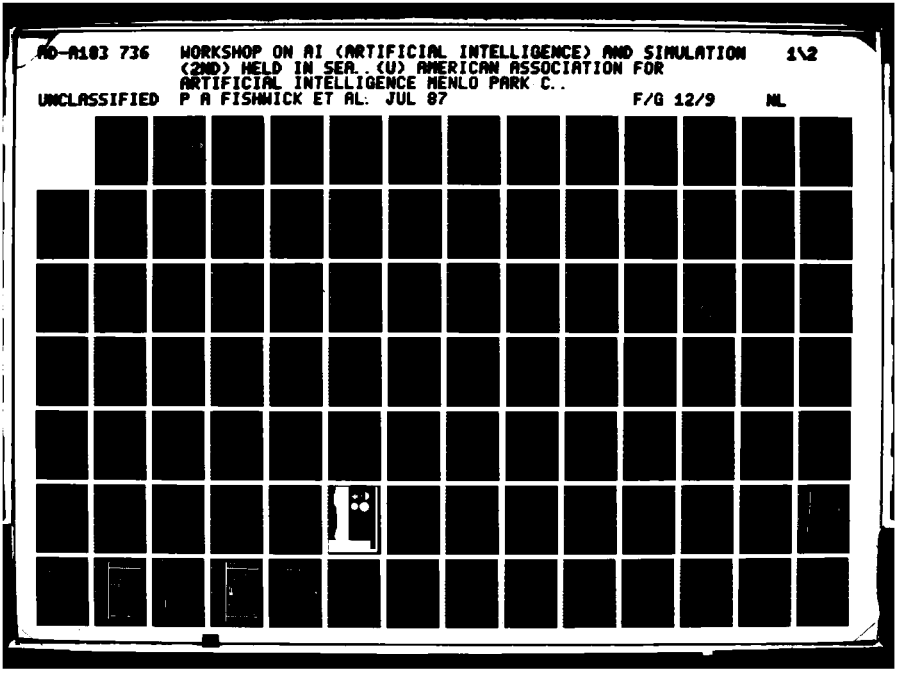
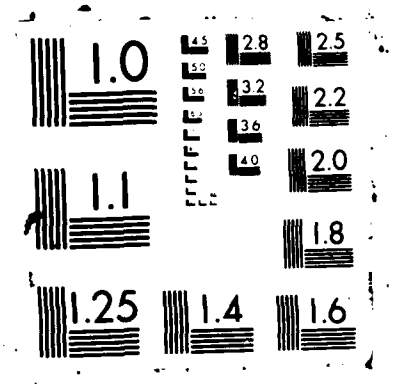
AD-A183 736 WORKSHOP ON AI (ARTIFICIAL INTELLIGENCE) AND SIMULATION (2ND) HELD IN SEA.. (U) AMERICAN ASSOCIATION FOR ARTIFICIAL INTELLIGENCE MENLO PARK C.. 1\2

UNCLASSIFIED P A FISHWICK ET AL. JUL 87 F/G 12/9 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5

(2)

SECOND WORKSHOP ON AI AND SIMULATION

14 JULY 1987

AAAI CONFERENCE

SEATTLE, WASHINGTON

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

American Association on Artificial Intelligence-Second Workshop on Artificial Intelligence (AI) and Simulation

TYPE OF REPORT & PERIOD COVERED: Workshop Report-14 July 1987

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

Paul A. Fishwick, Richard B. Modjeski, Bernard P. Ziegler, Ramana Reddy, Marilyn Stelzner, and Malcolm R. Railey

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS

American Association for Artificial Intelligence
445 Burgess Drive
Menlo Park, California 94025-3496

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

U.S. Army Concepts Analysis Agency
8120 Woodmont Avenue
Bethesda, MD 20814-2797

12. REPORT DATE: July 1987

13. NUMBER OF PAGES: 136

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This workshop was sponsored and funded by the American Association on Artificial Intelligence. See back for affliations of the authors.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Artificial Intelligence (U)
Computer Science (U)
Computer Simulation Modeling (U)
Expert Systems (U)
Knowledge Based Simulation (U)
Computer Modeling (U)
Operations Research (U)
Qualitative Simulation (U)
Simulation (U)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Second Artificial Intelligence (AI) and Simulation Workshop was held during the National Conference on Artificial Intelligence on July 14, 1987 at the University of Washington, Seattle. The following abstracts and papers were part of the workshop.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SECOND WORKSHOP ON AI AND SIMULATION

Panel Members

Dr. Paul A. Fishwick
Department of Computer and Information Science
University of Florida Gainesville, FL

Dr. Richard B. Modjeski
U.S. Army Concepts Analysis Agency
8120 Woodmont Avenue
Bethesda, Maryland

Dr. Bernard P. Ziegler
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ

Dr. Ramana Reddy, Director
Artificial Intelligence Laboratory
Department of Statistics and Computer Science
West Virginia University
Morgantown, WV

Marilyn Stelzner
IntelliCorp
1974 El Camino Real West
Mountain View, CA

Dr. Malcolm R. Railey, Chairman
BDM Corporation
7915 Jones Branch Drive
McLean, VA

SECOND WORKSHOP ON AI AND SIMULATION

8:00 - 8:15: Coffee and Donuts

8:15 - 8:30:

INTRODUCTION: Dr. Malcolm R. Railey, Chair

8:30 - 10:30:

IMPLEMENTATION

"ISIM: Towards an Integration of Artificial Intelligence and Simulation," Roy Masrani and Sheila McIlraith.

"Categorizing Process Abstraction in Simulation Modeling," Paul A. Fishwick.

METHODS

"Simulations in a Model-world Based on an Algebraic Approach," Charles Chiu.

"A Time Window Approach to Scheduling Simulation Events on a Parallel Processor," Alexis Wieland.

10:30 - 10:45: BREAK

10:45 - 11:45:

"QSOPS: An Integrated Knowledge-Based Environment for the Qualitative Simulation of Physical Systems," Alfred D. Round.

STRATEGIES

"Goal Directed Simulation," Marc R. Halley and Daniel Pliske.

11:45 - 1:00: LUNCH

1:00 - 3:00:

SYNTHESIS

"Intelligent Model Modification," Norman R. Nielsen.

"Modular Model-Based Simulation," Allen S. Matsumoto, Charles P. Kollar, Gary A. Strohm, and Arvind Sathi.

"Dynamic Planning Under Uncertainty Using Automated Model Construction and Risk Analysis," Louis Anthony Cox, Jr. and Richard Blumenthal.

ANALYSIS

"A Natural Language Interface for Simulation of Multistage Production Distribution Systems," Behrokh Khoshnevis and Wanda Austin.

"Stochastic Soft Simulation in Geological Exploration," Suresh G. Thadani, Francois Alabert, and Andre G. Journel.

3:00 - 3:15: BREAK

3:15 - 4:30:

"CONFIG Project: A Generic Tool for Qualitative-Simulation-Based Reasoning about Configurations of Engineered Systems," Jane T. Malin, Bryan D. Basham, and Rick Harris.

"Fault Diagnosis Based on Quantitative Models," Stefan Feyock.

"Simulation and Expert Systems for Finding Particle Beam Line Errors," Lawrence Selig, Scott Clearwater, Martin Lee, and Robert Engelmore.

"Reasoning about Diagnosis and Treatment in a Causal Medical Model Using Semi-quantitative Simulation and Inference," Lawrence E. Widman, M.D., Ph.D.

4:30 - Close:

Panel Discussion:

"What is the value added by combining AI and simulation and where do we go from here?", Dr. Paul Fishwick, Dr. Ramana Reddy, Dr. Richard Modjeski, Marilyn Steltzner, Dr. Ziegler, and Dr. Malcolm Railey.

SECOND WORKSHOP ON AI AND SIMULATION

Panel Members

Dr. Paul A. Fishwick
Department of Computer and Information Science
University of Florida Gainesville, FL

Dr. Richard B. Modjeski
U.S. Army Concepts Analysis Agency
8120 Woodmont Avenue
Bethesda, Maryland

Dr. Bernard P. Ziegler
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ

Dr. Ramana Reddy, Director
Artificial Intelligence Laboratory
Department of Statistics and Computer Science
West Virginia University
Morgantown, WV

Marilyn Stelzner
IntelliCorp
1974 El Camino Real West
Mountain View, CA

Dr. Malcolm R. Railey, Chairman
BDM Corporation
7915 Jones Branch Drive
McLean, VA

SECOND WORKSHOP ON AI AND SIMULATION

[3] "A Natural Language Interface for Simulation of Multistage Production Distribution Systems," Behrokh Khoshnevis and Wanda Austin.

[5] "Design of an Automatic System for Failure Analysis in Integrated Circuits," P. Mauri, M.Piccoli, and P. Mussio.

[7] "Stochastic Soft Simulation in Geological Exploration," Suresh G. Thadani, Francois Alabert, and Andre G. Journel.

[8] "Explanation Capabilities for Intelligent Manufacturing System Simulation," Peter Floss and Joseph Talavage.

[12] "Developing An Intelligent Knowledge-Based Simulation Agent," Joe Dombroski.

[13] "Object-oriented Simulation for Just-in-time Inventory Management", William Faught.

[15] "A Time Window Approach to Scheduling Simulation Events on a Parallel Processor," Duke P. Briscoe and Lisa M. Sokol.

[17] "The Role of Simulation in the Development of an Avionic Expert System," Sylvia P. Darnall.

[21] "AI and Simulation in Materials Science," Ralph J. Harrison.

[23] "Intelligent Model Modification," Norman R. Nielsen.

[24] "An AI-Based Tool for Simulation Data Base Integrity," Dennis W. Cooper.

[25] "Modular Model-Based Simulation," Allen S. Matsumoto, Charles P. Kollar, Gary A. Strohm, and Arvind Sathi.

[26] "Automatic Model Generation for Mechanical Devices," Andrew Gelsey.

SECOND WORKSHOP ON AI AND SIMULATION

[27] "Categorizing Process Abstraction in Simulation Modeling," Paul A. Fishwick.

[31] "Event Horizons in Artificial Intelligence Systems," Tad Hogg, Bernardo, A. Huberman, and Jeff Shrager.

[32] "Reasoning about Diagnosis and Treatment in a Causal Medical Model Using Semi-quantitative Simulation and Inference," Lawrence E. Widman, M.D., Ph.D.

[35] "ISIM: Towards an Integration of Artificial Intelligence and Simulation," Roy Masrani and Sheila McIlraith.

[36] "Model Generation from Specifications for Machining," Arie Ben-David, Leon Sterling, and Josph A. Kovach.

[38] "A Simulation-Based Prediction Mechanism for An Expert Factory Control System," Szu-Yung David Wu.

[39] "Miniworld Simulator for Machine Learning Systems," Dan Patterson.

[43] "A Distributed Simulation Environment - The Intelligent Vehicle Workstation," Kevin J. Lehnert, Daniel M. Donahue, Stanley K. Hill, Marion Lineberry, and Michael Sullivan.

[44] "QSOPS: An Integrated Knowledge-Based Environment for the Qualitative Simulation of Physical Systems," Alfred D. Round.

[45] "Simulation and Knowledge Systems," Michael Greenberg and Paul Cohen.

[46] "CONFIG Project: A Generic Tool for Qualitative-Simulation-Based Reasoning about Configurations of Engineered Systems," Jane T. Malin, Bryan D. Basham, and Rick Harris.

[48] "Knowledge-Based Simulation of Tactical Adversaries," Yee-yeen Chu and Azad M. Madni.

[49] "Transformational Approach to Automated Causal Model Generation," C.N. Lee, P. Liu, M.Y. Chiu, and S.J. Clark.

[52] "Goal Directed Simulation," Marc R. Halley and Daniel Pliske.

[53] "Parallel Marker Propagation System Construction Set," Howard Schneider.

SECOND WORKSHOP ON AI AND SIMULATION

[59] "Simulations in a Model-world Based on an Algebraic Approach," Charles Chiu.

[60] "Dynamic Planning Under Uncertainty Using Automated Model Construction and Risk Analysis," Louis Anthony Cox, Jr. and Richard Blumenthal.

[61] "Fault Diagnosis Based on Quantitative Models," Stefan Feyock.

[62] "Simulation of Chemical Processes in Preliminary Design: Model-Construction, Abstraction, and Control of Information-Flow," Theodore Kritikos, Michael L. Mavrovouniotis, and George Stephanopoulos.

[65] "Simulating the Behavior of Complex Devices for Model-Based Troubleshooting," Walter Hamscher.

[66] "Benchmarks for Research in Planning," Thomas Dean.

[67] "Simulation and Expert Systems for Finding Particle Beam Line Errors," Lawrence Selig, Scott Clearwater, Martin Lee, and Robert Engelmore.

[69] "Causality in Simulation," Stefan Bernemann and Bernd Hellingrath.

[72] Problem Solving Coupling Interpretation of Naive Physics Simulations Based on Analogical Representation and Logical Inferences," Francesco Gardin and Hadley Taylor.

[77] "New Knowledge Representations for Object Oriented Simulation," Steven C. Bankes.

A NATURAL LANGUAGE INTERFACE FOR SIMULATION OF MULTISTAGE PRODUCTION DISTRIBUTION SYSTEMS

Behrokh Khoshnevis and Wanda Austin
University of Southern California
Los Angeles, Ca 90089-1452

[3]

Artificial intelligence and simulation can be combined to provide managers with sophisticated computer aided systems engineering tools to support system analysis and macro level decision analysis. This abstract describes a cognitive simulation environment that has been developed for analysis of multistage production-distribution systems. It uses a combination of natural language understanding and expert systems technologies to generate DYNAMO codes for simulation modeling of systems represented in natural language.

