAMBDA) NIL)
END-CONSTRAINTS

```

```
BEGIN-TESTS
SET-NAMES-TESTS
(INSP_1_TRUE INSP_1_FALSE INSP_2_TRUE INSP_2_FALSE)
SET-TEST-RELATIONSHIPS
(INSP_1_TRUE (ALPHA BETA DELTA) NIL (INSP_1_FALSE))
(INSP_1_FALSE NIL (ALPHA BETA DELTA) (INSP_1_TRUE))
(INSP_2_TRUE NIL (DELTA EPSILON) (INSP_2_FALSE))
(INSP_2_FALSE (DELTA EPSILON) NIL (INSP_2_TRUE))
END-TESTS
END
```

# Appendix A: Steady State Equations for Heat Exchanger with Bypass

*Local (basic) equations*

$$F_C = k_c(P_{CI} - P_{CO})^{1/2} \quad (A.1a)$$

$$F_1 = k_1(P_{HI} - P_1)^{1/2} \quad (A.1b)$$

$$F_H = k_h(P_1 - P_2)^{1/2} \quad (A.1c)$$

$$F_B = k_b f^+(V)(P_1 - P_2)^{1/2} \quad (A.1d)$$

$$F_2 = k_2(P_2 - P_{HO})^{1/2} \quad (A.1e)$$

$$F_1 = F_H + F_B \quad (A.1f)$$

$$F_2 = F_H + F_B \quad (A.1g)$$

$$\rho C_p F_C (T_5 - T_4) = \rho C_p F_H (T_1 - T_3) \quad (A.1h)$$

$$\rho C_p F_C (T_5 - T_4) = \frac{UA_c((T_1 - T_5) - (T_3 - T_4))}{\ln((T_1 - T_5)/(T_3 - T_4))} \quad (A.1i)$$

$$\rho C_p (F_B T_1 + F_H T_3) = \rho C_p F_2 T_2 \quad (A.1j)$$

*Local (latent) equations*

$$T_3 = (T_1(1 - \alpha) + T_4(e^\gamma - 1))/(e^\gamma - \alpha) \quad (A.2a)$$

$$T_5 = (\alpha T_1(e^\gamma - 1) + e^\gamma T_4(1 - \alpha))/(e^\gamma - \alpha) \quad (A.2b)$$

where

$$\alpha = \rho C_p F_H / \rho C_p F_C ; \alpha \neq 1$$
$$\gamma = UA_C(1 - \alpha) / \rho C_p F_H ; \gamma \neq 0$$

$$\rho C_p F_B(T_1 - T_2) = \rho C_p F_H(T_2 - T_3) \quad (\text{A.2c})$$

*Global (latent) equations*

$$F_1 = F_2 \quad (\text{A.3a})$$

$$\rho C_p F_C(T_5 - T_4) = \rho C_p F_1(T_1 - T_2) \quad (\text{A.3b})$$

$$k_h F_B = k_b f^+(V) F_H \quad (\text{A.3c})$$

## APPENDIX B: Non-Causal Confluences for Continuous Stirred Tank Reactor with Recycle

*Basic confluences*

$$[dF_0] = [dP_{I0}] - [dP_{00}] \quad (B1)$$

$$[dF] = [dP_B] + [dP_p] - [dP_T] \quad (B2)$$

$$[dF_P] = [dV_L] + [dP_T] - [dP_{0P}] \quad (B3)$$

$$[dF_R] = [dV_R] + [dP_T] - [dP_{0R}] \quad (B4)$$

$$[dF_W] = [dV_T] + [dP_{IW}] - [dP_{0W}] \quad (B5)$$

$$[dF] = [dF_R] + [dF_P] \quad (B6)$$

$$[dF] = [dF_O] + [dF_R] \quad (B7)$$

$$[dL] = [dV] \quad (B8)$$

$$[dP_B] = [dL] \quad (B9)$$

$$[dC_{A0}] + [dF_0] + [dF_R] = [dr_A] + [dV] + [dC_A] + [dF]; F > F_R \quad (B10)$$

$$[dF] + [dC_B] = [dr_A] + [dV] + [dF_R] \quad ; F > F_R \quad (B11)$$

$$[dF] + [dT] = [dF_0] + [dT_0] + [dF_R] + [dT_R] + [dr_A] + [dV] \quad (B12)$$

$$[dr_A] = [dk_r] + [dC_A] \quad (B13)$$

$$[dk_T] = [dT] \quad (B14)$$

$$[dV] = [dF_0] + [d\tau] \quad (B15)$$

$$[dP_p] + [dF] = 0 \quad (B16)$$

$$[dF_R] + [dT] + [dT_{WI}] = [dF_W] + [dT_{WO}] + [dT_R] \quad (B17)$$

$$[dL_M] = [dL] \quad (B18)$$

$[dL_M] = [dL_S]$  : or  $[dC_L] = [dL_M] - [dL_S]$  ;  $[dL_M] \neq [dL_S]$  equivalent to  
 $[dL_M] = [dL_S] + ([dC_L] + [dL_S]) \cdot \delta(\delta([dL_M] - [dL_S]))$  (B19)

$$[dV_L] = [dC_L] \quad (B20)$$

$$[dT_M] = [dT] \quad (B21)$$

$[dT_M] = [dT_S]$  : or  $[dC_T] = [dT_M] - [dT_S]$  ;  $[dT_M] \neq [dT_S]$  equivalent to  
 $[dT_M] = [dT_S] + ([dC_T] + [dT_S]) \cdot \delta(\delta([dT_M] - [dT_S]))$  (B22)

$$[dV_T] = [dC_T] \quad (B23)$$

$$[dF_{RM}] = [dF_R] \quad (B24)$$

$[dF_{RM}] = [dF_S]$  : or  $[dC_R] = [dF_S] - [dF_{RM}]$  ;  $[dF_{RM}] \neq [dF_S]$  equivalent to  
 $[dF_{RM}] = [dF_S] + ([dF_S] - [dC_R]) \cdot \delta(\delta([dF_{RM}] - [dF_S]))$  (B25)

$$[dV_R] = [dC_R] \quad (B26)$$

### *Latent confluences*

$$[dF_O] = [dF_P] \quad (B27)$$

$$[dP_p] + [dP_B] + [dV_R] = [dF] + [dF_R] + [dP_{0R}] \quad (B28)$$

$$[dr_A] = [dT] + [dC_A] \quad (B29)$$

$$[dF_O] + [dC_{AO}] + [dF_R] = [dF] + [dT] + [dC_A] + [dV] ; nA_0 e^{-Ea/RT} V C_A^{n-1} + F > F_R \quad (B30)$$

$$[dF_R] + [dT] + [dV] + [dC_A] = [dF] + [dC_B] ; F > F_R \quad (B31)$$

$$[dF_0] + [dC_{AO}] = [dT] + [dV] + [dC_A] \quad (B32)$$

$$[dF_0] + [dC_B] = [dT] + [dV] + [dC_A] \quad (B33)$$

$$[dF_0] + [dC_{AO}] = [dr_A] + [dV] + [dC_A] \quad (B34)$$

$$[dF_0] + [dC_B] = [dr_A] + [dV] \quad (B35)$$

$$[dC_{AO}] = [dC_A] + [dC_B] \quad (B36)$$

$$[dF_R] + [dT] + [dC_A] = [dF_O] + [dC_{AO}] + [dT_O] + [dT_R] ; T > T_R \quad (B37)$$

$$[dF_R] + [dT] = [dF_O] + [dC_B] + [dT_O] + [dT_R] ; T > T_R \quad (B38)$$

$$[dT_{LM}] + [dT_{WO}] + [dT_{WI}] = [dT] + [dT_R] \quad (B39)$$

$$[dF_W] + [dT_{WO}] = [dU] + [dT_{WI}] + [dT_{LM}] \quad (B40)$$

$$[dF_R] + [dT] = [dU] + [dT_{LM}] + [dT_R] \quad (B41)$$

$$[dH_L] + [dT_{WI}] = [dF_W] + [dT_{WO}] \quad (B42)$$

$$[dH_L] + [dT_R] = [dF_R] + [dT] \quad (B43)$$

$$[dH_L] = [dT_{LM}] + [dU] \quad (B44)$$

$$[dH_L] + [d'f_{WI}] = [dU] + [dF_W] + [dF_R] + [dT] \quad (B45)$$

$$[dH_g] = [dr_A] + [dV] \quad (B46)$$

$$[dH_g] = [dF_0] + [dC_B] \quad (B47)$$

$$[dH_T] + [dT_O] = [dF_O] + [dT] \quad (B48)$$

$$[dH_g] = [dH_L] + [dH_T] \quad (B49)$$

## APPENDIX C: Confluences Derived from ESDG for Continuous Stirred Tank Reactor with Recycle

$$[dC_A] = [dC_{AO}] - [dr_A] - [d\tau] \quad \dagger(C1)$$

$$[dC_B] = [dr_A] + [d\tau] \quad \dagger(C2)$$

$$\begin{aligned} [dC_L] = [dL_M] - [dL_S] &+ \delta([dL_M]) \cdot ([dF_O] + [dF_R] + [dF] + [dP_{OP}] \\ &+ [dV_R] - [dF] - [dF_R] - [dP_{OR}] - [dP_P]) \end{aligned} \quad \dagger(C3)$$

$$[dC_R] = [dF_S] - [dF_{RM}] + \delta([dF_{RM}]) \cdot ([dP_{OR}] - [dP_T]) \quad \dagger(C4)$$

$$\begin{aligned} [dC_T] = [dT_M] - [dT_S] &+ \delta([dT_M]) \cdot ([dF_R] + [dT_{WI}] + [dr_A] + [dT_O] \\ &+ [dV] + [dP_{OW}] - [dU] - [dP_{IW}] \\ &- [dF_O] - [dF_R]) \end{aligned} \quad \dagger(C5)$$

$$[dF] = [dP_B] + [dP_P] - [dP_T] \quad (C6)$$

$$[dF_O] = [dP_{IO}] - [dP_{OO}] \quad (C7)$$

$$[dF_P] = [dV_L] + [dP_T] - [dP_{OP}] \quad (C8)$$

$$[dF_R] = [dV_R] + [dP_T] - [dP_{OR}] \quad (C9)$$

$$[dF_{RM}] = [dF_R] \quad (C10)$$

$$[dF_W] = [dV_T] + [dP_{IW}] - [dP_{OW}] \quad (C11)$$

$$[dk_r] = [dT] \quad (C12)$$

$$[dL] = [dV] \quad (C13)$$

$$[dL_M] = [dL] \quad (C14)$$

$$[dP_B] = [dL] \quad (C15)$$

$$[dP_p] = -[dF] \quad (C16)$$

$$[dP_T] = [dF] - [dF_p] - [dF_R] - \delta([dF_p]) \cdot [dV_L] \quad \dagger(C17)$$

$$[dr_A] = [dC_A] + [dC_{AO}] + [dk_r] \quad \dagger(C18)$$

$$[dT] = [dF_O] + [dV] + [dT_O] + [dT_R] + [dr_A] - [dF_O] - [dF_R] \quad \dagger(C19)$$

$$[dT_M] = [dT] \quad (C20)$$

$$[dT_R] = [dT] + [dT_{WI}] + [dF_R] - [dF_W] - [dU] \quad \dagger(C21)$$

$$[dT_{WO}] = [dT] + [dT_{WI}] + [dU] + [dF_R] - [dF_W] \quad \dagger(C22)$$

$$[d\tau] = [dV] - [dF] - [dF_O] - [dF_R] \quad \dagger(C23)$$

$$[dV] = [dF_O] - [dF] - [dF_R] \quad \dagger(C24)$$

$$[dV_L] = [dC_L] \quad (C25)$$

$$[dV_R] = [dC_R] \quad (C26)$$

$$[dV_T] = [dC_T] \quad (C27)$$

$$\begin{aligned} & [dT] = [dT_O] + [dr_A] + [dV] - [dF_O] - [dF_R] \\ \text{or} \quad & [dT_R] = [dF_R] + [dT_{WI}] - [dF_W] - [dU] \end{aligned} \quad \dagger(C28)$$

$$\begin{aligned} & [dT] = [dT_O] + [dT_R] + [dV] - [dF_O] - [dF_R] \\ \text{or} \quad & [dr_A] = [dC_{AO}] + [dC_A] \end{aligned} \quad \dagger(C29)$$

$$\begin{aligned} & [dT] = [dT_O] + [dV] - [dF_O] - [dF_R] \\ \text{or} \quad & [dT_R] = [dF_R] + [dT_{WI}] - [dF_W] - [dU] \\ \text{or} \quad & [dr_A] = [dC_{AO}] + [dC_A] \end{aligned} \quad \dagger(C30)$$

$$\begin{aligned} & [dV] = [dF_O] \\ \text{or } & [dF] = [dP_P] \\ \text{or } & [dP_T] = -[dF_P] - \delta([dF_P]) \cdot [dV_L] \\ \text{or } & [dF_R] = [dV_R] - [dP_{OR}] \end{aligned} \quad \dagger(C31)$$

$\dagger$  Additional confluences provided by ESDG.

## Appendix D: Proof that Interphase Transport and Reversible Reactions Cannot Result in Inverse Variables

Consider the general non linear differential equations describing interphase transport of energy or chemical species:

$$dx_1/dt = -g(x_1) + f + \dots \quad (D.1a)$$

$$dx_2/dt = -h(x_2) + kf + \dots \quad (D.1b)$$

$$f = f^-(x_1) \cdot f^+(x_2) \quad (D.1c)$$

where  $x_1$  and  $x_2$  are temperature (specie concentration) in both phases, and  $k$  is 1. For reversible reactions, equations with the same structure result. Here,  $k$  is the stoichiometric coefficient.

Eliminating  $f^+$  and  $f^-$ , and deriving SDG arcs results in a positive cycle between  $x_1$  and  $x_2$ . Eliminating  $f^+$  and  $f^-$ , and linearizing in terms of deviation variables, we obtain

$$d(dx_1)/dt = -(a_{11} + f_1) \cdot (dx_1) + f_2 \cdot (dx_2) \quad (D.2a)$$

$$d(dx_2)/dt = k \cdot f_1 \cdot (dx_1) - (a_{22} + k \cdot f_2) \cdot (dx_2) \quad (D.2b)$$