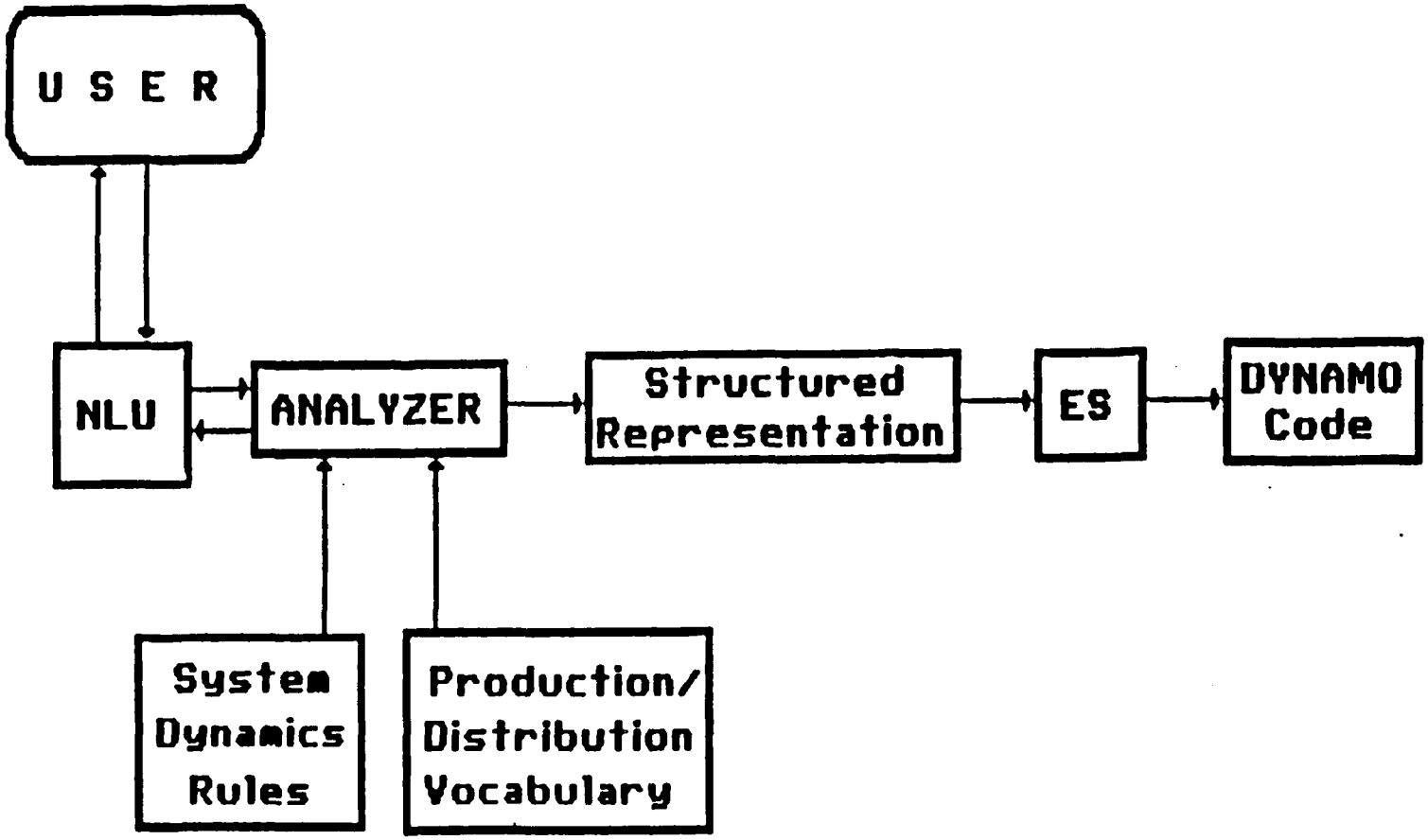
Model formulation is one of the most difficult tasks in building a simulation model. In most cases, there is a vast difference between how the user conceptualizes a problem and how the problem will ultimately be represented in the simulation model. By providing the user with a natural language interface, emphasis can be placed on problem definition and analysis of results rather than problem representation or the intricacies of a specific simulation language. Other advantages for the user are that more complex models can be developed by "non-programmers" and the human to computer interface requires no special training because it is natural language.

Simulation is an excellent and established analysis tool but it requires expertise in experimental design, probability, statistics, modeling, programming and in the problem domain. Engineers and managers need the capability to build simulation models correctly and easily without elaborate training in all of these areas. By combining artificial intelligence and simulation, an advanced knowledge processing system can be created which performs goal processing and goal realization by acquiring, analyzing, transforming and generating information.

An overview of the system is shown in Figure 1. The input to the system is a narrative description of the production-distribution system to be analyzed. The natural language understanding system uses the PhRasal ANalyzer developed by Wilensky and Arens. The knowledge base of the understanding system is restricted to vocabulary and pattern-concept pairs which are common to production-distribution systems. The knowledge base also has embedded knowledge about simulation for interpreting user plans. The system dynamics methodology, as developed by Forrester, provides the framework for representing the complex feedback relationships present in production-distribution systems. The structures for representing material and information flow are well defined and well behaved. The result is an object oriented structured representation of the system and system parameters. This representation is independent of a specific simulation language but forms the knowledge base for an expert system which automatically generates DYNAMO simulation code. The system is implemented in Lisp on a Vax 11/780. Samples of inputs and processed outputs will be provided at the workshop.

Simulation is a good way to study the behavior of complex systems which should not be manipulated in the real world or which are mathematically intractable. This cognitive simulation environment is designed to address some of the problems associated with conventional simulation practices. The research has also provided significant results into generalized systems analysis and problem solving. It also demonstrates the importance of building a reasoning system and knowledge base which support a natural language interface.

SYSTEM DESCRIPTION



References

1. Doukidis, G., R. Paul, "Experiences in automating the formulation of discrete event simulation models", in AI Applied to Simulation, Kerckhoffs, Vansteenkiste, Ziegler, eds., SCS Simulation Series, Vol 18 No. 1, 1986 pp 79-90.

2. Forrester, J., Industrial Dynamics, MIT 1961.

3. Groen, A. et. al, "The integration of simulation with knowledge-based systems", in AI Applied to Simulation, Kerckhoffs, Vansteenkiste, Ziegler, eds., SCS Simulation Series, Vol 18 No. 1, 1986 pp 189-197.

4. Heidorn, G., "English as a very high level language for simulation programming", Proc Symposium on Very High Level Languages, SIGPLAN Notices, 9, April 1974, 91-100.

5. Hendrix, G., "Natural language interface", in Natural Language Processing Tutorial No. 3, AAAI, 1983.

6. Henriksen, J., "The integrated simulation environment", Operations Research, 31, pp 1053-1072.

7. Khoshnevis, B., A. Chen, "An expert simulation model builder", Proceedings of SCS MultiConference on Intelligent Simulation Environments, 1986, pp 129-132.

8. Lehman, A., B. Knodler, E. Kivee, H. Szczerbicka , "Dialogue-oriented and knowledge-based modelling in a typical PC environment", in AI Applied to Simulation, Kerckhoffs, Vansteenkiste, Ziegler, eds., SCS Simulation Series, Vol 18 No. 1, 1986 pp 91-96.

9. Luker, P.A., "Modeller: Computer-assisted modelling of continuous systems", Simulation, 42:5 (May 1984) pp 205-214.

10. Mathewson, S. C., "Simulation program generators", Simulation, 23:6 (Dec 1974) pp 181-189.

11. Muetzelfeldt, R., A. Bundy, M.Uschold, D. Robertson, "ECO - An intelligent front end for ecological modelling", in AI Applied to Simulation, Kerckhoffs, Vansteenkiste, Ziegler, eds., SCS Simulation Series, Vol 18 No. 1, 1986 pp 67-70.

12. Oren, T., et al, "Architecture of MAGEST: a knowledge-based modelling and simulation system", Simulation in Research and Development, Elsevier Science Publishers (North-Holland).

13. Reddy, R., et al, "KBS: a knowledge based simulation system", Research Report, Carnegie Mellon University, 1985.

14. Reddy, R., "Epistemology of knowledge based simulation", Simulation , April 1987.

15. Reilly, K., W. Jones, P. Dey, "The simulation enviornment concept: artificial intelligence perspectives", Artificial Intelligence and Simulation, W. Holmes ed., SCS 1985.

16. Rich, E., Artificial Intelligence, McGraw Hill, 1983.

17. Rich, E., "Natural language interfaces", Computer, September, 1984, pp 39-47.

18. Richardson, G., A. Pugh III, Introduction to Systems Dynamics Modeling with Dynamo, MIT Press 1981.

19. Samad, T., S. Director , "Towards an natural language interface for CAD", Proceedings 22nd ACM/IEEE Design Automation Conference, June 1985 pp 2 - 8.

20. Shannon, R., R. Mayer, H. Adelsberger, "Expert systems and simulation", Simulation 44:6, pp 275-284.

21. Wilensky, R., Y. Arens, D. Chin, "Talking to UNIX in English: An overview of UC", Communications of the ACM, June 1984 pp 574-593.

22. Winston, P., Artificial Intelligence, Addison-Wesley, 1984, Chap 8.

23. Winston, P., B. Horn, LISP, Addison-Wesley, 1981, Chap 22.

24. Xiong, G., A. Song, "An expert system for dynamic system simulation", in AI Applied to Simulation, Kerckhoffs, Vansteenkiste, Ziegler, eds., SCS Simulation Series, Vol 18 No. 1, 1986 pp 106-110.

[5]

DESIGN OF AN AUTOMATIC SYSTEM FOR FAILURE ANALYSIS IN INTEGRATED CIRCUITS.

P.Mauri,M.Piccoli- S.G.S. Microelettronica S.p.A.
Reliability Laboratory
Via Tolomeo 1-20010 Cornaredo,Italy
P. Mussio -Dept.of Physics,Universita' di Milano
Milan,Italy

ABSTRACT. The aim of this paper is to report some exploratory steps performed in the design of an Automatic Assistant (A.A.) for Failure Analysis on Integrated Circuits, in particular for the diagnosis of intrinsic parasitic elements (1).

Failure analysis on integrated circuits is a diagnosis activity involving a great amount of different knowledge and skills (electronic, physics, use of microscopes and so on). Goal of this activity is finding failures sites, causes and mechanisms.This search is based on typical diagnosis procedure: definition of hypothesis,experiment design and subsequent verification.

The AA is designed to help the failure analyst in his activity of circuit diagnosis, increasing his overall effectiveness as a human problem solver. The AA has therefore to mimic some clerical activities usually performed by the analyst in his model definition and verification. In other words AA has the task to relieve the analyst from the repetitive analysis of what is well known or what can be derived by simple inference procedure from what is well known.

AA exploits three level of knowledge.

First it takes advantage of the surface-level knowledge (2,3), i.e. knowledge derived from the practical experience in the failure analysis of IC, which is not supported by some explicit physical or electronic model, even if it has overall coherence and takes into account implicitly physical laws.

Second, deep-level knowledge is used in which physical models are exploited in a qualitative approach i.e. the physical reasoning is based even on non numerical judgments.

Last, knowledge at the algorithmic level is used when it is possible (and necessary) to take advantage of numerical simulation for example of well defined parasitic elements.This last knowledge is codified in SPICE database(4,2).

So the main components of the system are (see annexed figure):

- an expert module based on the production system paradigm which encodes expert surface level of knowledge about technological process, physical structure and typical failure modality.

  The output of this module is the definition of parasitic elements related to physical structure and of degradation type related to work environment of device.

- a module oriented to qualitative physics for qualitative simulation (7,8,9) employing an archive of models of deep-level knowledge. It uses as input the information of former module and the circuit description to realize functional analysis in order to select among possible parasitic elements , for example, the most responsible ones. The models translate the possible circuit components behaviours into causal rules. The simulator uses these rules and expert interpretation of classical physics laws to perform analysis of damaged circuits.

- a module able to execute numerical simulation of some parts of the circuit (SPICE and a Circuit Modifier) whenever the qualitative simulator requests it.

Purpose of the integration of qualitative and quantitative simulation is to solve the problem of intrinsic ambiguity related to qualitative physics interpretations.

REFERENCES

[1] Bertotti,F. e Garue,S. Circuiti integrati bipolari,Zanichelli editore 1985

[2] Pan,Y Qualitative reasoning with deep-level mechanism models for diagnoses of mechanism failures IEEE Trans.Comp. 1984

[3] Pan,Y. and Tenenbaum,J.M. PIES: an Engineer's Do-it-yourself knowledge system for interpretation of parametric test data, AI Magazine FALL 86

[4] Ruehli,A.E. Circuit Analysis, Simulation and Design, North-Holland 1986

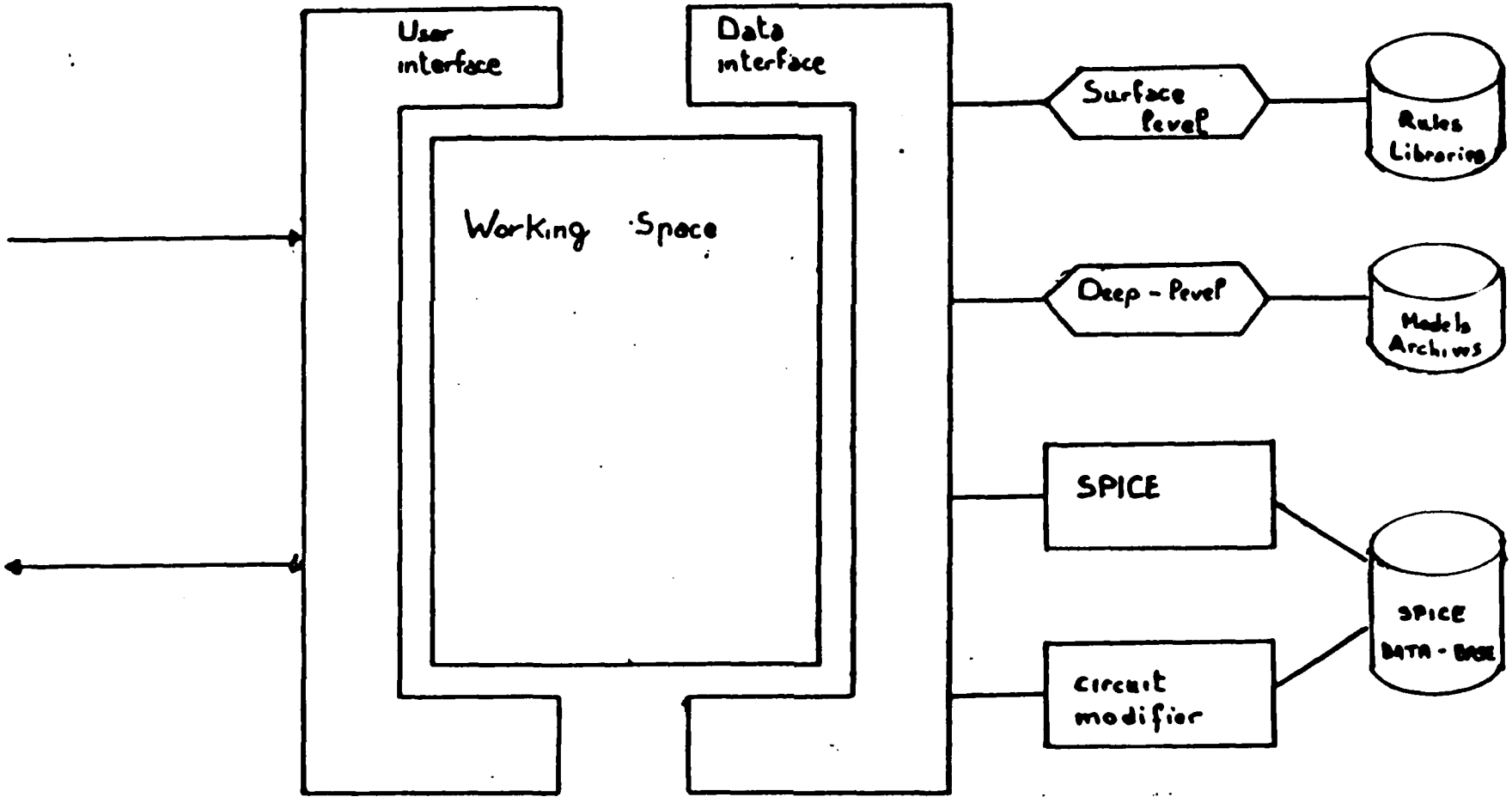
[5] Chandrasekaran,B. and Milne,R. Reasoning about structure behaviour and function, SIGART Newsletter, July 1985

[6] De Jong,K. Expert system for Diagnosing complex system failures, SIGART Newsletter, July 1985

[7] Kuipers,B. Commonsense reasoning about causality: deriving behaviour from structure, AI Journal 24, 1984

[8] De Kleer,J. How circuit works, AI Journal 24, 1984

[9] Williams,B.C. qualitative Analysis of MOS Circuits, AI Journal 24,1984


User interface
Data interface
Working Space
Surface level
Deep - level
SPICE
circuit modifier
Rules Libraries
Models Archives
SPICE DATA - BASE

[7]

Stochastic Soft Simulation in Geological Exploration

Suresh G. Thadani. Standard Oil Production Co
Francois Alabert. Stanford University
Andre G. Journel. Stanford University

Abstract

The characterization of the spatial variation of subsurface (reservoir) properties is a key problem in geological exploration for petroleum. The problem basically is characterized by the availability of three distinct types of information. These are (a) a set (usually sparse) of relatively precise well log measurements; (b) a set (usually copious) of indirect and less precise seismic trace information that covers the region of interest; and (c) subjective (expert) geological information about the region. The subjective type (c) information is available partly as inequality constraints on the property of interest and partly as geological heuristics that serve to constrain the set of allowable characterizations.

In applications of this type, there is a critical need for quantifiable and reliable measures of the uncertainty associated with specific characterizations of spatial variability. This paper presents a stochastic simulation approach to the problem that integrates the three types of information using a combination of Geostatistical, Pattern Recognition and AI techniques. In this approach, the spatial property to be investigated (eg. porosity of a geological unit) is modeled as a realization of a stationary spatial stochastic process. The first and second order spatial statistics of the above stochastic process model ie. the mean and spatial covariance, are estimated by using *both* the "hard" type (a) information and the "soft" type (b) and (c) information. More precisely, Statistical Pattern Recognition techniques are are used to infer distributional bounds on the property of interest at selected spatial locations in the prospect. These bounds are obtained by statistically calibrating the indirect and relatively imprecise seismic trace data to the precise and direct well-log measurements. The "hard" information, distributional bounds, and inequality constraints provided by the subjective information, are incorporated explicitly in the estimation process alluded to above.

Stochastic "soft" simulations consist in generating large numbers of equiprobable realizations of the above spatial stochastic process model. Each of these realizations can be shown to (a) honor the "hard" data , (b) honor the "soft" data, and (c) be consistent with the first and second order statistics of the stochastic model. The use of "hard" and "soft" information in the simulation process provides for much more reliable estimation of the statistics of the underlying stochastic model, while at the same time constraining the resulting realizations more realistically than if only "hard" information were used. As a final pruning step, "soft" information in the form of geological heuristics can, if available, be used to qualitatively filter out realizations that are geologically not allowable - here AI techniques can be used to implement the qualitative filters.

The simulated spatial realizations are usually displayed in the form of contour maps. As an illustration, (soft) simulated maps for the porosity in a selected geological unit of a prospect are illustrated in figures 1a and 1b. The map in figure 1a was generated using only "hard" information at 12 well locations in the prospect, while the map in figure 1b was generated using both the "hard" information above and "soft" information in the form of distributional bounds at 20 grid locations (grid size 10 x 10) . The differences between the "hard" and "soft" maps are significant and illustrate the effect of including "soft" information in the simulation. The analysis of such spatial realizations can provide valuable information on the uncertainty associated with the spatial variability of the properties being investigated. For example, the risk associated with property estimates at any given spatial location can be quantified by constructing emperical histograms of these estimates across the multiple realizations. In particular, the magnitude of the risk is determined by the amount and quality of all the available information.

Preliminary results of this research have demonstrated how "hard" and "soft" information can be successfully integrated into general soft simulation algorithms for characterizing the spatial variability of subsurface properties. While developed specifically for geological applications, soft simulation techniques are general enough to be applicable to other domains that are characterized by information of varying precision.

Acknowledgements

We are grateful to Standard Oil Production Company for permission to publish this paper.

References

GEMAN, S. AND GEMAN, D., 1984 Stochastic relaxations, Gibbs Distributions and the Bayesian Restoration of Images", IEEE Trans. Pattern Analysis and Machine Intelligence 6 (6)

JOURNEL, A.G. AND HUIJBREGTS, C., 1978, Mining Geostatistics, Academic Press, 491-498.

RIPLEY, B.D. 1981 Spatial Statistics, Wiley, New York

SHERIFF., 1973, Factors affecting seismic amplitudes- A review of physical principles Lithology and direct detection of hydrocarbons using geophysical methods, Symposium of the Geophysical Society of Houston,3-17. See also Geophysical Prospecting 23 (1975),125-138.

THADANI, S.G. 1978 Learning with an Affine Teacher, 4'th International Joint Conference on Pattern Recognition, Kyoto, Japan, 1978

Δz = 8000 *feet*

Δy = 5000 *feet*

● : wells

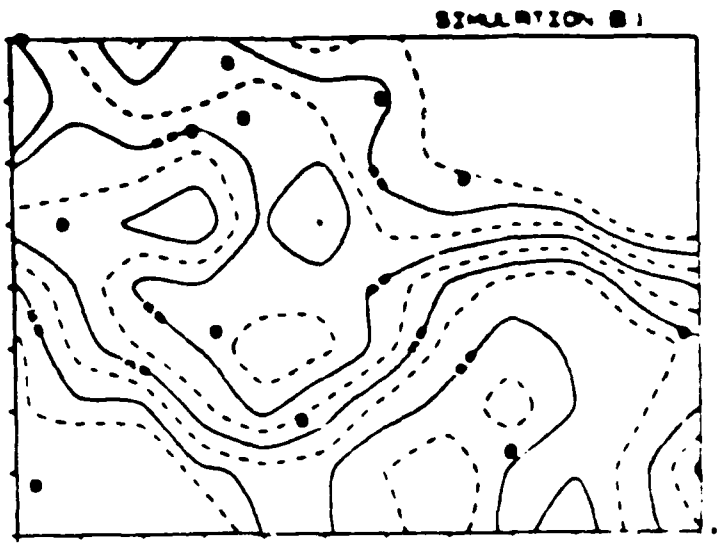


Figure 1a. Conditional Simulation : "hard" data only

Δz = 5000 *feet*

Δy = 5000 *feet*

● : wells

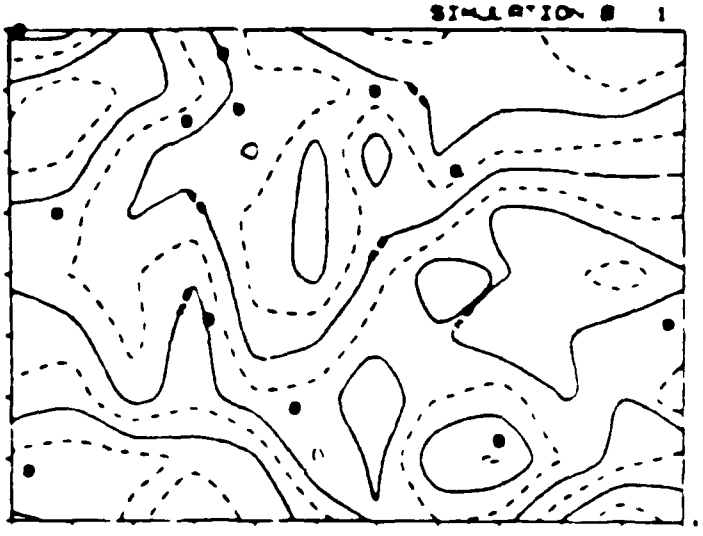


Figure 1b. Conditional Simulation : "hard" data and "soft" data

[8]

Explanation Capabilities for Intelligent Manufacturing System Simulation

Peter Floss and Joseph Talavage
Center for Intelligent Manufacturing Systems
Purdue University
West Lafayette, IN 47907

Extended Abstract

While discrete system simulation is a powerful and widely accepted tool for analyzing manufacturing systems, understanding the decisions made (e.g., with regard to scheduling) in complex models is a difficult task even for the user experienced in simulation methodology. A user, inexperienced in computer simulation, trying to gain insight into a manufacturing systems behavior, especially the complex intelligent manufacturing systems of the future, requires additional information beyond the traditional simulation trace. This paper describes an explanation facility, implemented in a Prolog network simulation language, for decisions made in manufacturing system simulations.

Simulation in Prolog

Network oriented simulation languages simplify the modeling process through their ease of use, but generally are inconvenient vehicles to model intelligent behavior such as complex decision making. The Prolog network simulation language developed here provides the capability to handle complex behavior through the logic programming paradigm along with having the usual advantages of the network orientation.

A network node in the simulation language consists of a set of clauses with the same functor. The language stores the event calender as an ordered set of facts in the Prolog database.

Explanation Facility

The concept of explanation systems developed out of research done in the 1970s on artificial intelligence and rule based expert systems in particular. The early explanation systems merely consisted of a trace of each rule used within the reasoning process. While explanation facilities have

been implemented in specific simulation models, this paper generalizes the capability to a simulation language.

The implemented explanation facility allows an inexperienced user to analyze system model behavior without extensive knowledge about system simulation. The explanation facility provides the user with insight into decisions made within the model, including:

1. decisions about part flow within the system
2. decisions which cause a lack of flow for a part in the system
3. decisions about resource allocations

The explanation system stores a partial trace of the simulation run within the Prolog database and utilizes the relevant portions of the trace to explain the specific decisions being made. The explanations can be provided continously over the simulation run or only as requested by the user at simulation interrupts.

Conclusion

Explanation facilities implemented within a simulation language extend the capability of the language and the power of simulation as a tool for gaining insight into system behavior. This additional capability is particularly useful for the naive user of system simulation.

Developing An Intelligent Knowledge-Based Simulation Agent

[12]

Joe Dombroski
The MITRE Corporation, McLean, Virginia

Abstract

While current object-oriented simulation techniques are effective in representing simulation agents as objects, they are often lacking in facilities that allow the user to understand the behavior of those simulation agents. A simulation agent's behavior is usually only understood by tracing the messages that are sent and received by the corresponding simulation object behaviors. This means that despite the use of the object-oriented model, an agent's behavior is often spread throughout the simulation and is thus difficult to capture. We believe that this problem might be solved by introducing an agent level planning function to carry out much of the processing done by a simulation agent.

Some past systems have used a planning function to create a mission plan and have then executed that plan in an object-oriented simulation. Such systems typically monitor executions and report any plan step violations to the user. In this way, simulation action can be interpreted in terms of the plan that is being executed rather than as sets of distinct actions carried out by object behaviors. However, even with this approach, it is difficult to relate an individual agent's behavior to the plan steps that are being executed. In order to do this, we need to integrate the planning function at the agent level. That is, each agent is modeled as possessing the planning expertise to act on the set of goals which are in that agent's domain. Rather than responding to incoming messages with a set of already defined behaviors, the agent uses its planning mechanism to process goals set for it by other simulation agents. The aim of agent level planning is to create a model for an autonomous simulation agent that is able to both monitor and re-configure its plans relative to a dynamic simulation environment. This approach has potential benefits for both knowledge-based simulation user explanation and control functions since the goals that agent processes can be viewed both locally with respect to the agent and globally by considering how an agent's goals relate to a larger plan.

ABSTRACT

OBJECT-ORIENTED SIMULATION FOR JUST-IN-TIME INVENTORY MANAGEMENT

William Faught, Ph.D
Director of Advanced Product Development

IntelliCorp
1975 El Camino Real West
Mountain View, CA 94040
415-965-5500

IntelliCorp and Unisys (formerly Sperry) Corporation have jointly developed a system that allows manufacturing engineers to simulate just-in-time (JIT) inventory management techniques in a mainframe computer manufacturing facility. The system is built in SimKit, a set of tools that facilitate the development of knowledge-based, object-oriented simulations. SimKit is, in turn, built in and on KEE, IntelliCorp's knowledge-based system development environment.

Historically, simulation has had an important role in the design process, yet many times simulation results have been received with scepticism. While pointing to models as proof of concept, a great many end-users of simulation results have relied on their own judgement and experience for the actual design evaluation. Manual design evaluation is adequate in simple systems, but it likely to result in miscalculations in more complex applications. This is not to say that the use of simulation is unimportant at many stages of the design, implementation, and even use of large complex systems. Avoidance of simulation and discreditation of the resiults is more a commentary on the fact that existing simulation languages are difficult to use and validate and even more difficult to explain in convincing terms to persons not versed in the use of the given simulation language.

SimKit addresses these problems by providing a flexible environment for model construction and modification. Inheritance capabilities allow the developer to construct a master template for generic objects which include the objects' associated behavior. Model construction and modification consists of creating and manipulating members of these generic object classes, graphically. The requirements for alternative pull and push protocols between queueing stations for JIT are addressed with an object-oriented method language for behavior. The interactive nature of the language in conjunction with the above attributes provides a rapid prototyping power that allows developers and so-called "naive" end users to rapidly generate and test many alternatives and model very complex behavior.

A TIME WINDOW APPROACH TO SCHEDULING SIMULATION EVENTS ON A PARALLEL PROCESSOR

[15]

Duke P. Briscoe
Lisa M. Sokol

THE MITRE CORPORATION
7525 Colshire Drive
McLean, Virginia 22102
(703) 883-7855

ABSTRACT

Parallel computers have been successfully used to achieve greater execution speed for applications which have a simple, regular decomposition into parallel processes. The same improvements have been more difficult to achieve for complex, irregularly structured applications. Simulations pose particularly difficult problems for a parallel implementation because the inherent sequentiality required by causality must be preserved.

New approaches to discrete-event simulation are needed in order to generate enough concurrent simulation tasks to efficiently use parallel processors. One way of parallelizing a sequential discrete-event simulation is to relax some of the precedence constraints implicit in the chronological ordering of simulation events. Inappropriate constraint relaxation can cause the distortion of causality relationships between events, which will affect the simulation's fidelity. We have designed a variant of discrete-event simulation which uses a moving time window (MTW) to control the relaxation of precedence constraints and to limit the loss of fidelity. MTW is based on the familiar concepts of event-driven and time-stepped simulations, but it trys to overcome the deficiencies which those two approaches have for a parallel implementation. It is similar to the event-driven approach in that time progresses in jumps from event to event rather than by a fixed interval. It resembles the time-stepped approach in that all events within the interval of the time window are considered to be executable concurrently, except that the MTW adds a number of constraints on event execution order which act to preserve fidelity and causality. MTW further modifies the original time-stepped concept in that when the earliest scheduled event within the time interval of the window terminates, the time window can be adjusted to begin at the next scheduled event and end some fixed time later. The advantage of this is that it will bring a steady stream of executable events into the window if there is not a prolonged quiescent period in the simulation. Those events with times of occurrence within the interval of the time window can be scheduled for parallel execution, subject to the previously mentioned set of constraints.

In order to support our simulation work, we developed an object-oriented language, called Possum, for use with the BBN Butterfly Lisp programming environment. Possum objects can have read/write locks for their instance variables. A function can be rapidly mapped in parallel over the descendants of a class; parallel tasks are created in correspondence to the branching of the object inheritance hierarchy. Messages in Possum can be stated in an English-like format, which should help make the simulation code more readable. Possum has been used to implement the MTW algorithm.