Clearly,  $f_1$  and  $f_2$  are positive quantities. In addition, if there are no positive self-cycles associated with  $x_1$  and  $x_2$ ,  $a_{11}$  and  $a_{22}$  are positive.

The signed determinant of this subsystem (in a larger system of equations) is given by

$$(-1)^2 D = a_{11} \cdot a_{22} + a_{22} \cdot f_1 + k \cdot a_{11} \cdot f_2 \quad (D.3)$$

Clearly, the signed determinant is positive. From eq. (3.24) in Chapter 3, a necessary condition for inverse response due to negative feedback is that a signed determinant of one of the subsystems must be negative. However, from eq. (D3), the signed determinant associated with interphase transport or reversible reactions is positive. Thus positive cycles

in the SDG associated with interphase transport and reversible reactions cannot result in inverse response due to negative feedback.

## APPENDIX E: Attributes of Monitor and Screen Interface Objects

Table E.1 Sensor Monitor Object

Slot Name	Data Type	Comments
ASSOCIATION	symbol	MEASURED-OBJECT associated with SENSOR-MONITOR.
ID	symbol	Name of SENSOR-MONITOR instance.
MONITOR-EVENTS	list	POTENTIAL-EVENT instances associated with change in status of associated MEASURED-OBJECT.
MONITOR-TYPE	symbol	Type of DATA-MONITOR. SENSOR-MONITOR.
STATE-EVENTS	list	POTENTIAL-EVENT instances associated with change in state of associated MEASURED-OBJECT.

Table E.2 Constraint Monitor Object

Slot Name	Data Type	Comments
ASSOCIATION	symbol	MEASURED-OBJECT associated with CONSTRAINT-MONITOR.
ID	symbol	Name of CONSTRAINT-MONITOR instance.
MONITOR-EVENTS	list	POTENTIAL-EVENT instances associated with change in status of associated MEASURED-OBJECT.
MONITOR-TYPE	symbol	Type of DATA-MONITOR. CONSTRAINT-MONITOR.
STATE-EVENTS	list	POTENTIAL-EVENT instances associated with change in state of associated MEASURED-OBJECT.

Table E.3 Screen Interface Popup-choose Object

Slot Name	Data type	Comments
ANSWER	procedure	DEMON activated when object receives new value. Must be (activate-popup) or (activate-chosen-event)
BORDER-COLOR	symbol	Color for border of popup menu. Must be specified keyword. e.g. :blue, :brown, :cyan, :green, :red, etc.
CENTER	symbol	Specifies how menu is centered. Must be one of keywords :x, :y, :x-and-y, :no-centering.
CONTENTS	list	Holds a list of PM objects related to the POPUP-CHOOSE instance. Allows user to inspect attributes of PM object by asserting value of PM object to answer slot.
FORCE-CHOICE	symbol	Prevents user from exiting menu without making a choice. Value is keyword :yes.
INSTRUCTIONS	string	Non selectable text that provides information about menu.
LEFT	integer	Left boundary for menu. Must be between 0 and 78.
TOP	integer	Upper boundary for top of menu. Must be between 0 and 23.

## APPENDIX F: Attributes of (E)SDG Objects

Table F.1 Units Object

Slot Name	Data Type	Comments
ID	symbol	Name of UNITS instance. Must be unique, of form U-#.
NAME	symbol	Unique name identifying unit.
UNIT_ROOT_CAUSES	list	ROOT_CAUSES instances that directly affect VARIABLES in unit.
UNIT_VARIABLES	list	VARIABLES instances in unit.
*OPTIONAL-SLOTS*	symbol or list	Optional slots associated with flowsheet structure. If slot name ends in "s", attributes value is list. Otherwise symbol.

### OPTIONAL SLOTS FOR INSTANCES OF UNITS

DOWNSTREAM_UNIT	EFFLUENT_FROM_UNIT
FEED_TO_UNIT	INPUT_TO_CONTROLLER
INPUT_UNIT	INPUT_UNITS
JACKET_DOWNSTREAM_UNIT	JACKET_UPSTREAM_UNIT
OUTPUT_UNIT	OUTPUT_UNITS
REACTOR_DOWNSTREAM_UNITS	REACTOR_UPSTREAM_UNITS
SET_POINT_UNIT	UPSTREAM_UNITS
VALVE_CONTROLLER	

Table F.2 Root Causes Object

Slot Name	Data Type	Comments
ASSOCIATED_PDM_OBJECT	symbol	POTENTIAL-ROOT-CAUSE associated with root cause.
ASSOCIATED_UNIT	symbol	UNITS object associated with root cause.
DEVIATION_DIRECTION	symbol	Initial deviation of PRIMARY_DEVIATION_VAR. Must be HIGH or LOW.
ID	symbol	Name of ROOT_CAUSES instance.
NAME	symbol	Descriptive name of ROOT_CAUSES instance.
PRIMARY_DEVIATION_SCC	symbol	STRONGLY-CONNECTED-COMPONENT containing primary deviation variable.
PRIMARY_DEVIATION_VAR	symbol	Instance of VARIABLES that is first affected by root cause.
ROOT_CAUSE_CLASS	symbol	Descriptive classification of root cause.
ROOT_CAUSE_TYPE	symbol	Functional classification of root cause.

Table F.3 Variables Object

Slot Name	Data Type	Comments
ASSOCIATED_PDM_OBJECT	symbol	If variable denotes process observation, associated MEASURED-VARIABLE instance.
ASSOCIATED_UNIT	symbol	UNITS instance containing variable.
CAUSES_COMPENSATORY_RESPONSE_OF	list	List of variables in immediately downstream SCC that exhibit CR due to NFB when variable is perturbed.
CAUSES_INVERSE_RESPONSE_OF	list	List of variables in immediately downstream SCC that exhibit IR due to NFB when variable is perturbed.
DISPLAYS_COMPENSATORY_RESPONSE_TO	list	List of variables in immediately upstream SCC that cause CR of variable due to NFB.
DISPLAYS_INVERSE_RESPONSE_TO	list	List of variables in immediately upstream SCC that cause IR of variable due to NFB.
ID	symbol	Name of VARIABLES instance.
INIT_ARCS	list	List of CAUSAL_ARCS that initiate from variable.
LOCATION_IN_UNIT	symbol	Descriptive classification of variables location in ASSOCIATED_UNIT. e.g. feed, effluent, outlet, etc.
LOCATED_IN_SCC	symbol	STRONGLY_CONNECTED_COMPONENT instance to which variable belongs.
ROOT_CAUSES_ON_HIGH	list	ROOT_CAUSES listing variable as PRIMARY_DEVIATION_VARIABLE that cause initial state of variable to be high.
ROOT_CAUSES_ON_LOW	list	ROOT_CAUSES listing variable as PRIMARY_DEVIATION_VARIABLE that cause initial state of variable to be low.
SELF_CYCLE	symbol	Sign of corresponding diagonal element in corresponding system matrix. Must be one of +, 0, -.
SENSOR_SIGNAL	symbol	Indicates if variable is process observation. T or NIL.
SENSOR_SIGNALFAULTS	list	List of ROOT_CAUSES that cause state of variable to have an abnormal fixed value.
TERM_ARCS	list	List of CAUSAL_ARCS that terminate at variable.
VARIABLE_TYPE	symbol	Descriptive classification of variable. e.g., concentration, temperature, resistance.

Table F.4 Causal Arcs Object

Slot Name	Data Type	Comments
ALL_DELTA_VARIABLES	list	For ESDG arcs for CR, variables in SCC bypassed by arc.
ARC_TYPE	symbol	Classification of arc. One of STD, TEMP, ESDG.
DELTA_VARIABLE	symbol	For ESDG arcs for CR, last variable in SCC bypassed by arc.
DISABLING_CONDITIONS	list	List of instances of ROOT_CAUSES that deactivate arc.
ID	symbol	Name of CAUSAL_ARCS instance.
INT_VARIABLE	symbol	VARIABLES instance from which arc initiates.
MAGNITUDE	integer	Strength of arcs effect. One of 0 or 1.
NAME	symbol	Descriptive name of CAUSAL_ARCS instance.
NECESSARY_DISTURBANCE_VARIABLES	list	List of variables in immediately upstream SCC for which ESDG arc is derived.
SIGN	symbol	Direction of causal influence represented by arc. One of +, 0, -.
TERM_VARIABLE	list	VARIABLES instance at which arc terminates.
TIME_DELAY	integer	Qualitative delay of causal influence represented by arc. One of 0 or 1.

#### RULES FOR SPECIFYING ATTRIBUTES OF ESDG ARCS

**Disabling\_conditions:** Set union of disabling conditions of all SDG arcs bypassed by ESDG arc.

**Magnitude:** 1.

**Sign:** Product of signs of all SDG arcs bypassed by ESDG arc.

**Time\_delay:** 1.

Table F.5 Strongly Connected Component Object

Slot Name	Data Type	Comments
DISTURBANCE_VARIABLES	list	VARIABLES not in STRONGLY-CONNECTED-COMPONENT (SCC) from which arcs initiate to variables in the SCC.
ID	symbol	Name of STRONGLY-CONNECTED-COMPONENT instance.
INPUT_SCCS	list	STRONGLY-CONNECTED-COMPONENTS containing disturbance variables of SCC.
OUTPUT_SCCS	list	STRONGLY-CONNECTED-COMPONENT listing the SCC as their input sccs.
SCC_MATRIX	array	Matrix depicting effects of scc root cause and disturbance variables on .cc variables. Indexed by keys on x-pair-list and y-pair-list.
SCC_ROOT_CAUSES	list	ROOT_CAUSES (affecting more than one sensor signal) whose primary deviation variables are located in the SCC.
SCC-VARIABLES	list	VARIABLES located in the SCC.
SCC_X_PAIR_LIST	list	Association list of scc disturbance variables and scc root causes keyed by non-negative integer.
SCC_Y_PAIR_LIST	list	Association list of scc variables keyed by non-negative integer.

## **APPENDIX G: Frame Objects for Pipe Unit Model**

### **G.1. Unit Object**

```
(Define-Instance U-1
  (:Print-Name "PIPE-1"
   :Doc-String "Inlet Pipe"
   :Is Pipes)
  (Id U-1)
  (Name PIPE-1)
  (Upstream_Units FEED)
  (Downstream_Units U-2)
  (Unit_Root_Causes RC-LEKO-U-1 RC-BLCK-U-1)
  (Unit_Variables V-FL-U-1 V-HV-U-1 V-PI-U-1 V-PX-U-1
   V-QX-U-1 V-RS-U-1 V-TI-U-1 V-TX-U-1 V-TT-U-1))
```

### **G.2. Root Cause Objects**

```
(Define-Instance RC-LEKO-U-1
  (:Print-Name "PIPE-1_OUTLET_LEAK"
   :Doc-String "Outlet_Leak In Pipe: Pipe-1"
   :Is Root_Causes)
  (Id RC-LEKO-U-1)
  (Name PIPE-1_OUTLET_LEAK)
  (Root_Cause_Class OUTLET_LEAK)
  (Root_Cause_Type CONTAINMENT)
  (Associated_Unit U-1)
  (Primary_Deviation_Var V-QX-U-1)
  (Deviation_Direction HIGH))
```

```
(Define-Instance RC-BLCK-U-1
  (:Print-Name "PIPE-1_BLOCKAGE"
   :Doc-String "Blockage In Pipe: Pipe-1"
   :Is Root_Causes)
  (Id RC-BLCK-U-1)
  (Name PIPE-1_BLOCKAGE)
  (Root_Cause_Class BLOCKAGE)
  (Root_Cause_Type CONDUCTANCE)
  (Associated_Unit U-1)
  (Primary_Deviation_Var V-RS-U-1)
  (Deviation_Direction HIGH))
```

### G.3. Variable Objects

```
(Define-Instance V-QX-U-1
  (:Print-Name "PIPE-1_OUTLET_LEAK_FLOWRATE"
   :Doc-String "Outlet Leak_Flowrate In Pipe: Pipe-1"
   :Is Variables)
  (Id V-QX-U-1)
  (Name PIPE-1_OUTLET_LEAK_FLOWRATE)
  (Variable_Type LEAK_FLOWRATE)
  (Location_In_Unit OUTLET)
  (Self_Cycle -)
  (Sensor_Signal NIL)
  (Associated_Unit U-1)
  (Root_Causes_On_High RC-LEKO-U-1)
  (Init_Arcs SDG-ARC-2))
```

```
(Define-Instance V-RS-U-1
  (:Print-Name "PIPE-1_INTERNAL_RESISTANCE"
   :Doc-String "Internal Resistance In Pipe: Pipe-1"
   :Is Variables)
  (Id V-RS-U-1)
  (Name PIPE-1_INTERNAL_RESISTANCE)
  (Variable_Type RESISTANCE)
  (Location_In_Unit INTERNAL)
  (Self_Cycle -)
  (Sensor_Signal NIL)
  (Associated_Unit U-1)
  (Root_Causes_On_High RC-BLCK-U-1)
  (Init_Arcs SDG-ARC-1))
```

```
(Define-Instance V-PX-U-1
  (:Print-Name "PIPE-1_OUTLET_PRESSURE"
   :Doc-String "Outlet Pressure In Pipe: Pipe-1"
   :Is Variables)
  (Id V-PX-U-1)
  (Name PIPE-1_OUTLET_PRESSURE)
  (Variable_Type PRESSURE)
  (Location_In_Unit OUTLET)
  (Self_Cycle 0)
  (Sensor_Signal NIL)
  (Associated_Unit U-1)
  (Init_Arcs SDG-ARC-3)
  (Term_Arcs SDG-ARC-4 SDG-ARC-2))
```

```
(Define-Instance V-PI-U-1
  (:Print-Name "PIPE-1_INLET_PRESSURE"
   :Doc-String "Inlet Pressure In Pipe: Pipe-1"
   :Is Variables)
  (Id V-PI-U-1)
  (Name PIPE-1_INLET_PRESSURE)
  (Variable_Type PRESSURE)
  (Location_In_Unit INLET)
  (Self_Cycle 0)
  (Sensor_Signal NIL)
  (Associated_Unit U-1)
  (Init_Arcs SDG-ARC-6)
  (Term_Arcs SDG-ARC-5))
```

```
(Define-Instance V-FL-U-1
  (:Print-Name "PIPE-1_INTERNAL_FLOWRATE"
   :Doc-String "Internal Flowrate In Pipe: Pipe-1"
   :Is Variables)
  (Id V-FL-U-1)
  (Name PIPE-1_INTERNAL_FLOWRATE)
  (Variable_Type FLOWRATE)
  (Location_In_Unit INTERNAL)
  (Self_Cycle -)
  (Sensor_Signal NIL)
  (Associated_Unit U-1)
  (Init_Arcs SDG-ARC-5 SDG-ARC-4)
  (Term_Arcs SDG-ARC-6 SDG-ARC-3 SDG-ARC-1))
```

## G.4. Permanent Causal Arc Objects

```
(Define-Instance SDG-ARC-1
  (:Print-Name "CAUSAL_ARC_1"
   :Doc-String ""
   :Is_Causal_Arcs)
  (Id SDG-ARC-1)
  (Name SDG-ARC-1)
  (Sign -)
  (Magnitude 1)
  (Time_Delay 1)
  (Arc_Type STD)
  (Init_Variable V-RS-U-1)
  (Term_Variable V-FL-U-1))
```

```
(Define-Instance SDG-ARC-2
  (:Print-Name "CAUSAL_ARC_2"
   :Doc-String ""
   :Is_Causal_Arcs)
  (Id SDG-ARC-2)
  (Name SDG-ARC-2)
  (Sign -)
  (Magnitude 1)
  (Time_Delay 1)
  (Arc_Type STD)
  (Init_Variable V-QX-U-1)
  (Term_Variable V-PX-U-1))
```

```
(Define-Instance SDG-ARC-3
  (:Print-Name "CAUSAL_ARC_3"
   :Doc-String ""
   :Is Causal_Arcs)
  (Id SDG-ARC-3)
  (Name SDG-ARC-3)
  (Sign -)
  (Magnitude 1)
  (Time_Delay 1)
  (Arc_Type STD)
  (Init_Variable V-PX-U-1)
  (Term_Variable V-FL-U-1))
```

```
(Define-Instance SDG-ARC-4
  (:Print-Name "CAUSAL_ARC_4"
   :Doc-String ""
   :Is Causal_Arcs)
  (Id SDG-ARC-4)
  (Name SDG-ARC-4)
  (Sign +)
  (Magnitude 1)
  (Time_Delay 1)
  (Arc_Type STD)
  (Init_Variable V-FL-U-1)
  (Term_Variable V-PX-U-1))
```

(Define-Instance SDG-ARC-5  
(:Print-Name "CAUSAL\_ARC\_5"  
:Doc-String ""  
:Is Causal\_Arcs)  
(Id SDG-ARC-5)  
(Name SDG-ARC-5)  
(Sign -)  
(Magnitude 1)  
(Time\_Delay 1)  
(Arc\_Type STD)  
(Init\_Variable V-FL-U-1)  
(Term\_Variable V-PI-U-1))

(Define-Instance SDG-ARC-6  
(:Print-Name "CAUSAL\_ARC\_6"  
:Doc-String ""  
:Is Causal\_Arcs)  
(Id SDG-ARC-6)  
(Name SDG-ARC-6)  
(Sign +)  
(Magnitude 1)  
(Time\_Delay 1)  
(Arc\_Type STD)  
(Init\_Variable V-PI-U-1)  
(Term\_Variable V-FL-U-1))

## G.5. Temporary Causal Arc Objects

```
(Define-Instance SDG-ARC_100002
  (:Print-Name "CAUSAL_ARC_100002"
   :Doc-String " "
   :Is Causal_Arcs)
  (Id SDG-ARC-100002)
  (Name SDG-ARC-100002)
  (Sign +)
  (Magnitude 1)
  (Time_Delay 0)
  (Arc_Type TEMP)
  (Init_Variable V-PI-U-2)
  (Term_Variable V-PX-U-1))
```