REFERENCES

Lisa Sokol and Duke Briscoe, "A Time Window Solution of Scheduling Simulation Events on Parallel Processors," unpublished paper

Lisa Sokol and Duke Briscoe, "Object-oriented Simulation on a Shared Memory Parallel Architecture," Proceedings of the 1986 IEEE Expert Systems in Government Symposium, pp. 446 - 449

Sylvia P. Darnall
General Dynamics Fort Worth Division
P.O. Box 748 MZ 1772
Fort Worth, Texas 76101

Prepared for The Second Workshop on AI and Simulation AAAI-87, July 14, 1987, Seattle, Washington

THE ROLE OF SIMULATION IN THE DEVELOPMENT OF AN AVIONIC EXPERT SYSTEM

Testing a maturing knowledge-based system through increasingly rigorous levels of simulation facilitates the evolution of a rapidly prototyped system toward a fielded embedded system. Levels of simulation were developed for validating the expert system through its phased development. The particular expert system goal is the design and development of a real-time embeded avionic knowldege-based system which recommends the appropriate self-defense countermeasures in response to threats to the aircraft.

During the very early phases in the development of the expert system, menu-driven static situations were sufficient to test the rules and knowledge representations. As the system grew, it became necessary to dynamically model the informational complexity of an avionic environment. This environment simulator is itself a large Symbolics* hosted knowledge-based system comprised of multiple cooperating smaller knowledge-based systems. At this level, the simulation architecture involved frame-based hierarchical representation techniques with dynamic inheritance relationships. For our purpose a pure software simulator provided the flexibility and user friendliness necessary to the early testing and evaluation of system design concepts, as well as the incremental development of software functionality. The simulator also grew incrementally in order to support the developing expert sytem. The functionality was as generic as possible to avoid an airplane specific point design for the testbed. This allowed practical human interface to the prototype expert sytem for a variety of domain experts. Although this simulation approach is the most flexible and least expensive, it does lack overall credibility as a validation tool over other simulation models.

An intermediate test level is the use of a high fidelity hybrid simulation environment in which flight is simulated in software, but the Electronic Warfare model consists of actual Electronic Countermeasures (ECM) hardware in the laboratory. This allows testing the expert system for accuracy and speed in an extremely credible environment. Use of the knowledge-based simulator provided a straightforward method for defining the interface requirements for the hybrid simulation environment. The most credible testing occurs during flight test in which the embedded expert system software is flown across an Electronic Warfare test area. This is the most rigorous testing approach since it uses real hardware and a real aircraft. This is the final simulation level before actually deploying the expert system.

Throughout this project we have investigted the complex problem of validation for expert systems. We generated requirements for levels of simulation for avionics testing. We have applied emerging techniques such as cooperative problem solving, qualitative reasoning, and knowledge-based simulation. The use of a special purpose knowledge-based simulator was essential in the early phases of the development of the expert system. A conventional simulation did not possess the flexibility and degree of user friendliness required for rapid protyping, however as the expert system matured a high fidelity conventional simulator for validating the expert system was needed.

*Symbolics is a trademark of Symbolics, Inc.

AI AND SIMULATION IN MATERIALS SCIENCE

[21]

Ralph J. Harrison
U. S. Army Materials Technology Laboratory
Watertown, MA 02172-0001

From the vantage point of one who is interested in both AI as well as in the simulation of materials behavior, I would like to comment about two subjects relating to simulation and AI. The first subject deals with two apparently distinct comparisons between simulation methods and expert system methodology. One comparison is that probability is treated more simply in simulation; if the probability distribution of the initial conditions is known, the simulation itself will compute the resulting probability distribution for the output variables in a straightforward manner. On the other hand the treatment of probability in expert systems is inherently complex (1-4). This may look at first like a "plus" for simulation. By the same token the second comparison, which is that expert system production rules usually go from observed results to probable causes whereas simulations go the other way and usually cannot be reversed, might be counted as a "minus". Leaving out the details (5) of what I feel is the resolution of both these comparisons, one can say that the failure of simulation techniques to solve the so-called "inverse" problem is not absolute, especially when numerical approximations are considered, while the success of expert systems in this regard also has its qualifications. The essence of the resolution however is that the disparity between the ease of solution of the direct and that of the inverse problem is directly related to the complexity of the treatment of probability in expert systems.

The second subject for comment, relating more specifically to materials simulation, is prompted by the fact that many properties of solids depend upon fine details of microstructure, from atomistic to macroscopic in scale. It is not practical to consistently simulate all properties atomistically, much less to simulate the dynamical motions of all electrons and nuclei in a quantum mechanically exact manner. Multiple levels of abstraction in the simulation are necessary to bridge the many orders of magnitude gaps in scale. This problem is perhaps simpler and better defined for materials than for the general "real world" problem. As illustration, a state-of-the-art model of a crack under tension, used to study fracture, is one which treats the crack tip in atomistic detail while the stress in more remote regions is modeled by a finite element grid (6). Errors in this treatment arise from the omission of all levels between the finest and coarsest in the hierarchy; a level especially needed is one which describes dislocation motion, perhaps using elasticity theory. Difficulties in improving the model arise in interfacing different hierarchical levels while realistically treating interactions spanning a large but not well-separated range of time and spatial scales. Intelligent programs to determine the level of multi-level simulation necessary to answer particular questions, which can "zoom" in on some portion of a system to bring out details (7) and which can represent a hierarchy of production rule sets to simulate processes at several levels of abstraction (8), (9) have not been applied in materials property simulations. Spriet and Vansteenkiste (10) have suggested that for "soft" systems such as biological organisms where hierarchical separation in space, scale or time is difficult, qualitative methods for modeling may be appropriate. Qualitative methods (11) also tend to be more robust than quantitative models and possibly may be devised to give correct results over several levels, although not as accurately as would quantitative models restricted to a given level.

REFERENCES

1. G. C. Vansteenkiste, J. Spriet and J. Bens, "Structure Characterization for System Modeling in Uncertain Environments" Methodology in Systems Modeling and Simulation, B. P. Zeigler, M. S. Elzas, G. J. Klir, T. I. Oren, Eds., (Amsterdam: North-Holland Publishing Company, 1979) 289-303.

2. G. C. Vansteenkiste and J. A. Spriet, "Modeling Ill-Defined Systems," Progress in Modeling and Simulation, F. E. Cellier, Ed., (London, Academic Press, 1982) 11-38.

3. E. H. Shortliffe et al., "Reasoning Under Uncertainty," Part Four, Rule-Based Expert Systems, The MYCIN Experiments of the Stanford Heuristic Programming Project, B. G. Buchanan and E. H. Shortliffe, Eds., (Reading, MA, Addison-Wesley Publishing Co., 1984) 209-292.

4. R. K. Bhatnager and L. N. Kanal, "Handling Uncertain Information: A Review of Numeric and Non-Numeric Methods," Uncertainty in Artificial Intelligence, L. N. Kanal and J. F. Lemmer, Eds., (New York, Elsevier Science Publishing Company, 1986) 3-26.

5. R. J. Harrison, "AI and Simulation in Materials Science Research," Proceedings of Symposium on Applications of AI to Materials Science, R. J. Harrison and L. Roth, Eds., (Cleveland, AIME, 1987).

6. M. Mullins and M. A. Dokainish, "Simulation of the (001) Plane Crack In Alpha-Iron Employing a New Boundary Scheme," Philosophical Magazine A, 1982, Vol. 46, 771-787.

7. S. Ghosh, "On the Concept of Dynamic Multi-Level Simulation," 19th Annual Simulation Symposium, D. L. Kimbler, Ed., (Washington, D.C.: IEEE Computer Society Press, 1986), 205-210.

8. P. A. Fishwick, "Hierarchical Reasoning: Simulating Complex Processes Over Multiple Levels of Abstraction," (University of Pennsylvania, Ph.D. Dissertation, 1986).

9. A. Sydow, "Hierarchical Concepts in Modeling and Simulation," G. C. Vansteenkiste and J. A. Spriet, "Modeling Ill-Defined Systems," Progress in Modeling and Simulation, F. E. Cellier, Ed., (London, Academic Press, 1982) 103-118.

10. J. A. Spriet and G. C. Vansteenkiste, "Computer-Aided Modeling and Simulation," (London: Academic Press Inc., 1982).

11. D. G. Bobrow, Ed. Qualitative Reasoning About Physical Systems, (Cambridge, MA, MIT Press, 1986).

[23]

Intelligent Model Modification

Norman R. Nielsen
Intelligent Systems Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, Calif. 94025
(415) 859-2859
NIELSEN@STRIPE.SRI.COM

ABSTRACT

There are significant opportunities to apply AI-based reasoning techniques in a simulation. Obvious areas of application include the analysis of simulator output data, formulation of parameter or characteristic adjustments, and decision-making within the simulator. However, from the standpoint of the model user, perhaps the most significant impact lies in the area of intelligent model modification.

A useful model is one that undergoes frequent change. However, each change is also an opportunity for errors to be introduced. The basic model development process generally includes a significant testing and verification effort. However, changes made after a model is ready for use are often untested; such changes are for the most part "small" and are made "on the fly." As a result, model results often do not mean what they are thought to mean, because the actual model used was different from the model that the experimenter thought was being used.

Techniques are becoming available to reduce the incidence of such errors. For example, a graphical display of a network model will reduce undetected configuration errors, as humans are more likely to detect an unexpected change (or the absence of the expected change) in a visual display than they are in a text or numerical listing. However, many aspects of a model cannot be portrayed graphically, so other techniques must be employed.

The application of reasoning capabilities to the model modification process can produce significant benefits. Making a change to a characteristic of an object (the attribute of an entity) can be used to trigger the execution of a procedure or rule-set to infer what other changes might be required as a consequence of the initial change. In some cases derivative changes can be made automatically, eliminating the chance that the modeler might make an incomplete set of

changes and thus preventing the creation of dangling pointers, inconsistent values between objects, and the like. In other cases, all of the derivative changes cannot be made automatically, since the complete intention of the modeler cannot be inferred. However, all of the entities and attributes that may need modification can be identified to the modeler, along with suggestions, definitions, help text, and so forth as appropriate. Thus, the system can ensure that all needed changes are made, even though reliance must be placed on the modeler for decisions.

The main problem in implementing such an approach, in addition to creating the modification logic, is gaining control for the reasoner. Implementation is relatively easy to accomplish in a modeling system such as Simkit, because the underlying object-oriented representation of entities permits changes to be trapped. However, the technique can also be applied in conventional languages. For example, the process representation of entities in Simscript II.5 and Simula can be used to execute the modification logic.

At SRI we have built a model of a telecommunications network that is used for experimenting with network fault detection and corrective mechanisms. However, we are also using this model to experiment with the application of intelligence to maintenance issues. We have identified a set of 10 "typical" modifications (ranging from a relatively simple change to the bandwidth of a network link to a more complex change of the congestion detection and response algorithm) that a modeler might wish to make in the course of experimenting with a particular network. We are in the process of investigating the percentage of derivative changes and consistency checks that can reasonably be handled by:

- Model-dependent logic constructed by the model developer for application on a model-wide basis
- Generic logic that might be incorporated as a part of the modeling facility's capability.

References

1. Nielsen, N. R., "Expert Systems and Simulation," in *Computer Networks and Simulation III*, S. Schoemaker (Editor), Elsevier Science Publishers B.V., The Netherlands, 1986.

2. Nielsen, N. R., "Knowledge-Based Simulation Programming," *Proceedings of the 1986 NCC*, Vol. 55, AFIPS Press, Reston, Virginia, 1986.

[24]

AN AI-BASED TOOL FOR SIMULATION DATA BASE INTEGRITY

by

Dennis W. Cooper
Machine Intelligence Laboratory
General Research Corporation
P. O. Box 6770
Santa Barbara, California 93111

Historically, preparing input data for a simulation of any complexity has been a labor intensive process. Much of the labor is devoted to verifying that all the input has been specified and is consistent. Consistency means that the simulation data base has objects and/or attributes which are interrelated. These interrelationships are often times implicit in supporting documentation, even though they are explicit in the simulation code. If the input data developer ignores these interrelationships, erroneous simulation behavior can result, which may be difficult to detect.

One possible solution to this problem is to explicitly capture the nature of the constraints on the data in English-like rule form. These rules can be referred to as data integrity rules (DIRs). Anytime a simulation object is defined or modified, the appropiate DIRs can be referenced by a simple AI rule interrupter to verify consistency. DIRs that fail result in the offending object and attribute description being listed along with the failing DIR. The *IF* part of a DIR contains one or more clauses which describe the necessary conditions for the assertion stated in the *THEN* part of the DIR to hold. In other words, if any clause in the *IF* part evaluates to *FALSE*, the DIR is abandoned. If all the clauses in the *IF* part evaluate to *TRUE*, the *THEN* part is evaluated. If it is *FALSE*, the DIR fails and a report is made.

DIRs come in three types: (1) DIRs that assert the legal values that an attribute can have regardless of anything else, (2) DIRs that assert a dependency between one attribute and another within an object, and (3) DIRs that assert a dependency between an attribute of one object with the attribute(s) of another object. The latter type of DIR introduces an additional complexity in the rule form. That is, the rule must provide sufficient information to permit the rule interrupter to reference the *related* object.

Up to this point, we have described a tool for verifying the integrity of the input to a simulation. It has been our experience that many simulations once instantiated with the input will increase in size by a factor of 3 to 5 before reaching a steady state. This means that the simulation itself creates simulation objects. Can these be verified with our tool? In a very simple way, they can. A snapshot of the simulation data base can be taken and another set of DIRs (those pertaining to the objects created or modified by the simulation code) can be employed.

A prototype tool has been developed for the General Research Corporation simulation system, *SIMSYS*. The performance of the tool will be discussed.

[25]

Modular Model-Based Simulation

Allen S. Matsumoto, Charles P. Kollar, Gary A. Strohm, and Arvind Sathi

Carnegie Group Inc.
650 Commerce Court at Station Square
Pittsburgh, PA 15219

Abstract

Carnegie Group is currently developing a Model-Based Simulation system to provide a modeling capability with which users can easily construct models of manufacturing systems. The system contains a library of extensible components, and models are constructed by connecting them using semantically-defined relations. The user can view the model using several interconnected perspectives, and consider them separately or combined. The model can be executed as a discrete-event simulation, which will provide an approximation of the behavior of the real system.

Our approach is based on Modular Model-Based Simulation[1] with components implemented as Objects[2]. The library components have explicit external interfaces to other components, and encapsulated interiors. The user can connect them with standard well-defined relations to form system models. The set of basic constructs and relations provided were chosen to support factory modeling[3] and are augmented by model building and analysis tools.

The basic classes of library components are *activities*, *resources*, *material*, *plans*, and *goals*. Activities are defined by their resource requirements and their effects upon material. Resources are passive objects in the system which are used by activities. Material are objects which are consumed, modified, and produced by the system. A plan is a partially-ordered set of activities needed to produce a type of material from other materials. A goal is a desired performace criterion for the system; for example, an order is a goal to produce a quantity of material.