```
(Define-Instance SDG-ARC-100001
  (:Print-Name "CAUSAL_ARC_100001"
   :Doc-String " "
   :Is Causal_Arcs)
  (Id SDG-ARC-100001)
  (Name SDG-ARC-100001)
  (Sign +)
  (Magnitude 1)
  (Time_Delay 0)
  (Arc_Type TEMP)
  (Init_Variable V-PX-FEED)
  (Term_Variable V-PI-U-1))
```

## APPENDIX H: Algorithms for Deriving the ESDG

In this appendix, two of the more complicated algorithms used in deriving the ESDG are presented. These algorithms are subroutines of the main algorithm shown in Fig. 6.6. Other subroutines, e.g. finding acyclic paths in a sub graph by depth first search, are more straightforward and are variants of algorithms found in standard texts on graph theory.

The algorithms are expressed in terms of the version of **pidgin algol** used by Papadimitriou and Steiglitz [1]. Pidgin algol is an informal notation, and should not be viewed as a high-level programming language. We begin by introducing the notation of pidgin algol.

### H.1. Pidgin Algol<sup>1</sup>

The basic unit of a pidgin algol algorithm is the *statement*. A statement may be of different kinds.

1. *Assignment:* variable:= expression

We allow things like

$Q := \{s\}$

and even variants like

let  $x$  be any element of  $S$

or

set all labels to 0.

2. *Conditional:* **if** condition **then** statement 1

**else** statement 2

The **else** clause is optional. Typical conditions are

$x > \text{bound}$

$v = s$  **and** done = "yes"

(Note: the reserved words of pidgin algol are written in boldface.)

---

<sup>1</sup>This subsection is essentially taken in its entire form from [1], p. 24.

3. *For Statement:* **for** list **do** statement 1

Here *list* contains a list of parameters for which statement 1 is to be repeated.

Examples are

**for** j := 1,2,...n **do** A[j] := A[j + 1]

and

**for** all v ∈ V such that [v,u] ∈ E **do** rank[v] := 0

4. *While Statement:* **while** condition **do** statement 1

Statement 1 is executed repeatedly, as long as condition holds.

5. *Go-to Statement:* **go to** label

For example **go to** loop means that the (unique) statement prefixed by the label "loop:" should be executed next.

6. Compound Statement:

```
begin
    statement 1;
    statement 2;
    .
    .
    .
    statement k-1;
    statement k;
end
```

In a conditional statement, **then** must be followed by a single statement.

Compound statements are often used if several steps are to be executed if a condition holds. These statements are put together, separated by semicolons, and surrounded with a **begin-end** pair. To make the statements more readable, they are indented appropriately. Statements 1 through k could be of any of the sorts 1 through 6. If they are all assignments or go-to's, a compound statement is sometimes written without the **begin-end** pair, but with the statements separated by commas. For example,

$j := j + 1$ ,  $A[j] := i$ , go to loop  
stands for

```
begin
     $j := j + 1;$ 
     $A[j] := i;$ 
    go to loop
end
```

7. *Comments:* These are of the form (**comment**: this is a comment).
8. *Miscellaneous statements:* Practically almost anything is allowed, as long as it is readable and reasonably ambiguous. Extreme examples are

Construct the auxiliary network  $AN(f)$ .  
Augment the current matching using the path  $p$ .

## H.2. Bipartite Matching Algorithm<sup>2</sup>

The bipartite matching algorithm determines if the qualitative rank of a matrix. A bipartite graph,  $B(V, U, E)$ , is a graph in which its vertices are partitioned into two sets,  $V$  and  $U$ , such that each edge in  $E$  has one vertex in  $V$  and the other in  $U$ . After constructing an equivalent bipartite graph from the SDG (including its self-cycles), the algorithm can be used to determine the maximum matching of the bipartite graph. The bipartite matching algorithm is presented below.

**Input:** A bipartite graph  $B(V, U, E)$ .

**Output:** The maximum matching of  $B$ , represented by the array `mate`.

```
begin
    for all  $v \in V \cup U$  do  $\text{mate}[v] := 0$ ; (comment: initialize)
stage:   begin
            for all  $v \in V$  do  $\text{exposed}[v] := 0$ ;
```

---

<sup>2</sup>This subsection is essentially taken in its entire form from [1], p. 224.

```

A :=  $\emptyset$ ; (comment: begin construction of the auxiliary graph (V,A))
for all  $[v,u] \in E$  do
  if mate[u] = 0 then exposed[v] := u else
    if mate[u]  $\neq v$  then A := A  $\cup$  (v,mate[u]);

Q :=  $\emptyset$ ;
for all  $v \in V$  do if mate[v] = 0 then Q := Q  $\cup$  {v}, label[v] := 0;

while Q  $\neq \emptyset$  do
  begin
    let v be a node in Q;
    remove v from Q;

    if exposed[v]  $\neq 0$  then augment(v), go to stage; else
      for all unlabeled  $v'$  such that  $(v,v') \in A$  do
        label[v'] := v, Q := Q  $\cup$  {v'};

  end
end

procedure augment(v)
  if label[v] = 0 then mate[v] := exposed[v], mate[exposed[v]] := v;
  else begin
    exposed[label[v]] := mate[v];
    mate[v] := exposed[v];
    mate[exposed[v]] := v;
    augment(label[v]);
  end

```

### H.3. Algorithm for Finding Cycles in the SDG

A directed graph, G(V,A) is a graph in which directed arcs A exist between its vertices. A variant of the depth first search (DFS) algorithm is used to find all the cycles in a directed graph. After all cycles have been found, they can be grouped into strongly connected

components. These cycles (including self-cycles) are used to determine if certain structures in the SDG may lead to variables exhibiting IR or CR due to negative feedback.

**Input:** A graph  $G(V,A)$ .

**Output:** Cycles of  $G$ , represented by the list,  $L$ .

```

begin
     $L := \emptyset;$ 
     $U := V$ 
    for all  $u \in U$  do  $\text{visited}[u] := 0$ ; (comment: initialize)

ln1:   if  $U = \emptyset$  then return  $L$  else
        begin
             $T := \emptyset;$ 
             $Q := \emptyset;$ 
            let  $y$  be a variable in  $U$ ;
             $\text{visited}[y] := 1$ ;
            form a path  $p$  containing  $y$  as its only variable;
             $Q := \{p\} \cup Q$ ; (comment: initialize queue for DFS in SCC and
                               downstream SCCs)

ln2:   if  $Q = \emptyset$  then
        begin (comment: all new cycles in SCC containing  $y$  and its downstream
                  SCCs found)
             $T := \text{find-unique-cycles}(T)$ ;
            for all  $u \in U$  do if  $\text{visited}[u] = 1$  then remove  $u$  from  $U$ ;
            (comment: initialize unvisited variables)
             $L := L \cup T$ ; (comment: add new cycles to output list)
            go to ln1;
        end

        let  $p$  be the first member of  $Q$ ;
        remove  $p$  from  $Q$ ;
        let  $w$  be the last variable added to  $p$ ;
        find all children, c of w such that  $c \in U$ ;
    
```

```

NP :=  $\emptyset$ ; (comment: initialize new paths)
for all c do visited[c] := 1, extend p by c to form np, NP:= NP  $\cup$  {np};

AP :=  $\emptyset$ ; (comment: initialize acyclic paths)
for all np  $\in$  NP do if np has no repeated variables then AP:= AP  $\cup$  {np};
for all ap  $\in$  AP do remove ap from NP, Q := {ap}  $\cup$  Q;
(comment: add acyclic paths to front of queue)
for all np  $\in$  NP do T := T  $\cup$  truncate-cycle(np);
(comment: new cycles added to temporary list)

go to ln2; (comment: continue DFS)
end
end

```

*Note*

A cycle with n variables can be represented by a list of n + 1 variables. Thus each cycle may be represented by up to n different permutations, each starting at a different variable. For example, (1→2→3→4→1) and (3→4→1→2→3) are permutations of the same cycle. (4→3→2→1→4), represents another cycle oriented in the opposite direction.

```

procedure find-unique-cycles(L)
(comment: returns unique cycles in output P)
P :=  $\emptyset$ 
ln1: if L  $\neq \emptyset$  then begin
        let l be an element of L;
        remove l from L;
        if l is not a cyclic permutation of any element of P
            then add l to P;
        go to ln1;
end
else return P

```

```

procedure truncate-cycle(c)
begin
    for the cyclic path c remove all additional variables leaving only variables
    necessary to specify the cycle;
    return the cycle;
end

```

For example, if the input to truncate cycle is  $(1 \leftarrow 2 \leftarrow 3 \leftarrow 4 \leftarrow 1 \leftarrow 5 \leftarrow 6)$ , the procedure returns  $(1 \leftarrow 2 \leftarrow 3 \leftarrow 4 \leftarrow 1)$  as its output.

## H.4. References for Appendix H

1. Papadimitriou C. H. and K. Steiglitz (1982). Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall Inc. Englewood Cliffs, N.J.

# Appendix I: Evaluating Qualitative Determinants of Structures with Flow-Pressure Cycles

In Section 6.5.5.1, the size of the SDG was reduced by eliminating variables from cycles resulting from flow-pressure interactions in linear recycle and bypass configurations. In this appendix, we show that the qualitative determinants of these structures are zero before and after eliminating variables.

## I.1. Linear Flow-Pressure Cycles

The qualitative matrices corresponding to sub graphs of the SDG in Figs. 6.9a and 6.9b are

$$\begin{array}{c|ccccc} p_1 & o & - & o & o & o \\ f_1 & + & - & - & o & o \\ p_2 & o & + & o & - & o \\ f_2 & o & o & + & - & - \\ p_3 & o & o & o & + & o \end{array} \quad \text{and} \quad \begin{array}{c|cc} p_1 & o & - & o \\ f_2 & + & - & - \\ p_3 & o & + & o \end{array}$$

respectively. From the definition of the matrix determinant, it is easy to show that the values of both determinants are strictly zero-definite.

## I.2. Recycle Flow-Pressure Cycles

The linearized differential equations governing flow pressure in the recycle loop in Fig. 6.10a are of the form:

$$dp_1/dt = (a_{13} \cdot f_3 - a_{12} \cdot f_2) + a_{11} \cdot f_1 \quad (I.1)$$

$$df_2/dt = b_2 \cdot (p_1 - p_2) - f_2 \quad (I.2)$$

$$df_3/dt = b_3 \cdot (p_2 - p_1) - f_3 \quad (I.3)$$

$$dp_2/dt = (a_{22} \cdot f_2 - a_{23} \cdot f_3) - a_{24} \cdot f_4 \quad (I.4)$$

The determinant of the system matrix formed by  $p_1$ ,  $f_2$   $f_3$  and  $p_2$  is:

$$D = (b_2 \cdot b_3 - b_2 \cdot b_3) \cdot (a_{12} \cdot a_{23} - a_{13} \cdot a_{22}) = 0 \quad (I.5)$$

The reduced graph is identical to that obtained after reduction in Appendix I.1, and its determinant is zero.

### I.3. Bypass Flow-Pressure Cycles

The linearized differential equations governing flow pressure in the bypass loop in Fig. 6.11a are of the form:

$$dp_1/dt = -(a_{13} \cdot f_3 + a_{12} \cdot f_2) + a_{11} \cdot f_1 \quad (I.6)$$

$$df_2/dt = b_2 \cdot (p_1 - p_2) - f_2 \quad (I.7)$$

$$df_3/dt = b_3 \cdot (p_1 - p_2) - f_3 \quad (I.8)$$

$$dp_2/dt = (a_{22} \cdot f_2 + a_{23} \cdot f_3) - a_{24} \cdot f_4 \quad (I.9)$$

The determinant of the system matrix formed by  $p_1$ ,  $f_2$   $f_3$  and  $p_2$  is:

$$D = (b_2 \cdot b_3 - b_2 \cdot b_3) \cdot (a_{12} \cdot a_{23} - a_{13} \cdot a_{22}) = 0 \quad (I.10)$$

The reduced graph is identical to that obtained after reduction in Appendix I.1, and its determinant is zero.

## APPENDIX J: Algorithms for Deriving the PM

In this appendix, two of the algorithms used in deriving the PM are presented. The algorithms are subroutines of the main algorithm shown in Fig. 6.12. The algorithms are expressed in pidgin algol, introduced in Appendix H. Tables of possible transitions of result nodes associated with compiled causal links attributed to successor paths derived by the algorithm described in Section J.2 are presented in Section J.3.

### J.1. Algorithm for Finding Primary Deviation Paths of Root Causes

The primary deviation paths of a root cause are the acyclic paths from which local cause links are derived. These are paths in the ESDG that may have minimum delays, depending on numerical values of time delays of arcs for the actual system. A variant of branch and bound search is used to find these paths.

**Input:** An augmented<sup>1</sup> directed graph,  $G(V,A)$ ; The root cause  $r$ .

**Output:** The list,  $D$ , of primary deviation paths for  $r$ .

```
begin
    Q :=  $\emptyset$ 
    form a path  $p$  containing  $r$  as its only node;
    Q :=  $\{p\} \cup Q$ ; (comment: initialize queue)
    D := find-primary-deviation-paths(Q);
    return D;
end
```

```
procedure find-primary-deviation-paths(Q)
begin
    (comment: Q is a queue with one path)
    S :=  $\emptyset$ ;
```

---

<sup>1</sup>The directed graph  $G(V,A)$  is augmented with a set of root causes,  $R$ .

$I := \emptyset;$   
**for all**  $p \in Q$  **do if** the last node added to  $p$  is a sensor signal  
**then**  $S := \{p\} \cup S$  **else**  $I := \{p\} \cup I;$   
**(comment:** initialize paths ending at signal variables and paths with  
zero additional delay)  
 $Q := I;$

**ln1:** **if**  $Q := \emptyset$  **then go to** ln2; **(comment:** no paths can be extended by arcs  
with zero delay)

let  $p$  be the first member of  $Q$ ;  
remove  $p$  from  $Q$ ;  
let  $w$  be the last node added to  $p$ ;  
find all children,  $c$ , of  $w$  such that the delay of the arc from  $w$  to  $c$  is 0;  
 $NS := \emptyset$ ; **(comment:** initialize new paths that end at sensor signal)  
 $NI := \emptyset$ ; **(comment:** initialize new paths with zero additional delay)  
**for all**  $c$  **do if**  $c$  is a signal variable **then**  
**begin**  
  extend  $p$  by  $c$  to form  $ns$ ;  
   $NS := \{ns\} \cup NS$ ;  
**end**  
**else**  
**begin**  
  extend  $p$  by  $c$  to form  $ni$ ;  
   $NI := \{ni\} \cup NI$ ;  
**end**  
**for all**  $ns \in NS$  **do if** the last node,  $c$ , added to  $ns$  is repeated or none of  
necessary disturbance variables of the last arc<sup>2</sup>,  $a$ , are  
in  $ns$ , or one of the delta variables of  $a$  is in  $ns$   
**then remove**  $ns$  **from**  $NS$ ;  
**(comment:** not valid acyclic path)  
**for all**  $ni \in NI$  **do if** the last node,  $c$ , added to  $ni$  is repeated or none of  
necessary disturbance variables of the last arc,  $a$ , are  
in  $ni$ , or one of the delta variables of  $a$  is in  $ni$

---

2These conditions refer to ESDG arcs.

```

then remove ni from NI;
(comment: not valid acyclic path)
S := NS U S;
I := NI U I;
for all ni ε NI do Q := {ni} U Q; (comment: add to front of queue)
go to ln1; (comment: continue DFS along arcs with zero delays)

ln2: if S ≠ φ then return S else
      begin
        DP := φ
        for all i ε I do
          begin
            let w be the last node added to i;
            find all children, c, of w such that the delay of the arc from w to c is 1;
            for all c do extend i by c to form dp, DP := {dp} U DP;
          end
        end (comment: extend each path by arcs with a delay of one)
        for all dp ε DP do if the last node, c, added to dp is repeated or none of
          necessary disturbance variables of the last arc, a, are
          in dp, or one of the delta variables of a is in dp
          then remove dp from DP;

        if DP ≠ φ then
          begin
            D = φ
            for all dp ε DP do
              begin
                Q1 = φ;
                Q1 := {dp} U Q1;
                D := D U find-primary-deviation-paths(Q1);
              end
            return D
          end
        else return φ
      end

```

## J.2. Algorithm for Finding Successor Paths from a Sensed Node

Successor paths from a sensed node of a signal variable are acyclic paths from which "not" and "only if exists" compiled causal links are derived. These are paths in the ESDG that may have minimum delays, depending on numerical values of time delays of arcs for the actual system. A variant of branch and bound search is used to find these paths. Associated with some of these paths are activating conditions. These activating conditions are the disabling conditions of other successor paths in the list. If the activating conditions of a path is the empty set, then the path is always active. Otherwise, the conditions indicate that the path may be active only when one of the root causes in the list occurs.

**Input:** A directed graph,  $G(V,A)$ ; The signal variable  $sv$ ,

The sensed node of the signal variable  $sn$ .

The set, EXR of root causes that can affect at most one signal variable  
(i.e. sensor failures).

**Output:** The list,  $L$ , of ordered pairs,  $l$ , for  $sn$ , and  $sv$ . The first element of  $l$ ,  $s$ , is the successor path and the second,  $ac$  the paths activating conditions.

```
begin
    Q :=  $\emptyset$ ;
    form a path p containing sn as its only variable;
    disb-conds[p] :=  $\emptyset$ ;
    new-disab-conds[p] :=  $\emptyset$ ;
    zero-delay-tag[p] := "true";
    if sn = sv then signal-var-in-path[p] := "true"
        else signal-var-in-path[p] := "false";
    Q := {p}  $\cup$  Q; (comment: initialize queue)
    L := find-successor-paths(Q,sv,EXR);
    return L;
end
```

```
procedure find-successor-paths(Q,sv,EXR)
    begin
```

(comment: Q is a queue with one path)  
let DC := `disab-conds[p]`; (comment: find root causes that disable path)  
`SP:= φ;`  
`I := φ;`  
**for all** `p ∈ Q` **do if** either the last variable, v, added to p is a signal variable  
and zero-delay-tag[p] = "false",  
or the last variable, v, added to p is a signal variable  
and signal-var-in-path[p] = "true" and  $v \neq sv$   
**then** `SP := {p} U SP` **else** `I := {p} U I;`  
(comment: initialize successor paths (SP) and paths with  
zero delay (I))  
`Q := I;`  
`CP := φ;` (comment: initialize new paths with more than one variable (CF)  
that have a zero delay and end at sv)

**ln1:** **if** `Q := φ` **then go to ln2;** (comment: no paths can be extended by arcs  
with zero delay)

let `p'` be the first member of `Q`;  
remove `p'` from `Q`;  
let `w` be the last variable added to `p'`;  
find all children, `c`, of `w` such that the delay of the arc from `w` to `c` is 0;  
`NSP := φ;` (comment: initialize new sucessor paths)  
`NI := φ;` (comment: initialize new paths with zero delay)  
`NCP := φ;` (comment: initialize new paths with zero delay ending at sv)  
**for all** `c` **do if** `c` is a signal variable **then**  
**begin**  
  extend `p'` by `c` to form `nsp`;  
  `NSP := {nsp} U NSP;`  
**end**  
**else**  
**begin**  
  extend `p'` by `c` to form `ni`;  
  `NI := {ni} U NI;`  
**end**  
**for all** `nsp ∈ NSP` **do if** the last variable, `c`, added to `nsp` is repeated or

none of necessary disturbance variables of last arc<sup>3</sup>, a,  
 are in nsp, or one of the delta variables of a is in nsp  
**then remove nsp from NS**  
**(comment:** not valid acyclic path)  
**else**  
**begin**  
       disab-conds[nsp] := DC  
       let c be the last variable added to nsp;  
       let dsb be the disabling conditions of the  
           last arc, a, in nsp;  
       new-disab-conds[nsp]  
           := dsb U new-disab-conds[p'];  
       zero-delay-tag[nsp] := zero-delay-tag[p'];  
       **if** c = sv **then** signal-var-in-path[nsp] := "true"  
           **else** signal-var-in-path[nsp]  
               := signal-var-in-path[p'];  
**end;**  
**for all** ni ε NI **do if** the last variable, c, added to ni is repeated or none of  
           necessary disturbance variables of the last arc, a, are  
           in ni, or one of the delta variables of a is in ni  
**then remove ni from NI**  
**(comment:** not valid acyclic path)  
**else**  
**begin**  
       disab-conds[ni] := DC  
       let c be the last variable added to ni;  
       let dsb be the disabling conditions of the  
           last arc, a, in ni;  
       new-disab-conds[ni]  
           := dsb U new-disab-conds[p'];  
       zero-delay-tag[ni] := zero-delay-tag[p'];  
       signal-var-in-path[ni] := signal-var-in-path[p'];  
**end;**  
**for all** nsp ε NSP **do if** zero-delay-tag[nsp] = "true" **and**

---

3These conditions refer to ESDG arcs, when the acyclic path traverses more than one SCC.