We use the mechanism of inheritance to define new classes as specializations of existing classes. In particular, the library components are specializations of the basic classes. Models are assembled by creating instances of library components and specifying the relations among them. That is, a model is a "semantic network" of instances of library components.

The user can view a model from different perspectives. The *Functional* view is given by the set of activities a system is capable of performing. The *Physical* view is defined by the resources present in the system. The *Operational* view consists of the (process) plans of the system. The *Dynamic* view is shown by orders for material flowing through the system over time. The connections among the model components are defined by relations. The relations define the combined model, and can be selectively chosen to project the model into its different perspectives. For example, an activity could be "before" another, and would "require" resources. These different relations can be separately used to consider activities from either operational or physical views.

After defining the structure of a model, its dynamic behavior can be analyzed by executing it using discrete-event simulation. Statistics are collected by attaching standard instruments to model objects. Simulation results are collected in a relational data base for later presentation and analysis. Several different versions of the model can be run and compared.

The complete system is an integration of Semantic Modeling and Discrete-Event Simulation techniques which provides a modeling environment for building and analyzing manufacturing system models. The Knowledge Base has been structured to allow the model to be constructed from components which can be selectively projected onto separate modeling views. The ability to consider the model from these different perspectives increases the model's understandability. The integrated model is then executed as a simulation model to analyze the effects of the interactions among the model components.

References

1. Oren, T.I., "Simulation and Model-Based Methodologies", *NATO ASI Series*, Vol. F10 1984.

2. Stefik, M. and D. Bobrow, "Object-Oriented Programming: Themes and Variations", *AI Magazine*, Vol. 6No. 4 1986.

3. Sathi, N., M.S. Fox, B. Baskaran, J. Bouer., "Simulation Craft: An Artificial Intelligence Approach to the Simulation Life Cycle", *Proceedings of the SCS Summer Simulation Conference*, July 1986.

Automatic Model Generation for Mechanical Devices

{26}

Andrew Gelsey
Dept. of Computer Science, Yale University
New Haven, CT 06511
GELSEY@YALE

[Gelsey 1987] (copy attached) describes a fully implemented program which produces a kinematic analysis of a mechanism directly from a representation of its raw physical structure. The following abstract describes ongoing but not fully implemented research in which the output of kinematic analysis is used for the automatic generation of models for simulation. The kinematic analysis is useful both in step by step simulation of machine behavior and in the detection of behavioral loops.

Figure 1 shows a CAD/CAM solid model of a gasoline engine. The kinematic analyzer uses spatial reasoning and knowledge of machine kinematics to identify the *kinematic pairs* in the machine and the mathematical functions relating the position of any one kinematic pair to any other. Each kinematic pair has only one degree of freedom, so calculating results of forces acting on the machine is simplified since only the component of a force acting in the possible direction of motion of the kinematic pair is significant. Also, the only direction of motion possible for a kinematic pair is that dictated by its geometry, so the range of changes that can occur during one step of a simulation is severely limited and predetermined.

The engine in Figure 1, like many mechanisms, has only one net degree of kinematic freedom. This fact may be automatically determined by composing the mathematical functions relating positions of kinematic pairs to find that for any position of the camshaft (for example), the position of each other kinematic pair will have a definite computable value. Thus the only state variables needed to represent the mechanical portion of the system are its position and velocity. For example during the power stroke of the gasoline engine in Figure 1, at the start of one step of the simulation the engine has a particular position and velocity. In time Δt, the position of the engine (i.e. the changes in values of the parameters specifying the positions of the kinematic pairs) will change by an amount Δx which may be calculated. At this point in the simulation, the gas in the cylinder has a certain pressure. Knowing this pressure and Δx, the energy transferred from the gas to the engine may be calculated. Knowing the new kinetic energy of the mechanical system will give the velocity at the end of the simulation step, and the energy loss and volume change of the gas will yield new values for temperature and pressure of the gas. Thus the information needed for the next step of simulation will be available.

The detection of behavioral loops for this system is straightforward. Since kinematic considerations will force the engine to go through the same series of positions repeatedly, the only question is how the speed

will behave as a function of time. Since frictional losses increase with velocity while positive contributions to the energy of the system don't, the engine may in general be expected to settle down to a steady speed (if it doesn't halt). Thus as soon as the simulation returns to the same position several times with approximately the same velocity (which is easy to detect), a loop will be identified.

Figure 2, which portrays the escapement mechanism of a clock or watch, presents a more difficult simulation problem because the mechanism has more than one kinematic degree of freedom. An escapement regulates the speed of a timepiece by allowing the drive wheel powered by the mainspring to advance by only one tooth for each oscillation of the balance wheel. When the mechanism of Figure 2 is processed by the kinematic analyzer, it is revealed to have three degrees of freedom, as a result of having three kinematically independent subsystems each with one degree of freedom: the balance wheel and its spring, the lever, and the drive wheel and the entire rest of the clock. Thus at each step of the simulation it is necessary to determine how each of the three subsystems will move, and what forces each will exert on the others.

The more difficult part of the problem is to determine what forces are exerted, but once again kinematics simplifies the problem significantly. Typically, the strength of a force matters very little, and accuracy to the nearest order of magnitude is adequate. Direction of the force is more important, but an approximate value is generally acceptable. Consider the lever in the escapement, which is part of a revolute pair having one degree of freedom. The lever is relatively light compared to the forces exerted on it by either the drive wheel or the balance wheel. Typically, if the direction of the force exerted is such as to cause the lever to experience revolute motion, then it will move until the geometry of the situation changes. The details of the forces applied don't matter, and the only question is which of two possible directions the lever will go.

Since kinematic analysis allows us to turn large amounts of geometry into a few numbers, loop detection is simplified. If during a simulation the position and velocity parameter values for the three subsystems return to values they had earlier, then a loop has been found. Since the drive wheel and lever spend most of their time at zero velocity in one of two positions, finding matching points in parameter space to detect the loop becomes fairly straightforward.

References

Andrew Gelsey 1987 Automated Reasoning about Machine Geometry and Kinematics, *Proceedings of the 3rd IEEE Conference on Artificial Intelligence Applications.*

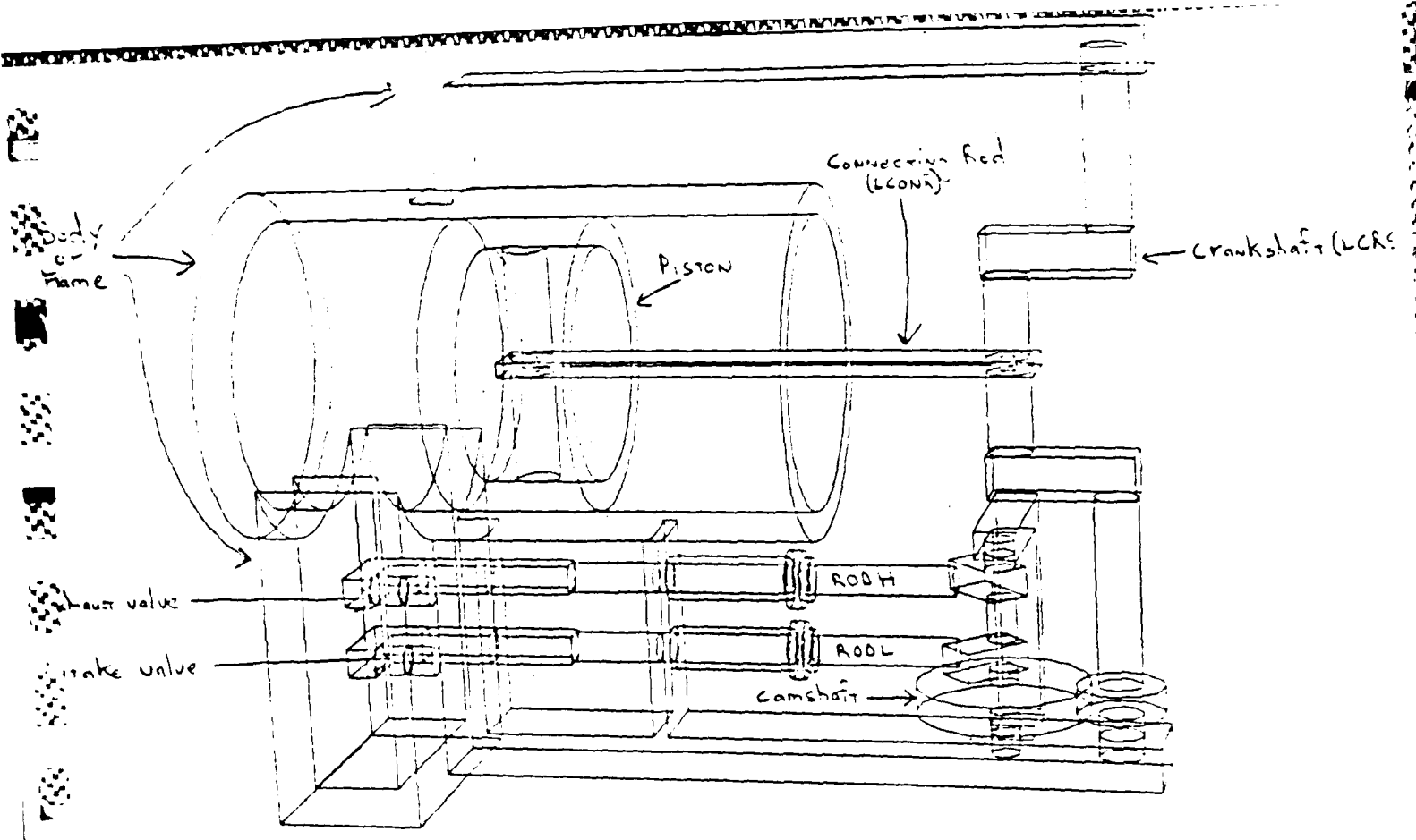


Figure 1 Gasoline lawnmower engine

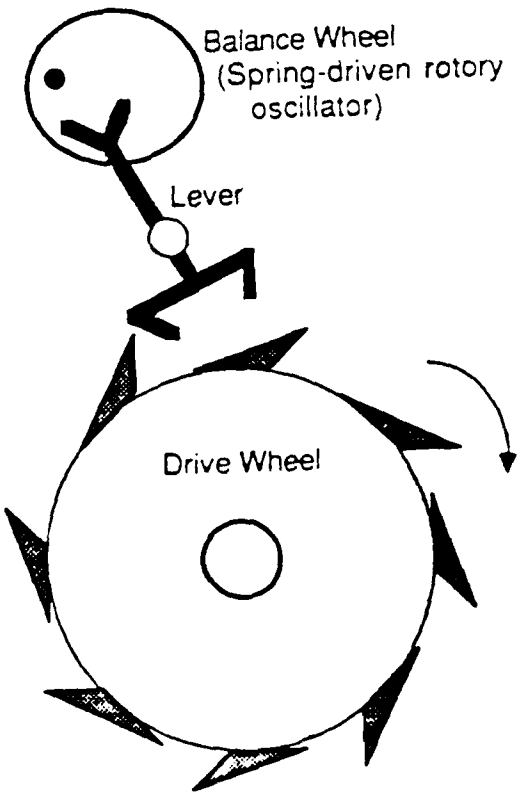


Figure 2 Escapement Mechanism

From: Proceedings of the Third IEEE Conference on Artificial Intelligence Applications

Automated Reasoning about Machine Geometry and Kinematics

Andrew Gelsey
Computer Science Department
Yale University
10 Hillhouse Avenue
New Haven, CT 06520
Arpanet: GELSEY@YALE

Abstract

Reasoning about the physical world has recently become an important AI research topic, but the spatial reasoning aspects have tended to be ignored. I present work on the application of spatial and geometrical reasoning to the problem of reasoning about mechanical devices. My program takes a CAD/CAM solid model of a mechanical device and from that produces a *kinematic analysis* of the device. I illustrate the kinematic analysis process with two examples: a piston/crankshaft mechanism and a differential.

Much recent Artificial Intelligence research has concerned reasoning about the physical world ([Forbus 1984], [DeKleer and Brown 1984], [Kuipers 1984]). However, little of this research has involved a significant amount of geometrical reasoning (but see [Stanfill 1983]). My research focuses on spatial and geometric reasoning as applied to the domain of reasoning about mechanical devices.

My program takes a geometric representation of a mechanical device as input and uses that to produce a "kinematic analysis" of the device which describes what motions the various parts of the machine undergo and the relationships between these motions. Kinematics is the study of the motion of objects without considering the forces that cause the motion. In the design of a mechanical device, the primary means by which the motions of the device are constrained to be those the designer wishes is by appropriate specification of the geometry of the mechanism. Thus, given appropriate knowledge of the machine domain, we may determine the kinematics of a device just by examining its geometry.

Reuleaux [1876] defined a number of concepts which we shall find useful, such as the *kinematic pair* — a pa r of parts which constrain each other's motion. For example, in Figure 1 the crankshaft is paired with the frame in such a way that its only possible motion relative to the frame is rotation about a single axis.

Typically a part will be a member of more than one kinematic pair, so that kinematic pairs will be linked into a *kinematic chain*, as is the case in Figure 1. Reuleaux classified kinematic pairs into two categories: *lower pairs* and *higher pairs*. Lower pairs are kinematic pairs in which contact between the two elements of the pair takes place continuously at all points on a *surface* of contact which

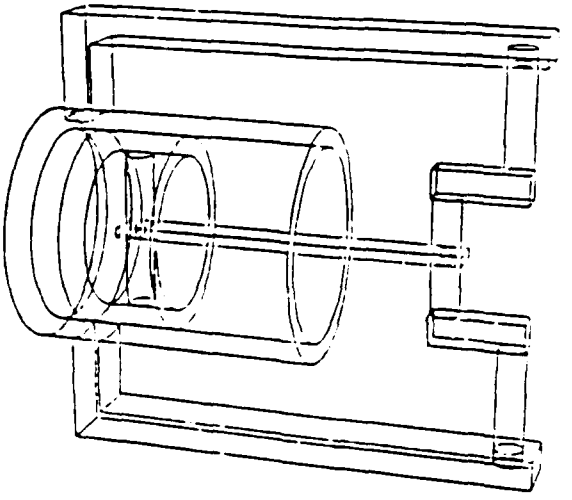


Figure 1 Piston and Crankshaft Mechanism

therefore must have the same form in both objects. Examples of lower pairs include the *revolute* or turning pair, the *prismatic* or sliding pair, and the screw. Only a few lower pairs are geometrically possible. Higher pairs are kinematic pairs in which contact between the elements takes place along lines or points of contact rather than over a full surface. The most prevalent higher pairs are gears.

Though the possible variety of higher pairs is much greater than that of lower pairs, in actual machinery lower pairs are more common. The reason for this phenomenon is physical rather than geometrical. Consider Figure 1. If the piston were a triangular prism rather than a cylinder, the mechanism would behave is exactly the same manner, but an implementation of it in real metal would wear out much sooner since all contact would be along three lines instead of an entire surface. And in general it is clear that given a choice between a higher pair and a lower pair to do a specific job, one would always prefer the lower pair because it would have a much longer working life. However, such a choice is not always possible. For example, no lower pair will do the job a gear does. So we might consider lower pairs to be the glue which holds mechanisms together, with an occasional higher pair inserted to do a job that a lower pair cannot perform.