```

    signal-var-in-path[nsp] = "false"
    then remove nsp from NSP;
(comment: remove as paths are successor-paths of sv's precursor)
for all nsp  $\in$  NSP do if zero-delay-tag[nsp] = "true" and
    signal-var-in-path[nsp] = "true"
    then NCP := NCP U {nsp}, remove nsp from NSP;
(comment: do not search along these paths as it isolates other paths)
CP := NCP U CP;
SP := NSP U SP;
I := NI U I;
for all ni  $\in$  NI do Q := {ni} U Q; (comment: add to front of queue)
go to ln1; (comment: continue DFS along arcs with zero delays)

```

ln2:

```

NDC :=  $\phi$ ; (comment: initialize new disabling conditions common to
all successor paths)
if SP  $\neq \phi$  then
    begin
        let NDC be the universal set;
        for all sp  $\in$  SP do NDC := NDC  $\cap$  new-disab-conds[sp];
    end;

if SP  $\neq \phi$  and NDC =  $\phi$  then
    begin
        L :=  $\phi$ 
        for all sp  $\in$  SP do begin
            pdc := disb-conds[sp];
            pndc := new-disab-conds[sp];
            ac := disb-conds[sp] - new-disab-conds[sp] - EXR;
            if pdc  $\neq \phi$  and ac =  $\phi$  then remove sp from SP
            else begin
                activate-conds[sp] := ac;
                L := L U {(sp activate-conds[sp])};
            end;
        end;
        return L;
    end;

```

```

DI =  $\emptyset$ ;
for all i  $\in$  I do
begin
    let w be the last variable added to i;
    find all children, c, of w such that the delay of the arc from w to c is 1;
    for all c do
        begin
            let a be the arc between w and c;
            extend i by c to form di;
            if c is not repeated in di, and at least one of the necessary disturbance
                variables of a4 is in di and none of the delta variables of a is in di
                (comment: valid acyclic path)
        then begin
            let dsb be the disabling conditions of the arc from w to c;
            if c = sv then signal-var-in-path[di] := "true"
                else signal-var-in-path[di] := signal-var-in-path[i];
            zero-delay-tag[di] := "false";
            new-disab-conds[di] := new-disab-conds[i]  $\cup$  dsb;
            disb-conds[di] := DC;
            DI := DI  $\cup$  {di};
        end;
    end;
end;

```

CP := CP  $\cup$  DI

```

if CP  $\neq$   $\emptyset$  and NDC  $\neq$   $\emptyset$  then
begin
    L :=  $\emptyset$ 
    for all sp  $\in$  SP do begin
        pdc := disb-conds[sp];
        pndc := new-disab-conds[sp];
        ac := disb-conds[sp] - new-disab-conds[sp] - EXR;

```

---

<sup>4</sup>These conditions refer to ESDG arcs, when the acyclic path traverses more than one SCC.

```

if pdc  $\neq \emptyset$  and ac  $= \emptyset$  then remove sp from SP
else begin
    activate-conds[sp] := ac;
    L := L U {(sp activate-conds[sp])};
end;
end;

FSP :=  $\emptyset$ ; (comment: initialize paths for further search)
for all cp  $\in$  CP do disb-conds[cp] := disb-conds[cp] U NDC;
for all cp  $\in$  CP do begin
    pdc := disb-conds[cp];
    pndc := new-disab-conds[cp];
    ac := disb-conds[cp] - new-disab-conds[cp] - EXR;
    if pdc  $= \emptyset$  or ac  $\neq \emptyset$ ;
    then begin
        disb-conds[cp] := ac;
        new-disab-conds[cp] :=  $\emptyset$ ;
        FSP := {cp} U FSP;
    end;
    end;
for all fsp  $\in$  FSP do begin
    NQ :=  $\emptyset$ ;
    NQ := NQ U {fsp};
    L := L U find-successor-paths(NQ,sv,EXR);
    end;
return L;
end;

if CP  $\neq \emptyset$  then
begin
    L :=  $\emptyset$ 
    FSP :=  $\emptyset$ ; (comment: initialize paths for further search)
    for all cp  $\in$  CP do begin
        pdc := disb-conds[cp];
        pndc := new-disab-conds[cp];
        ac := disb-conds[cp] - new-disab-conds[cp] - EXR;
        if pdc  $= \emptyset$  or ac  $\neq \emptyset$ ;

```

```

then begin
    disab-conds[cp] := ac;
    new-disab-conds[cp] := φ;
    FSP := {cp} U FSP;
    end;
    end;
for all fsp ∈ FSP do begin
    NQ := φ;
    NQ := NQ U {fsp};
    L := L U find-successor-paths(NQ,sv,EXR);
    end;
return L;
end;

if CP = φ then return φ;

end

```

### J.3. Possible Transitions of Compiled Causal Links

In this section, tables showing transitions associated with result nodes of compiled causal links are presented. The transitions are attributed to successor relationships between source and result nodes of the links, and are derived from the signs of successor paths, equivalent paths and the consequent state of the source node. The sign of a path in the ESDG is the product of all signs in the path. In addition, if any variable in the path is one of the delta variables of an upstream arc, a delta condition is associated with the sign. This indicates that the particular path may only be associated with transitions where the result node returns back to the normal state. Thus, possible signs associated with paths in the ESDG are, +, -,  $+δ$  and  $-δ$ .

Because all arcs in equivalent paths have a time-delay of 0, the paths necessarily contain only standard SDG arcs. Therefore, the signs of equivalent paths may only be + or -. Tables J.1 and J.2 show possible transitions of the result node for combinations of signs of successor paths and consequent states of the source state, for equivalent paths of positive

sign. Transitions for negative signed equivalent paths may be deduced using symmetry relations.

**Table J.1 Possible Transitions Associated with Causal Links  
(Source Node and Result Node Different)**

Source State Sign of Successor Path	High	Normal	Low
+	normal-high low-normal low-high	high-normal low-normal	normal-low high-normal high-low
-	normal-low high-normal high-low	high-normal low-normal	normal-high low-normal low-high
$+δ$	low-normal	high-normal low-normal	high-normal
$-δ$	high-normal	high-normal low-normal	low-normal

**Table J.2 Possible Transitions Associated with Causal Links  
(Source Node and Result Node Identical)**

Source State Sign of Successor Path	High	Normal	Low
+	none	none	none
-	high-normal high-low	normal-low normal-high	low-normal low-high
+ $\delta$	none	none	none
- $\delta$	high-normal	none	low-normal

## APPENDIX K: Portion of the PM for Jacketed Reactor Process

```
;*****  
;; SECTION 1: Screen Interface Instances *  
;*****  
  
(MAKE-INSTANCE CHOOSE-TEST  
  :print-name "CHOOSE-TEST"  
  :doc-string ""  
  :is 'POPUP-CHOOSE  
  :slots  
   '(  
    (INSTRUCTIONS :VALUE  
     (" ** Select Test ** "))  
    (BORDER-COLOR :VALUE :CYAN)  
    (ANSWER :WHEN-MODIFIED (ACTIVATE-POPUP))  
    (TOP :VALUE 8)  
    (LEFT :VALUE 32)  
    (CENTER :VALUE :NO-CENTERING)  
    (CONTENTS :VALUE  
      (( " CANCEL " :CANCEL)  
       (" PUMP LEAK INSPECTION_TRUE " T-PUMP LEAK INSPECTION_TRUE)  
       (" PUMP LEAK INSPECTION_FALSE " T-PUMP LEAK INSPECTION_FALSE)  
       •  
       •  
       (" CSTR FIRE INSPECTION_TRUE " T-CSTR FIRE INSPECTION_TRUE)  
       (" CSTR FIRE INSPECTION_FALSE " T-CSTR FIRE INSPECTION_FALSE)  
     ))  
  ))  
  
(MAKE-INSTANCE CHOOSE-SYSTEM  
  :print-name "CHOOSE-SYSTEM"  
  :doc-string ""  
  :is 'POPUP-CHOOSE  
  :slots  
   '(  
    (INSTRUCTIONS :VALUE  
     (" ** Select System ** "))  
    (BORDER-COLOR :VALUE :CYAN)  
    (ANSWER :WHEN-MODIFIED (ACTIVATE-POPUP))  
    (TOP :VALUE 8)  
    (LEFT :VALUE 50)  
    (CENTER :VALUE :X)  
    (CONTENTS :VALUE  
      (( " CANCEL " :CANCEL)))  
    (FORCE-CHOICE :VALUE :YES)  
  ))  
  
(MAKE-INSTANCE CHOOSE-CONSTRAINT  
  :print-name "CHOOSE-CONSTRAINT"  
  :doc-string ""  
  :is 'POPUP-CHOOSE
```

```

:slots
  (
(INSTRUCTIONS :VALUE
  (" ** Select Constraint ** "))
(BORDER-COLOR :VALUE :CYAN)
(ANSWER :WHEN-MODIFIED (ACTIVATE-POPUP))
(TOP :VALUE 8)
(LEFT :VALUE 47)
(CENTER :VALUE :NO-CENTERING)
(CONTENTS :VALUE
  ( (" CANCEL " :CANCEL)
    (" INVENTORY_CONSTRAINT " MC-INVENTORY)

  •

  ))
(FORCE-CHOICE :VALUE :YES)
))

(MAKE-INSTANCE CHOOSE-MONITOR
:print-name "CHOOSE-MONITOR"
:doc-string ""
:is 'POPUP-CHOOSE
:slots
  (
(INSTRUCTIONS :VALUE
  (" ** Select Monitor ** "))
(BORDER-COLOR :VALUE :CYAN)
(ANSWER :WHEN-MODIFIED (ACTIVATE-POPUP))
(TOP :VALUE 8)
(LEFT :VALUE 44)
(CENTER :VALUE :NO-CENTERING)
(CONTENTS :VALUE
  ( (" CANCEL " :CANCEL)
    ("REACTOR_TEMPERATURE_SENSOR " REACTOR_TEMPERATURE_SENSOR)

  •

    ("INVENTORY_CONSTRAINT " INVENTORY_CONSTRAINT)
  ))
(FORCE-CHOICE :VALUE :YES)
))

(MAKE-INSTANCE CHOOSE-POTENTIAL-CAUSE
:print-name "CHOOSE-POTENTIAL-CAUSE"
:doc-string ""
:is 'PCHOOSE-LINK"
:doc-string ""
:is 'POPUP-CHOOSE
:slots
  (
(INSTRUCTIONS :VALUE
  (" ** Select Link ** "))
(BORDER-COLOR :VALUE :CYAN)
(ANSWER :WHEN-MODIFIED (ACTIVATE-POPUP))
(TOP :VALUE 8)
(LEFT :VALUE 57)
(CENTER :VALUE :NO-CENTERING)
(CONTENTS :VALUE
  ( (" CANCEL " :CANCEL)
    (" CAUSAL-LINK-1 " CAUSAL-LINK-1)

```

```

        •
        •
        (" CAUSAL-LINK-334 " CAUSAL-LINK-334)
))
  (FORCE-CHOICE :VALUE :YES)
))

(MAKE-INSTANCE CHOOSE-VARIABLE
  :print-name "CHOOSE-VARIABLE"
  :doc-string ""
  :is 'POPUP-CHOOSE
  :slots
    '(
      (INSTRUCTIONS :VALUE
        (" ** Select Variable ** "))
      (BORDER-COLOR :VALUE :CYAN)
      (ANSWER :WHEN-MODIFIED (ACTIVATE-POPUP))
      (TOP :VALUE 8)
      (LEFT :VALUE 51)
      (CENTER :VALUE :NO-CENTERING)
      (CONTENTS :VALUE
        ('(
          (" CANCEL " :CANCEL)
          (" REACTOR_TEMPERATURE " MV-REACTOR_TEMPERATURE)

          •
          (" LEVEL_CONTROLLER_SIGNAL " MV-LEVEL_CONTROLLER_SIGNAL)
        )))
      (FORCE-CHOICE :VALUE :YES)
    ))

(MAKE-INSTANCE CHOOSE-NEXT-EVENT
  :print-name "CHOOSE-NEXT-EVENT"
  :doc-string ""
  :is 'POPUP-CHOOSE
  :slots
    '(
      (INSTRUCTIONS :VALUE
        (" ** Enter Next Event ** "))
      (BORDER-COLOR :VALUE :CYAN)
      (ANSWER :WHEN-MODIFIED (ACTIVATE-CHOSEN-EVENT))
      (TOP :VALUE 8)
      (LEFT :VALUE 32)
      (CENTER :VALUE :NO-CENTERING)
      (CONTENTS :VALUE
        ('(
          (" CANCEL " :CANCEL)
          (" REACTOR_TEMPERATURE : NORMAL --> HIGH " EV-1)
          (" REACTOR_TEMPERATURE : NORMAL --> LOW " EV-2)
          (" REACTOR_TEMPERATURE : HIGH --> NORMAL " EV-3)
          (" REACTOR_TEMPERATURE : HIGH --> LOW " EV-4)
          (" REACTOR_TEMPERATURE : LOW --> NORMAL " EV-5)
          (" REACTOR_TEMPERATURE : LOW --> HIGH " EV-6)
          (" REACTOR_TEMPERATURE : SUSPICIOUS " EV-7)
          (" REACTOR_TEMPERATURE : ACCEPTABLE " EV-8)

          •
          •
          (" INVENTORY_CONSTRAINT : NORMAL --> HIGH " EV-113)
        )))
      )
    ))
)
)
)
```

```

(" INVENTORY_CONSTRAINT : SUSPICIOUS " EV-119)
( " PUMP_LEAK_INSPECTION : TRUE "
  T-PUMP_LEAK_INSPECTION_TRUE)
( " PUMP_LEAK_INSPECTION : FALSE"
  T-PUMP_LEAK_INSPECTION_FALSE)
))
(FORCE-CHOICE :VALUE :YES)
))

*****  

;; SECTION 2: Potential Event Instances *
*****
;
```

•

```

(DEFINE-INSTANCE EV-120
  (:print-name "EV-120"
   :doc-string ""
   :is POTENTIAL-EVENT)
  (ID EV-120)
  (TYPE CONSTRAINT)
  (OBJECT MC-INVENTORY)
  (PRIOR-STATUS FAILED)
  (CONSEQUENT-STATUS OK)
  (SPECIAL-CONDITIONS)
  (EXCLUSIVE-EVENTS EV-119)
  (ACTIVE NO)
  )

(DEFINE-INSTANCE EV-119
  (:print-name "EV-119"
   :doc-string ""
   :is POTENTIAL-EVENT)
  (ID EV-119)
  (TYPE CONSTRAINT)
  (OBJECT MC-INVENTORY)
  (PRIOR-STATUS OK)
  (CONSEQUENT-STATUS FAILED)
  (SPECIAL-CONDITIONS)
  (EXCLUSIVE-EVENTS EV-120)
  (ACTIVE NO)
  )



•


(DEFINE-INSTANCE EV-117
  (:print-name "EV-117"
   :doc-string ""
   :is POTENTIAL-EVENT)
  (ID EV-117)
  (TYPE CONSTRAINT)
  (OBJECT MC-INVENTORY)
  (PRIOR-STATE LOW)
  (CONSEQUENT-STATE NORMAL)
  (EXCLUSIVE-EVENTS EV-113 EV-114 EV-115 EV-116 EV-118)
  (ACTIVE NO)
  )
```

```
(DEFINE-INSTANCE EV-116
  (:print-name "EV-116"
   :doc-string ""
   :is POTENTIAL-EVENT)
  (ID EV-116)
  (TYPE CONSTRAINT)
  (OBJECT MC-INVENTORY)
  (PRIOR-STATE HIGH)
  (CONSEQUENT-STATE LOW)
  (EXCLUSIVE-EVENTS EV-113 EV-114 EV-115 EV-117 EV-118)
  (ACTIVE NO)
  )
```

•

```
(DEFINE-INSTANCE EV-7
  (:print-name "EV-7"
   :doc-string ""
   :is POTENTIAL-EVENT)
  (ID EV-7)
  (TYPE VARIABLE)
  (OBJECT MV-REACTOR_TEMPERATURE)
  (PRIOR-STATUS OK)
  (CONSEQUENT-STATUS FAILED)
  (SPECIAL-CONDITIONS)
  (EXCLUSIVE-EVENTS EV-8)
  (ACTIVE NO)
  )
```

```
(DEFINE-INSTANCE EV-6
  (:print-name "EV-6"
   :doc-string ""
   :is POTENTIAL-EVENT)
  (ID EV-6)
  (TYPE VARIAELE)
  (OBJECT MV-REACTOR_TEMPERATURE)
  (PRIOR-STATE LOW)
  (CONSEQUENT-STATE HIGH)
  (EXCLUS!VE-EVENTS EV-1 EV-2 EV-3 EV-4 EV-5)
  (ACTIVE NO)
  )
```

•

```
;*****  
;; SECTION 3:Compiled Causal Link Instances *  
;*****
```

```
(DEFINE-INSTANCE CAUSAL-LINK-334
  (:print-name "CAUSAL-LINK-334"
   :doc-string ""
   :is COMPILED-CAUSAL-LINK)
  (ACTIVE YES)
  (ONLY-IF-CONDITIONS (:ONLY-IF-TRANSIENT))
  (ID CAUSAL-LINK-334)
  (SOURCE-NODE MC-MOL_BALANCE)
```

```

(SOURCE-STATUS FAILED)
(RESULT-NODE MC-MOL_BALANCE)
(RESULT-STATUS OK)
)

•

(DEFINE-INSTANCE CAUSAL-LINK-330
  (:print-name "CAUSAL-LINK-330"
   :doc-string ""
   :is COMPILED-CAUSAL-LINK)
(ACTIVE YES)
(ONLY-IF-CONDITIONS (:ONLY-IF-TRANSIENT))
(ID CAUSAL-LINK-330)
(SOURCE-NODE MV-LEVEL_CONTROLLER_SIGNAL)
(SOURCE-STATUS FAILED)
(RESULT-NODE MV-LEVEL_CONTROLLER_SIGNAL)
(RESULT-STATUS OK)
)

•

(DEFINE-INSTANCE CAUSAL-LINK-316
  (:print-name "CAUSAL-LINK-316"
   :doc-string ""
   :is COMPILED-CAUSAL-LINK)
(ACTIVE YES)
(ONLY-IF-CONDITIONS (:ONLY-IF-TRANSIENT))
(ID CAUSAL-LINK-316)
(SOURCE-NODE MC-MOL_BALANCE)
(SOURCE-STATE LOW)
(RESULT-NODE MC-MOL_BALANCE)
(RESULT-STATE NORMAL)
)

•

(DEFINE-INSTANCE CAUSAL-LINK-315
  (:print-name "CAUSAL-LINK-315"
   :doc-string ""
   :is COMPILED-CAUSAL-LINK)
(ACTIVE YES)
(ONLY-IF-CONDITIONS (:ONLY-IF-TRANSIENT))
(ID CAUSAL-LINK-315)
(SOURCE-NODE MC-MOL_BALANCE)
(SOURCE-STATE HIGH)
(RESULT-NODE MC-MOL_BALANCE)
(RESULT-STATE NORMAL)
)

•

(DEFINE-INSTANCE CAUSAL-LINK-308
  (:print-name "CAUSAL-LINK-308"
   :doc-string ""
   :is COMPILED-CAUSAL-LINK)
(ACTIVE YES)
(ONLY-IF-CONDITIONS (:ONLY-IF-TRANSIENT))

```

```

(ID CAUSAL-LINK-308)
(SOURCE-NODE MV-LEVEL_CONTROLLER_SIGNAL)
(SOURCE-STATE LOW)
(RESULT-NODE MV-LEVEL_CONTROLLER_SIGNAL)
(RESULT-STATE NORMAL)
)

(DEFINE-INSTANCE CAUSAL-LINK-307
  (:print-name "CAUSAL-LINK-307"
    :doc-string ""
    :is COMPILED-CAUSAL-LINK)
  (ACTIVE YES)
  (ONLY-IF-CONDITIONS (:ONLY-IF-TRANSIENT))
  (ID CAUSAL-LINK-307)
  (SOURCE-NODE MV-LEVEL_CONTROLLER_SIGNAL)
  (SOURCE-STATE HIGH)
  (RESULT-NODE MV-LEVEL_CONTROLLER_SIGNAL)
  (RESULT-STATE NORMAL)
)

(DEFINE-INSTANCE CAUSAL-LINK-306
  (:print-name "CAUSAL-LINK-306"
    :doc-string ""
    :is COMPILED-CAUSAL-LINK)
  (ACTIVE YES)
  (NOT-CONDITIONS
    (:NOT
      (PRC-REACTOR_FEED_SOURCE_TEMPERATURE_LOW
       PRC-JACKET_EFFLUENT_SINK_PRESSURE_LOW
      .
      .
      PRC-REACTOR_TEMPERATURE_SENSOR_FAILED_LOW)))
  (ID CAUSAL-LINK-306)
  (SOURCE-NODE MV-CW_FLOW_CONTROLLER_SIGNAL)
  (SOURCE-STATE LOW)
  (RESULT-NODE MV-CW_FLOW_CONTROLLER_SIGNAL)
  (RESULT-STATE NORMAL)
)

.

.

.