Many CAD/CAM solid modeling systems use a representation known as Constructive Solid Geometry [Requicha 1980], which is the input representation that my

(This research was supported by a grant from ITT)

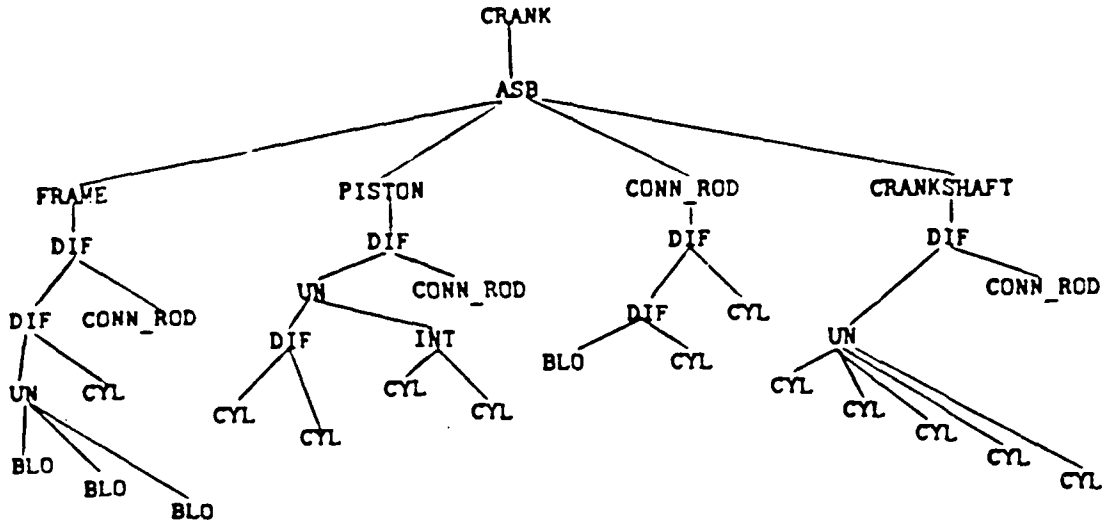


Figure 2 CSG Tree for Crank Mechanism

program expects.* In the CSG representation, each part in a machine is represented as a closed subset of three-dimensional Euclidean space which is formed by applying the Boolean set operations of union, intersection, and difference to a small set of primitive subsets of three-dimensional space such as the block, cylinder, sphere, cone, and torus. For example, a square plate with a hole in it might be represented as the difference of block and a cylinder, where the block would be appropriately sized and positioned to represent the plate, and the cylinder would have the correct diameter and position so that the difference operation would create the desired hole in the plate. The actual representation is a binary tree whose internal nodes are set operations and rigid motions and whose leaves are primitive solids.

CSG is an appealing choice of representation for several reasons. A machine design can be specified in a natural and straightforward way using a CSG representation, and most parts of typical machines can be specified in a relatively simple and compact way using CSG. See Figure 2 for an example of a CSG tree for the mechanism in Figure 1. CSG is also a good representation to use when symmetry is an important issue, as it is in the machine domain, because the possible symmetries of a part with a CSG representation may be computed easily. The five CSG primitives mentioned earlier have clearly defined symmetries, and the symmetries of combined primitives may be calculated simply. For example, two solids with prismatic symmetry along parallel axes will combine to form an object with the same prismatic symmetry. The square plate with a hole in it described earlier is an example of such a combination. Both the block and the cylinder have prismatic symmetry along parallel axes and the combination does also.

The top level algorithm employed by my kinematic analysis program is the following:

- Identify lower pairs by the symmetries of the common surfaces shared by the elements of the pair
- Identify higher pairs (particularly gears and cams) by noticing appropriately intersecting motion envelopes
- Find constraints on the relative and absolute positions and orientations of kinematic pairs
- Detect relationships between the motion of one pair and that of another due to relative geometric configurations of the pairs
- Compose these relationships to form new relationships

The first step for kinematic analysis is to identify the lower pairs, which we do by examining the geometry of their shared surface area. If the machine is represented using CSG, a lower pair must meet four requirements:

- Each of the two parts of the machine which are the elements of the pair must have a subpart (typically a CSG primitive) which shares a common symmetry with a subpart of the other element. Since the symmetry of a mechanical part represented in CSG comes from the symmetries of the primitives in the CSG tree for the part, and a typical part is made from a small number of primitives, it is straightforward to find all the possible symmetries in a particular part, and to find other parts in the mechanism with common symmetries.
- The primitive or set of primitives with the symmetry must be solid in one of the two parts and hollow (as a result of a difference operation) in the other. For example, in Figure 1 the CSG tree for the piston contains a solid cylinder which fits inside a hollow cylinder in the CSG tree for the frame and thus a cylindrical pair is formed.
- The primitive or set of primitives in the two parts must be identical except for possible differences in length along the axis of symmetry, and must have

* The program makes use of the PADL-2 solid modeling system developed by the Production Automation Project at the University of Rochester.

the same spatial position except for a possible offset along the axis of symmetry. For example, in Figure 1 the piston and the cylindrical hole in the frame into which it fits both have the same cross section but differ in length, and their positions correspond except for a relative offset along the axis of symmetry.

- The hollow space which forms one half of the pair must be surrounded by solid material. For example, in Figure 1 the piston and frame would not form a lower pair if the cylindrical solid which was subtracted from the frame was not subtracted in an area where there were other solid primitives to make a hole in.

The second step in the kinematic analysis is the identification of higher pairs, normally either gears or cams. The teeth of a gear typically have a profile curve that is either a cycloid or an involute, which can't be formed directly from the simpler CSG primitives listed earlier, but must have an additional geometric modifier specifying the shape of one of its surfaces. A gear pair can be identified by noticing two parts which have "geared" surfaces in contact and are both connected by revolute joints to a third part. Alternatively, if the gears don't happen to be precisely in contact in the original CSG tree, their motion envelopes (which will be cylindrical) may be formed and checked for intersection. Note that a gear will always be a member of a (lower) revolute pair as well as of a (higher) gear pair.

Cam pairs are identified by a two step procedure. First, parts which are elements of revolute pairs are examined to find subparts without the appropriate rotational symmetry, i.e. "suspicious bulges". The motion envelopes of these potential cams are then formed and checked to see if they intersect any part which is not known to be kinematically paired with the part containing the cam. (For example, the crankshaft in Figure 1 has suspicious bulges whose motion envelopes intersect the connecting rod, but the two parts form a revolute pair so they could not also have a cam relationship.) Typically a cam's motion envelope will intersect a part which is an element of a prismatic joint, which is then pushed when the cam is in the right position. The cam relationship may then be determined from the relative position of the cam and the prismatic pair it pushes.

For the purpose of reasoning about interactions between kinematic pairs, we represent a kinematic pair as an *axis* and one or more *parameter values*. For a revolute, screw, or cylindrical pair, the axis is just the axis of revolution while for a prismatic pair the axis is any line parallel to the direction of prismatic motion. For revolute and screw pairs, we will make use of the convention that the parameter value for the pair will be the number of degrees that the solid element of the pair has moved, relative to the hollow element of the pair, from its position in the initial CSG representation of the machine that was input to the kinematic analysis program. For prismatic joints, the parameter value will be the number of distance units the solid element has moved relative to the hollow element.

The axis of a cylindrical pair like the piston and frame in Figure 1 is also well defined although the cylindrical pair is different from the three pairs discussed above in that it has two degrees of freedom. (Note that in Figure 1 the piston's rotational freedom is eliminated by external constraints, effectively reducing it to a prismatic pair. So the position of a cylindrical pair must by represented by two parameters instead of one.

The abstraction of a higher pair is frequently less straightforward than that of a lower pair, but it is typically less important. Consider a kinematic pair consisting of two gears. There will always be a third part in the machine with which both gears form revolute pairs. The existence of the gear pair will impose a constraint on the parameters values of the two revolute pairs, which is that one value must be a constant multiple of the other. It would be possible to define a convention to associate some parameter with the gear pair, but that would be superfluous because that information is already included in the values of the parameters for the two revolute pairs. Cams similarly impose constraints on the parameter values for lower pairs.

The third step in the high level kinematic analysis algorithm is to find constraints on the relative and absolute positions and orientations of kinematic pairs, such as:

- Constraints imposed by the properties of rigid bodies. For example, if one part is an element of two different kinematic pairs (the usual case), and in the original CSG tree the axes of those two pairs have some interesting geometric relation like being parallel or perpendicular or a certain distance apart, then the program may conclude that this relation will hold regardless of what motions the mechanism makes, because the part is a rigid body.
- A typical machine will have an element called the "frame" which will never move and will serve to provide a global coordinate system. A simple but important constraint on any axis of a pair which includes the frame as an element is that the axis will always have the same position.
- The existence of certain constraints may imply the existence of other constraints. For example, in Figure 1 the crankshaft/frame axis is parallel to the crankshaft/connecting rod axis and they will always stay parallel, and the crankshaft/connecting rod axis is parallel to the piston/connecting rod axis and they will always stay parallel, so we may conclude that the crankshaft/frame axis will always stay parallel to the piston/connecting rod axis.

When a kinematic analysis program analyzes the mechanism in Figure 1, we would like it to "understand" that when the crankshaft turns the piston will move back and forth. In concrete terms, this requirement means that the program should discover what the relationship is between the parameter describing the position of the crankshaft/frame revolute pair and the parameter describing the position of the piston/frame prismatic pair. This relationship is clearly not a one-to-one function since

any one position of the piston can correspond to two different positions of the crankshaft. A natural way to express this relationship is as a set of "monotonic segments" — subsets of the relationship in which both parameters vary monotonically and where a precise function relating them can therefore be given. The output of my kinematic analysis program is a set of relationships between parameterizations of kinematic pairs, expressed in terms of monotonic segments.

The fourth step of the kinematic analysis process is to find these relationships. Most of the elementary mechanisms found in typical machines are either gears, cams, or linkages [Suh and Radcliffe 1978]. They are formed from the kinematic pairs whose identification we have discussed above, and they impose the relationships we wish to find.

- The presence of a gear pair imposes a very simple linear algebraic relationship on the revolute pairs which allow the gears to turn. The parameter value (as described above) for one of these revolute pairs will always be a constant multiple of the parameter value for the other revolute pair, and this constant multiple may be computed by comparing the diameters of the two gears.
- A cam pair imposes a relationship between the parameter value for the revolute joint of the camshaft (i.e. the angle through which it has turned) and the parameter value of the prismatic pair which the cam pushes. An approximate version of this relationship which suffices for many purposes is a mapping taking one range of camshaft angles into the maximum displacement for the prismatic pair and the complementary range of angles into the minimum displacement.
- Transformations of frame of reference also yield relationships between parameter values. For example, if one part is an element of two coaxial revolute pairs A and B then the motion of the other element of pair A as seen from the frame of reference of the other element of pair B is the (vector) sum of the parameters of the two pairs.
- The relationships imposed by linkages may often be found by simple reasoning about triangle geometry, since knowledge of trigonometry and triangle geometry may be applied not just to points but also to distances between sets of parallel axes. For example, in Figure 1, the three parallel axes discussed above may be projected onto a perpendicular plane, and the resulting points will form a triangle whose properties will impose additional constraints on the mechanism. This transformation is important in the analysis of linkages, because the majority are planar linkages whose revolute joints have mutually parallel axes.

The fifth and final step in doing the kinematic analysis is to compose the relationships found in order to form new relationships. The relationships between parameter values of kinematic pairs are expressed as one-to-one mathematical functions (which are split into monotonic segments if necessary). The composition of two one-to-one functions yields yet another one-to-one function. For example, if in Figure 1 a gear on the crankshaft were to drive a camshaft, then given the position of the camshaft we would know the position of the crankshaft, and given the position of the crankshaft we would know the position of the piston, so therefore by composing these two functions we would get a new function such that given the position of the camshaft we would know the position of the piston.

```
Kinematic Pairs
< LFRAME   LPISTON > (type = PRISMATIC)
                     axis = [-1 0 0] [-4.0 0] ratio = 0
 PERPENDICULAR to < LCONN   LPISTON >
 PERPENDICULAR to < LFRAME  LCRS >
 FIXED
< LFRAME   LCRS > (type = REVOLUTE)
                  axis = [32.0 0] [32 7.0] ratio = 0
 PARALLEL to < LCONN   LPISTON >
 PARALLEL to < LCONN   LCRS >
 DISTANCE to < LCONN   LCRS >  = 7
 PERPENDICULAR to < LFRAME   LPISTON >
 FIXED
< LCONN   LPISTON > (type = REVOLUTE)
                     axis = [-8 -0 5.0] [-8.0 5.0] ratio = 0
 PARALLEL to < LFRAME   LCRS >
 PARALLEL to < LCONN   LCRS >
 DISTANCE to < LCONN   LCRS >  = 33
 PERPENDICULAR to < LFRAME   LPISTON >
< LCONN   LCRS > (type = REVOLUTE)
                  axis = [26 -0.0] [26.-0.0] ratio = 0
 PARALLEL to < LCONN   LPISTON >
 DISTANCE to < LCONN   LPISTON >  = 33
 PARALLEL to < LFRAME   LCRS >
 DISTANCE to < LFRAME   LCRS >  = 7
Interrelationships
Monotonic Segment
 < LFRAME   LCRS  > goes from     0 to   180
 < LFRAME   LPISTON> goes from    0 to   -14
Monotonic Segment
 < LFRAME   LCRS  > goes from   180 to   360
 < LFRAME   LPISTON> goes from  -14 to     0
```

Figure 3 Kinematic Analysis of Crank Mechanism

Thus, following classic tradition, we have reduced the problem of reasoning about geometric relations to the simpler problem of reasoning about algebraic relations. We can expect this technique of kinematic analysis to extend to very complex machines with large numbers of parts, because we will use geometric reasoning to discover relationships between the motions of parts and their near neighbors, and since these relationships will be expressed as one-to-one mathematical functions we may compose them to an arbitrary extent, producing relationships between the motions of parts and far distant other parts.

The output from my program in analyzing the mechanism in Figure 1 appears in Figure 3. The first stage of the analysis is to find lower pairs from local geometric properties using the algorithm discussed earlier. The program finds four cylindrical pairs and then uses local geometry to show three of the four are actually revolute pairs, by showing that prismatic motion of the pairs is blocked at each end. No higher pairs are found in the second stage of analysis.

The third stage of the analysis looks for constraints on the interactions of the pairs. Since this mechanism is a linkage, most of the constraints have to do with relations between axes such as parallelness and perpendicularity. After using transitivity to conclude that the crankshaft /frame axis will always stay parallel to the piston/connecting rod axis, as discussed above, the program

uses this new constraint to conclude that the piston frame pair is actually a prismatic pair rather than a cylindrical pair.