(DEFINE-INSTANCE CAUSAL-LINK-292
  (:print-name "CAUSAL-LINK-292"
    :doc-string ""
    :is COMPILED-CAUSAL-LINK)
  (ACTIVE YES)
  (NOT-CONDITIONS
    (:NOT
      (PRC-CONCENTRATION_A_SENSOR_FAILED_LOW
       PRC-CONCENTRATION_A_SENSOR_BIASED_LOW
       PRC-PRODUCT_FLOW_SENSOR_BIASED_LOW
      .
      .
      PRC-PRODUCT_FLOW_SENSOR_FAILED_LOW
      PRC-CSTR_INLET_BLOCKAGE)))

```

```

(ID CAUSAL-LINK-292)
(SOURCE-NODE MV-LEVEL_CONTROLLER_SIGNAL)
(SOURCE-STATE LOW)
(RESULT-NODE MV-CONCENTRATION_A)
(RESULT-STATE HIGH)
)

•

(DEFINE-INSTANCE CAUSAL-LINK-2
  (:print-name "CAUSAL-LINK-2"
    :doc-string ""
    :is COMPILED-CAUSAL-LINK)
  (ACTIVE YES)
  (NOT-CONDITIONS
    (:NOT
      (PRC-REACTOR_TEMPERATURE_SENSOR_FAILED_LOW
      •
      •
      PRC-CSTR_CATALYST_1_DEACTIVATION
      PRC-JACKET_EFFLUENT_SINK_PRESSURE_LOW)))
  (ID CAUSAL-LINK-2)
  (SOURCE-NODE
    MV-REACTOR_TEMPERATURE)
  (SOURCE-STATE LOW)
  (RESULT-NODE
    MV-REACTOR_TEMPERATURE)
  (RESULT-STATE NORMAL)
  )

•

;;*****
;; SECTION 4: Measured Variable Instances *
;;*****



(DEFINE-INSTANCE MV-LEVEL_CONTROLLER_SIGNAL
  (:print-name "MV-LEVEL_CONTROLLER_SIGNAL"
    :doc-string "OUTPUT SIGNAL in CONTROLLER: LEVEL_CONTROLLER"
    :is MEASURED-VARIABLES)
  (LOCAL-CAUSES PRC-REACTOR_LEVEL_SENSOR_BIASED_LOW
    PRC-REACTOR_LEVEL_SENSOR_BIASED_HIGH
    •
    •
    PRC-CSTR-INLET-BLOCKAGE)
  (EVENT-SET EV-105 EV-106 EV-107 EV-108 EV-109 EV-110 EV-111 EV-112)
  (ID MV-LEVEL_CONTROLLER_SIGNAL)
  (TYPE VARIABLE)
  (SUCCESSOR-LINKS CAUSAL-LINK-255 CAUSAL-LINK-266
    •
    CAUSAL-LINK 330)
  (PRECURSOR-LINKS CAUSAL-LINK-10 CAUSAL-LINK-11
    •

```

CAUSAL-LINK-308 CAUSAL-LINK 330)

)

•  
\*\*\*\*\*  
;; SECTION 5: Measured Constraint Instances \*  
\*\*\*\*\*

(DEFINE-INSTANCE MC-INVENTORY  
  (:print-name "MC-INVENTORY"  
    :doc-string ""  
    :is MEASURED-CONSTRAINTS)  
  (LOCAL-CAUSES PRC-CSTR\_LEAK\_TO\_ENVIRONMENT  
  
    •  
    PRC-PRODUCT\_FLOW\_SENSOR\_FAILED\_LOW)  
  (EVENT-SET EV-113 EV-114 EV-115 EV-116 EV-117 EV-118 EV-119 EV-120)  
  (ID MC-INVENTORY)  
  (TYPE CONSTRAINT)  
  (SUCCESSOR-LINKS CAUSAL-LINK-309 CAUSAL-LINK-310 CAUSAL-LINK 331)  
  (PRECURSOR-LINKS CAUSAL-LINK-309 CAUSAL-LINK-310 CAUSAL-LINK 331)  
  (CURRENT-STATE NORMAL)  
)

•  
\*\*\*\*\*  
;; SECTION 6: Potential Root Cause Instances \*  
\*\*\*\*\*

(DEFINE-INSTANCE PRC-CSTR\_LEAK\_TO\_ENVIRONMENT  
  (:print-name "PRC-CSTR\_LEAK\_TO\_ENVIRONMENT"  
    :doc-string  
    "LEAK\_TO\_ENVIRONMENT in  
      JACKETED\_CONTINUOUS\_STIRRED\_TANK\_REACTOR: CSTR"  
    :is POTENTIAL-ROOT-CAUSE)  
  (ID PRC-CSTR\_LEAK\_TO\_ENVIRONMENT)  
  (PRIMARY-DEVIATION  
    ((MV-LEVEL\_CONTROLLER\_SIGNAL LOW)  
      (MV-REACTOR\_LEVEL LOW))

•  
•  
  (MC-INVENTORY LOW)  
  (MC-MOL\_BALANCE LOW))  
))  
(APPLICABLE-TESTS T-CSTR\_LEAK\_INSPECTION\_TRUE  
                  T-CSTR\_LEAK\_INSPECTION\_FALSE)  
(PRIOR-PROBABILITY 0.0)  
)

(DEFINE-INSTANCE PRC-FEED\_FLOW\_SENSOR\_FAILED\_HIGH  
  (:print-name "PRC-FEED\_FLOW\_SENSOR\_FAILED\_HIGH"  
    :doc-string "SENSOR FAILED HIGH in SENSOR: FEED\_FLOWRATE\_SENSOR"  
    :is POTENTIAL-ROOT-CAUSE)  
  (ID PRC-FEED\_FLOW\_SENSOR\_FAILED\_HIGH)

```

(PRIMARY-DEVIATION
  ((MV-FEED_FLOWRATE HIGH)
   (MC-INVENTORY LOW)
   (MC-MOL_BALANCE LOW)
   (MV-FEED_FLOWRATE NIL NIL FAILED)
   (MC-INVENTORY NIL NIL FAILED)
   (MC-MOL_BALANCE NIL NIL FAILED))
))

(PRIOR-PROBABILITY 1.0)
)

```

•

```

;*****;;
;; SECTION 7: Test Instances *
;*****;;

```

```

(DEFINE-INSTANCE T-PUMP_LEAK_INSPECTION_TRUE
  (:print-name "T-PUMP_LEAK_INSPECTION_TRUE"
   :doc-string ""
   :is TEST)
  (ID T-PUMP_LEAK_INSPECTION_TRUE)
  (EXCLUSIVE-TESTS T-PUMP_LEAK_INSPECTION_FALSE)
  (ONLY-IF-CONDITIONS
    (:ONLY-IF (PRC-PUMP_LEAK_TO_ENVIRONMENT))) )

```

```

(DEFINE-INSTANCE T-PUMP_LEAK_INSPECTION_FALSE
  (:print-name "T-PUMP_LEAK_INSPECTION_FALSE"
   :doc-string ""
   :is TEST)
  (ID T-PUMP_LEAK_INSPECTION_FALSE)
  (EXCLUSIVE-TESTS T-PUMP_LEAK_INSPECTION_TRUE)
  (NOT-CONDITIONS
    (:NOT (PRC-PUMP_LEAK_TO_ENVIRONMENT))) )

```

•

```

;*****;;
;; SECTION 8: Constraint Monitor Instances *
;*****;;

```

```

(DEFINE-INSTANCE MOL_BALANCE_CONSTRAINT
  (:print-name "MOL_BALANCE_CONSTRAINT"
   :doc-string ""
   :is CONSTRAINT-MONITOR)
  (STATE-EVENTS EV-137 EV-138 EV-139 EV-140 EV-141
   EV-142)
  (MONITOR-EVENTS EV-143 EV-144)
  (PREDICTION-VALID NO)
  (PREDICTED-STATUS OK)
  (PREDICTED-TREND STEADY)
  (PREDICTED-STATE NORMAL)
  (PERCEIVED-TREND STEADY)
  (PERCEIVED-STATE NORMAL)
  (PERCEIVED-MONITOR-STATUS OK)
  (RANGE-LCL 0.0)
  (RANGE-NOMINAL 0.0)

```

```

(RANGE-UCL 202.0)
(SDEV-LCL 0.0)
(SDEV-LWL 0.0)
(SDEV-NOMINAL 0.0)
(SDEV-UWL 999.9)
(SDEV-UCL 999.9)
(MEAN-LCL -3.02743)
(MEAN-LWL -2.25309)
(MEAN-NOMINAL -3.31778F-01)
(MEAN-UWL 1.61864)
(MEAN-UCL 2.39298)
(POLYNOMIAL-ORDER 3)
(DRIFT-CONSTANT 0.03)
(FILTER-CONSTANT 0.7)
(FILTERED-SAMPLE-SIZE 10)
(RAW-SAMPLE-SIZE 1)
(DRIFT-HORIZON 10)
(ANALYSIS-HORIZON 3)
(PREDICTION-HORIZON 1)
(MAXIMUM-VALUE 100.0)
(MINIMUM-VALUE -100.0)
(ID MOL_BALANCE_CONSTRAINT)
(ASSOCIATION MC-MOL_BALANCE)
(MONITOR-TYPE CONSTRAINT-MONITOR)
)

```

```

;*****;;
;; SECTION 9: Sensor Monitor Instances      *
;*****;;

```

```

(DEFINE-INSTANCE LEVEL_CONTROLLER_OUTPUT_SIGNAL
  (:print-name "LEVEL_CONTROLLER_OUTPUT_SIGNAL"
   :doc-string ""
   :is SENSOR-MONITOR)
(STATE-EVENTS EV-105 EV-106 EV-107 EV-108 EV-109
  EV-110)
(MONITOR-EVENTS EV-111 EV-112)
(PREDICTION-VALID NO)
(PREDICTED-STATUS OK)
(PREDICTED-TREND STEADY)
(PREDICTED-STATE NORMAL)
(PERCEIVED-TREND STEADY)
(PERCEIVED-STATE NORMAL)
(PERCEIVED-MONITOR-STATUS OK)
(RANGE-LCL 0.0)
(RANGE-NOMINAL 7.91379F-01)
(RANGE-UCL 101.0)
(SDEV-LCL 0.0)
(SDEV-LWL 0.0)
(SDEV-NOMINAL 4.11021F-01)
(SDEV-UWL 999.9)
(SDEV-UCL 999.9)
(MEAN-LCL 24.053)
(MEAN-LWL 24.3801)
(MEAN-NOMINAL 25.2507)
(MEAN-UWL 26.0157)
(MEAN-UCL 26.3428)
(POLYNOMIAL-ORDER 3)

```

(DRIFT-CONSTANT 0.05)  
(FILTER-CONSTANT 0.8)  
(FILTERED-SAMPLE-SIZE 10)  
(RAW-SAMPLE-SIZE 3)  
(DRIFT-HORIZON 10)  
(ANALYSIS-HORIZON 9)  
(PREDICTION-HORIZON 3)  
(MAXIMUM-VALUE 101.0)  
(MINIMUM-VALUE -1.0)  
(ID LEVEL\_CONTROLLER\_OUTPUT\_SIGNAL)  
(ASSOCIATION MV-LEVEL\_CONTROLLER\_SIGNAL)  
(MONITOR-TYPE SENSOR-MONITOR)  
)

•