The fourth stage of the analysis applies basic triangle geometry to the triangle formed by the always parallel crankshaft/frame, crankshaft/connecting rod, and piston/connecting rod axes. The previous stage of analysis showed that two sides of the triangle had unchanging lengths. Therefore, if the position of the piston is known it determines the length of the third side of the triangle and thus by the side-side-side theorem uniquely determines the triangle and therefore the position of the crankshaft. Similarly, if the position of the crankshaft is known it determines an angle of the triangle and by the side-side-angle theorem (where the angle is by the shorter side) determines the triangle.

Therefore a strict relationship between the piston and crankshaft parameters has been shown to hold. The triangle inequality shows that the two extreme points of the piston's motion will occur at distances which are the sum and difference of the lengths of the other two sides of the triangle. When the extreme points are reached the three parallel axes are coplanar and their projections are colinear, at which point the triangle is no longer a triangle since one of its angles is 0 or 180 degrees. Since there are two ways to place a triangle in a plane if the positions of two of its points are fixed, there are two ways for the mechanism in Figure 1 to make the transition between its two extreme points, so the relation between the piston and crankshaft has two monotonic segments.

The precise function relating the positions of the piston and crankshaft in the two monotonic segments is a fairly complicated combination of trigonometric functions and their inverses. The above analysis gives us enough information about the triangle to find the exact function, but in many cases this is unnecessary, and what we really want is just the definition of the monotonic segments of the relationship in terms of their extreme points and the fact the the relationship is monotonic within the segment. For example, if we want to know the average turning speed of the crankshaft given a certain driving rate of the piston the precise form of the function is irrelevant as long as we know that when the piston goes back and forth once the crankshaft makes exactly one complete turn. Thus in Figure 3 the internal details of the functions in the monotonic segments are not specified.

Note that the result of the kinematic analysis process is a mathematical relationship between the position of the crankshaft and the position of the piston in the form of a mapping. This mapping is independent of what particular *motion* the crankshaft is subjected to, that is, from the parameterization of the crankshaft's position with respect to time. For example, if the crankshaft was turning at a constant speed, its position would be a linear function of time, and we could compose that function with the mapping our kinematic analysis gives us to get an expression for the piston's position as a function of time which would show that the piston was oscillating back and forth at a constant frequency.

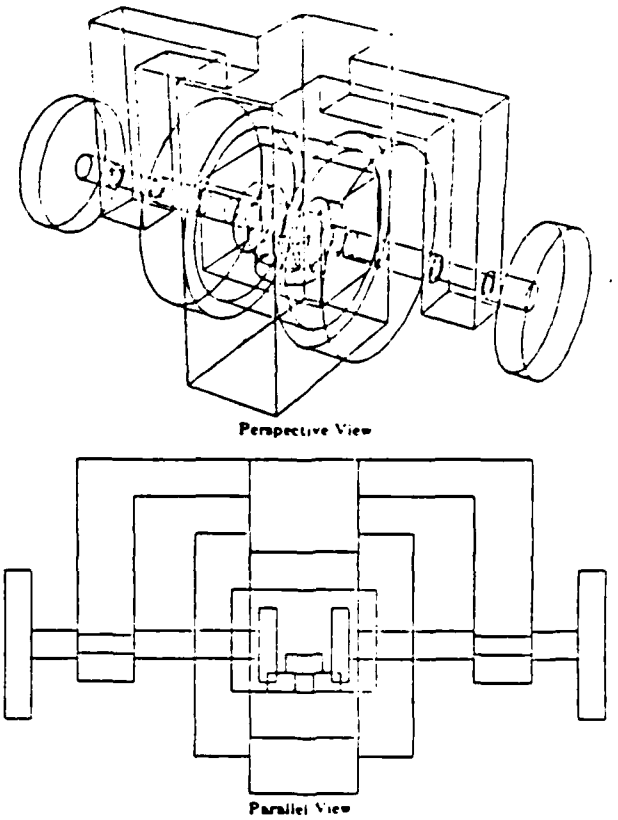


Figure 4 Differential Mechanism

Now let us consider the process which my program uses to create a kinematic analysis of the differential in Figure 4. Note that in Figure 4 the gears are displayed as cylinders, due to a limitation in our CAD/CAM solid modeling program. Also note that the differential has only one pinion connecting the two half axles, rather than four as a real differential would. This simplification is irrelevant to the behavior of the program since the four pinions behave in an identical fashion, and the program would produce exactly the same analysis with four pinions as with one.

The output of the kinematic analyzer when working on the differential appears in Figure 5. The first two steps of the kinematic analysis process proceed by the standard methods described earlier and identify both the lower pairs and the higher pairs (which are all gears in this case). The constraints found in the third stage of processing turn out to be irrelevant to the final analysis here, and the important work is done in the fourth and fifth steps. Note that the program determines that the pairs formed by the differential box and the two half axles are actually revolute pairs rather than cylindrical pairs since both the differential box and the two half axles form revolute pairs with the frame.

In the fourth stage of the analysis, the local relationships are found. Each gear pair yields a simple linear relationship: the parameter of one revolute joint is always a constant multiple of the parameter of another rev-

```
Kinematic Pairs
< LFRAME   GEARR > (type = REVOLUTE)
                   axis = [18 76.0.0],[19 76.0.0] ratio = 0
< LFRAME   GEARL > (type = REVOLUTE)
                   axis = [18 76.0.0],[19 76.0.0] ratio = 0
< LFRAME   DBOI > (type = REVOLUTE)
                   axis = [18 76.0.0],[19 76.0.0] ratio = 0
< GEARM   GEARR > (type = GEAR)
                   axis = [0.0.0],[0.0.0] ratio = 1
< GEARM   GEARL > (type = GEAR)
                   axis = [0.0.0],[0.0.0] ratio = 1
< GEARM   DBOI > (type = REVOLUTE)
                   axis = [0.-1.0],[0.-2.0] ratio = 0
< DBOI   GEARR > (type = REVOLUTE)
                   axis = [-12.0.0],[-11.0.0] ratio = 0
< DBOI   GEARL > (type = REVOLUTE)
                   axis = [-12.0.0],[-11.0.0] ratio = 0
Interrelationships
Monotonic Segment
 < DBOI   : GEARL > = -1*< DBOI : GEARR>
Monotonic Segment
 < LFRAME   GEARR > =  1*< GEARM   DBOI > +  1*< LFRAME   DBOI>
Monotonic Segment
 < LFRAME   GEARL > = -1*< GEARM   DBOI > +  1*< LFRAME   DBOI>
Monotonic Segment
 < LFRAME   GEARL > =  1*< DBOI : GEARL> +  1*< LFRAME   DBOI>
Monotonic Segment
 < LFRAME   GEARR > =  1*< DBOI : GEARR> +  1*< LFRAME   DBOI>
Monotonic Segment
 < GEARM   : DBOI  > = -1*< DBOI   GEARL>
Monotonic Segment
 < GEARM   DBOI  > =  1*< DBOI : GEARR>
```

Figure 5 Kinematic Analysis of the Differential (edited)

olute joint, with the constant factor being the gear ratio Frame of reference transformations also yield linear relationships: for example, the angular displacement of the right gear relative to the frame is simply the sum of its displacement relative to the gear box plus the displacement of the gear box relative to the frame.

The relationships we are really interested in are generated in the fifth and final stage of the kinematic analysis. Since the relationships found in the fourth stage are of the simplest possible kind — just linear mappings — composing them into new relationships (which are also linear mappings) is very straightforward. In Figure 5, four of the interrelationships listed are found in the fourth stage of processing from local geometry, and the other three are generated in the fifth stage by composing relationships found in the fourth stage. The differential mechanism has two degrees of freedom, unlike most common mechanisms which have only one degree of freedom [Hunt 1959], but since our analysis works by simply composing mathematical functions it works perfectly well in this case.

In conclusion, let us examine one potential use for this kinematic analysis. Consider a hypothetical program which designs new machines and has a kinematic analyzer as one of its modules. Suppose the design program was designing a four-wheeled land vehicle like an automobile. If the design program initially chose front-wheel drive and a rear axle which was a single rigid assembly then by running simulations it might notice a bug in this design which is that when the vehicle turned, one or both of the rear wheels would slip since they would have different turning radii and therefore could not travel at the same speed during a turn without slipping.

If the design program was able to guess that a differential might help it solve this problem, it would have to make use of the kinematic analysis of the differential to determine whether it could actually do the job. Part of the output of the kinematic analysis is that the total angular distance through which one half-axle turns is equal to the angular distance through which the differential box has turned *plus* the distance through which the middle pinion gear has turned, and the total angular distance through which the *other* half-axle turns is equal to the angular distance through which the differential box has turned *minus* the distance through which the middle pinion gear has turned. By dividing all these angular distances uniformly by a time parameter we see that exactly the same relationships hold for the angular *speeds*. At the kinematic analysis level the differential mechanism has two degrees of freedom so its motion is determined both by the speed at which the differential box is turned by the drive shaft and by the speed at which the middle pinion gear turns.

The hypothetical designer program knows that to avoid slipping the ratio of the speeds of the two half axles must be the reciprocal of the ratio of their turning radii, and that this ratio will be enforced by the friction of the wheels with the road unless overcome by brute force as in the case of a single rigid axle. This additional constraint combines with the output of the kinematic analysis mentioned above to restrict the entire system to a single degree of freedom — the speed of the drive shaft. The three (linear) equations of constraint may be combined using simple algebra to show that the speed of the outer half axle will equal the speed of the differential box *plus* a fully determined speed, and the speed of the inner half axle will equal the speed of the differential box *minus* the same speed, and therefore that the differential mechanism satisfies the requirement that the vehicle be able to turn without having any of its wheels slip.

Bibliography

Johan de Kleer and J. S. Brown 1984 A qualitative physics based on confluences, *Artificial Intelligence* **24**, 7–23

Kenneth Forbus 1984 Qualitative process theory, *Artificial Intelligence* **24**, 85–168

Kenneth H. Hunt 1959 *Mechanisms and Motion*. John Wiley & Sons, Inc., New York

Benjamin Kuipers 1984 Commonsense reasoning about causality: deriving behavior from structure, *Artificial Intelligence* **24**, 169–203

Aristides A. G. Requicha 1980 Representations for Rigid Solids: Theory, Methods, and Systems, *ACM Computing Surveys* **12**, 437–464

Franz Reuleaux 1876 *The Kinematics of Machinery*. Macmillan and Co., London

Chung Ha Suh and Charles W. Radcliffe 1978 *Kinematics and Mechanisms Design*. John Wiley & Sons, New York, etc.

Craig Stanfill 1983 *Form and Function: The Representation of Machines*. TR-1347, Computer Science Dept., University of Maryland, November, 1983

[27]

Categorizing Process Abstraction in Simulation Modeling

Paul A. Fishwick
Department of Computer and Information Science
University of Florida
Bldg. CSE
Gainesville, FL 32611
(CSNET: *fishwick@ufl.edu*)

Abstract

Large and complex simulation models often require some method for abstraction or simplification; models are better comprehended and manipulated when they are defined over multiple abstraction levels. Within the general bounds of simulation modeling, there has been a considerable amount of work in many different disciplines concerning methods for representing abstract models. Figure 1 depicts some of these areas. Many of these disciplines, however, do not address issues concerned with *multiple* abstraction levels and the *mapping problem* among levels.

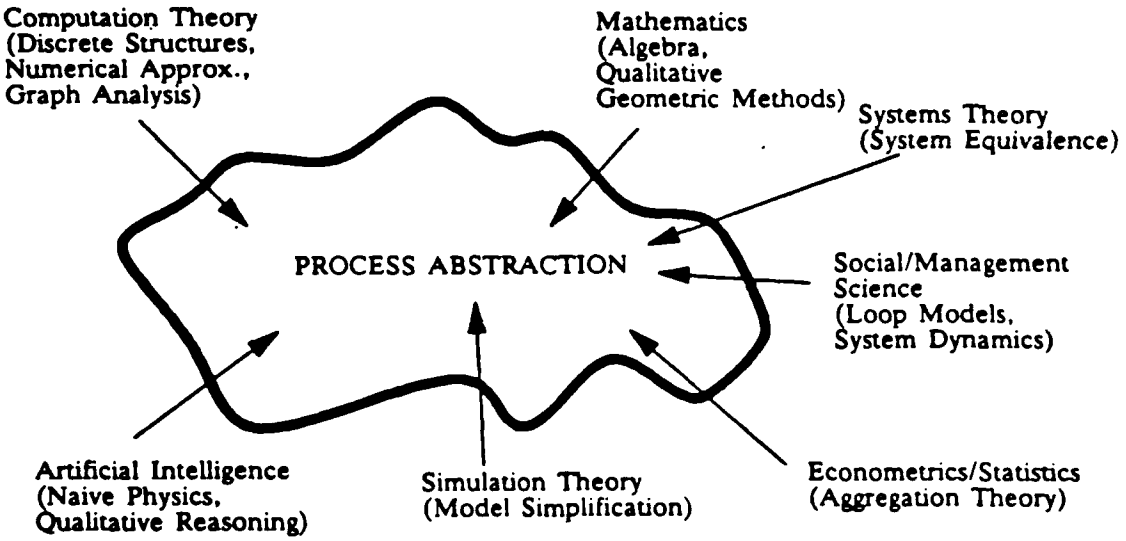


Figure 1 – Applications using Process Abstraction

There has been substantial work dealing with formalisms for simplification [Zeigler 76] and abstraction in scene animation [Fishwick 86] but much further research needs to be done to study tradeoffs between complexity and sufficiency in modeling. We propose a

theory of process abstraction [Fishwick 87] which attempts to categorize fundamentally different types of abstraction. We will present formalisms for valid methods of abstraction (see figure 2) and show their application using an example process. The process is defined as a partially ordered abstraction network whose nodes represent abstraction levels and whose arcs represent valid abstractions.

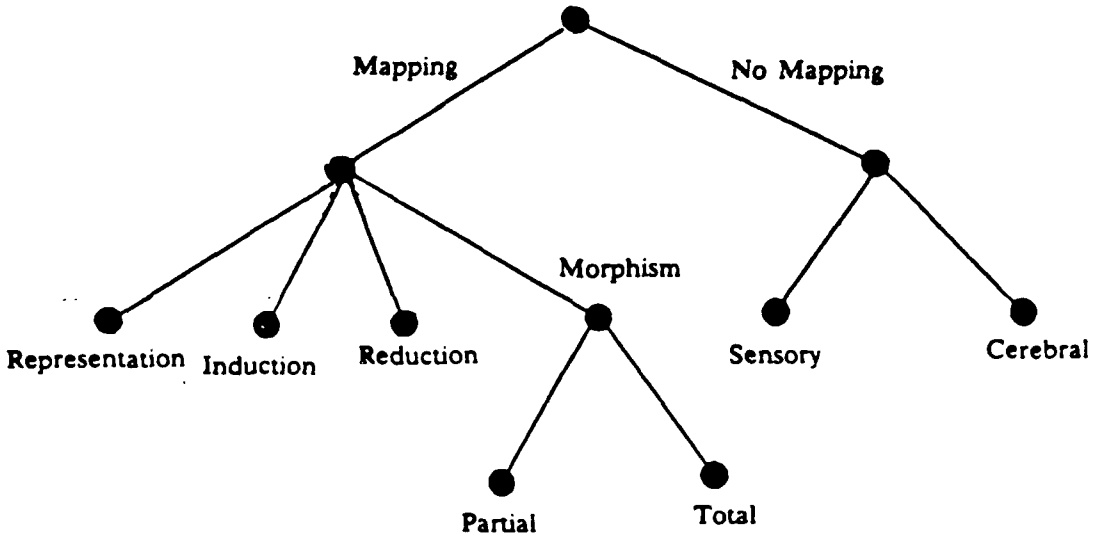


Figure 2 – A Taxonomy for Process Abstraction

The primary motivation for this work is to create a formal foundation as a step to better understanding the nature of process modeling – in many instances, the simulationist begins with "intuitive models" such as those defined using sensory and cerebral abstraction as shown in figure 2. It is then natural to progress to more formal methods once there is a greater understanding of the process. A tenet of process abstraction is that we should endeavor to maintain each abstraction level (from intuitive to formal) in an abstraction network rather than to immediately dispose of older, less accurate models once a deeper model has been found. We will still continue to validate models and dispose of incorrect models but not at the expense of correct, qualitative models.

Workshop Issues

The research in artificial intelligence (including naive physics and qualitative reasoning) provides useful concepts when looking for the more intuitive modeling methods. For instance, confluences [de Kleer 84] can be used as a language for qualitative modeling; however, some important questions should be posed. What are the constraints on defining valid "mental models?" Many different modeling methods have been proposed – are there any benefits to using one method over another? What are the relationships and differences between the AI area of qualitative physics and other areas such as system dynamics and qualitative modeling [Puccia 85]? Also, can formal theories developed in systems and simulation theory be used to place qualitative modeling (i.e. mental model-

ing) on firmer ground? The AI & Simulation Workshop will provide the proper forum for discussing some of these key issues.

References

Zeigler, Bernard P., *Theory of Modeling and Simulation*, John Wiley & Sons, 1976.

Fishwick, Paul A., *Simulating Complex Processes over Multiple Levels of Abstraction*, Ph.D. Dissertation, University of Pennsylvania, 1986.

Fishwick, Paul A., *The Role of Process Abstraction in Simulation*, submitted to IEEE Transactions on Systems, Man & Cybernetics, 1987.

DeKleer, Johan, and Brown, John Seely *A Qualitative Physics Based on Confluences*, Artificial Intelligence, 24 (1–3), 1984.

Puccia, Charles J. and Levins, Richard, *Qualitative Modeling of Complex Systems*, Harvard University Press, 1985.

[31]

Event Horizons in Artificial Intelligence Systems

Tad Hogg, Bernardo A. Huberman and Jeff Shrager

Intelligent Systems Laboratory
Xerox Palo Alto Research Center
Palo Alto, CA. 94304

Abstract

Cognitive models and large scale artificial intelligence systems undergo sudden phase transitions from disjointed parts into coherent structures as their topological connectivity increases beyond a critical value. This provides a dramatic instance of global behavior which is not readily apparent in small-scale simulations. These situations, ranging from production systems to semantic net computations, are characterized by event horizons in space-time that determine the range of causal connections between processes. At transition, these event horizons undergo explosive changes in size. This provides a general methodology for analyzing associationist models of memory and the behavior of large scale computation. The theory is experimentally tested in spreading activation networks.

[32]

REASONING ABOUT DIAGNOSIS AND TREATMENT IN A CAUSAL MEDICAL MODEL USING SEMI-QUANTITATIVE SIMULATION AND INFERENCE

Lawrence E. Widman, M.D., Ph.D.
Division of Cardiology, Department of Medicine
University of Texas Health Science Center
San Antonio, Texas 78284

This project seeks to develop improved knowledge representation and simulation-based reasoning algorithms for domains dealing with time-varying phenomena.

Improvements would allow expert programs to (1) perform model-based diagnosis using observable signs and symptoms, (2) recommend treatments based on a global understanding of hypothesized disorder(s), (3) evaluate the outcome of treatment to refine the initial diagnosis and detect new concurrent faults, and (4) communicate symbolically with the user when explaining its reasoning and acquiring new information.

The fundamental approach consists of (1) hypothesizing initial conditions in the "present" time by constrained propagation of available information, then (2) reasoning forward in time by simulation.

Simulation using a symbolic model can be done by defining functional building blocks for describing the expert domain in symbolic terms, translating the domain into a set of first order ordinary differential equations on the basis of the building block definitions, establishing initial conditions for the equations by constraint propagation (see below), and integrating the equations numerically by standard techniques [1]. This approach combines the descriptive power of mathematics with the symbolic reasoning power of expert system technology.

The approach has been tested in the medical domain. A symbolic model of the human cardiovascular system was built from well-established causal relationships. Very few of the relationships were specified quantitatively: most were specified qualitatively by direction of influence (positive or negative) only. Time delays were specified by order of magnitude. Slopes of relationships between variables were specified when necessary to resolve ambiguity of conflicting interactions; all data were taken from standard domain textbooks.

The medical model contained 46 variables, of which only 14 did not participate in feedback relationships with other variables in the model.

The approach was tested in two ways: inference of possible faults from limited observations of a test case with an unknown fault, and forward simulation of a known fault.

Inference of possible faults was tested by presenting the value of a single model variable to a constraint propagation algorithm and requiring it to generate sets of self-consistent model states which included the given variable value [2]. This algorithm used a search strategy in which the relationships between variables specified in the model were interpreted as constraint equations.

Constraint propagation was performed by domain independent rules which recognized morphologic patterns in the model. The algorithm identified single faults properly. Interestingly, its semi-quantitative value assignments, when mapped onto a numerical space, achieved correlation coefficients up to 0.9 with the values produced from the same faults by forward simulation (Fig. 1).

Forward simulation was tested in sixteen classic cardiovascular disorders, such as decreased contractility, increased pulse rate, decreased systemic vascular resistance, and decreased gravity [1]. In all cases, the output of the simulation was qualitatively correct: all variables changed in the appropriate direction on the proper time scale with semi-qualitatively correct magnitudes of change.

Symbolic model-based simulation using symbolic functional building blocks for describing the expert domain may be useful in a variety of physically realizable domains. The symbolic form of the model also lends itself to simulation at multiple levels of abstraction and automatic analysis of simulation results.

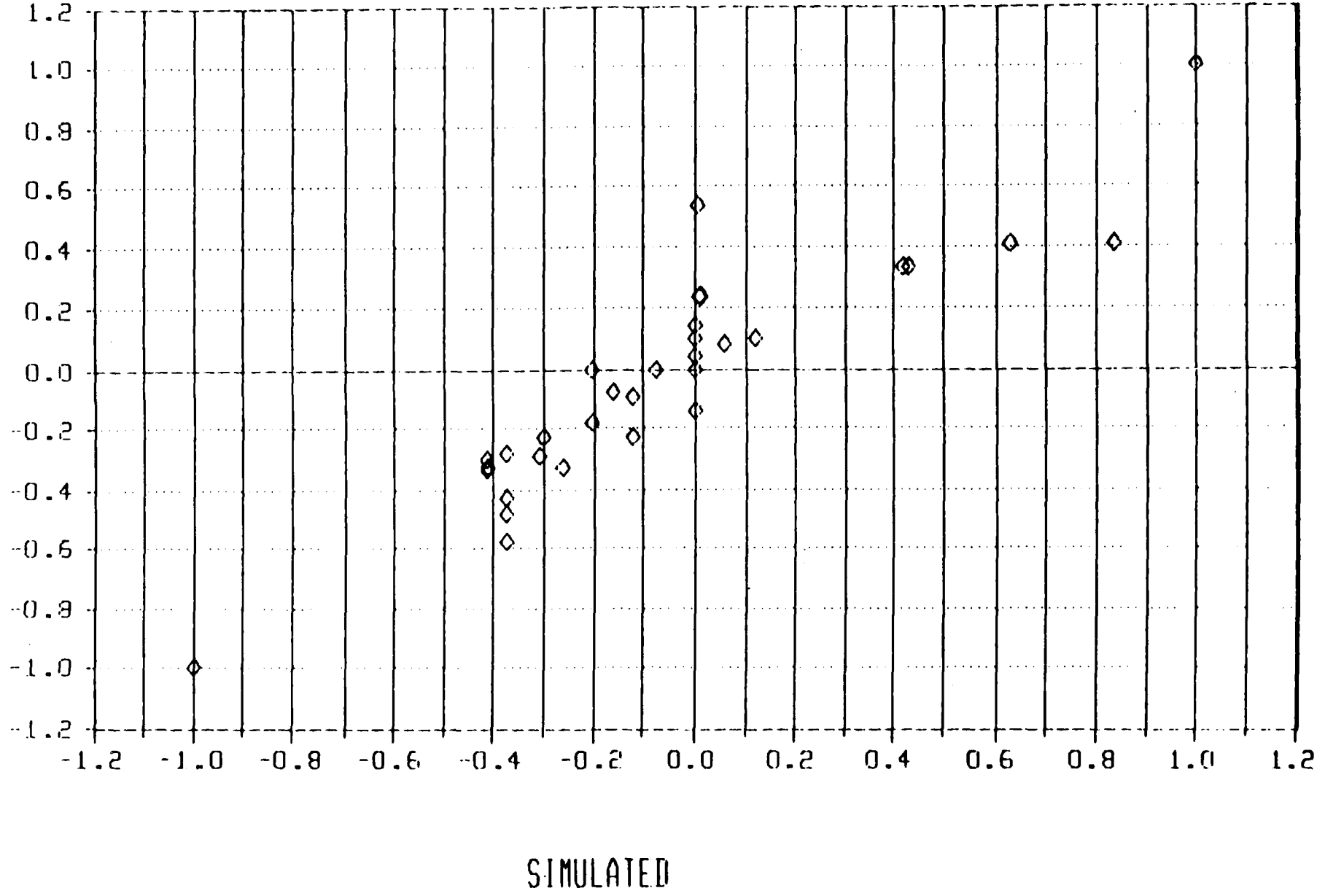
Figure 1: Scatter plot of steady-state simulated values versus values inferred by constraint propagation, for decreased Contractility.

References:

[1] Widman, L.E. Representation Method for Dynamic Causal Knowledge Using Semi-Quantitative Simulation. Fifth World Conference on Medical Informatics. 1986: 180-184.

[2] Widman, L.E., Lee, Y.-B., and Y.-H. Pao. Diagnosis of Medical Causal Models by Semi-Quantitative Reasoning. (submitted as an invited paper to the 1987 Symposium on Computer Applications in Medical Care, Washington, D.C.)

FIG 1: SIMULATED VS INFERRED MODEL VALUES



ISIM: Towards an integration of artificial intelligence and simulation [35]

Roy Masrani, Sheila McIlraith
Advanced Technologies Department
Alberta Research Council
Calgary, CANADA

This paper outlines work in progress at the Alberta Research Council aimed at integrating reasoning and representation techniques developed in Artificial Intelligence with discrete-event simulation. The paper describes ISIM (Intelligent Simulation): a discrete-event simulation environment built on KnowledgeCraft (KC). ISIM investigates two avenues for integrating AI and simulation:

- a goal directed expert system that determines the appropriate simulation parameters for achieving prestated objectives
- an exploratory system that experiments with the simulation model to find interesting relationships between different components in the model

The first step in achieving either of these goals is to enhance KC to make it more suitable for simulation problems. We take concepts from a well-known simulation environment called DEMOS (Birtwistle, 1979) and implement them in KC. The resulting system imposes a DEMOS-like framwork on the simulation model, forcing its description in terms of entities, resources, queues and events. A monitoring system automates the data collection and statistical analysis tasks to provide the expert system with a behavioral description of the model. The expert system can, in turn, react to the statistics by manipulating the model in some way. Figure 1 illustrates the interrelationships between the three modules in ISIM: the model representation, the monitoring system and the expert system. This paper continues with a discussion of these components after an overview of the salient features of KC. KC is a Lisp-based schema (aka frame) representation language combined with Ops5 and Prolog for forward and backward chaining, agendas, contexts, and various window and user-interface tools (Pepper & Kahn, 1986). The paper will refer to the simple simulation scenario described in Box 1.

> A coal transportation system comprises a number of trains that move coal from mines along the railway to a port. Ships then transport the coal to an island-the final destination. Mines produce coal and deposit it on stockpiles for the trains to pick-up. Similarly, the trains deposit coal on stockpiles at the port.

Box 1. Sample simulation scenario

KnowledgeCraft

A standard schema represenation language is augmented by demons, user-defined relations and object-oriented programming facilities. **Demons** are lisp functions that can potentially mediate interactions with any frame in the system. Depending on how it is defined, a demon can "fire" before or after a value in a slot is affected (either accessed or modified). The demon can change the value or have a "side-effect" like updating some statistic collected for the slot. **Relations** in KC (such as *is-a*, *instance*) can be programmed to suite the needs of the application. Some aspects of relations open to customization include the specification of which slots to inherit (or not) from the the parent schemas, the path to take for finding slots and actions to perform when the relation is created. Finally, **object-oriented programming** is implemented by treating values in slots as methods to be invoked when called to do so. For example, the schema *polygon* can have a slot *draw* and the value *draw-polygon-fn*. The function

(call-method 'polygon 'draw)

will execute the method specified in the *draw* slot for *polygon*.

Integrated with this representation scheme are various programming utilities typically used in AI applications. Forward and

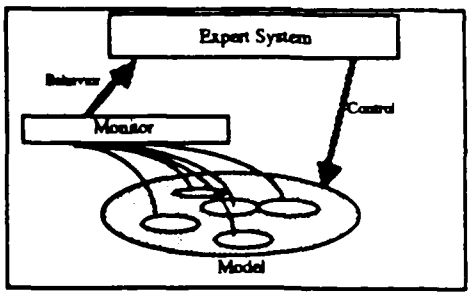


Figure 1. ISIM components

backward chaining deduction is implemented by Ops5 and Prolog respectively. Four types of agendas are available: simulation queue, scheduled queue, manual queue and imperative queue. The crucial difference between these queues lies in how events on them are executed. For instance, events on the imperative queue take precedence over all queues. Events on a scheduled queue are executed in real (absolute) time. Simulation queues are ordered according to the time on each event. Executing events on this queues advances the global simulation time. Manual queues, finally, do not have any automatic agenda mechanisms built in. All events on these queues must be dealt with manually!

A simple and elegant facility is available for implementing some form of "hypothetical" reasoning. Essentially, all frames in the system are associated with a context (a default is provided if none is specified). By creating a new "child" context, the system saves the current context and makes modifications only to the child context. This provides a form of non-determinism where a line of reasoning can be tested and adopted if the results are promising, or abandoned if not. Several contexts can be saved during the course of execution to compare the results of various hypotheses.

Model representation

Although the representation language in KC is powerful and expressive, it lacks the appropriate mechanisms for building simulation models. To rectify this, we develop a discrete-event simulation adopted from DEMOS in KC. DEMOS is an ideal candidate because it is an established simulation environment built on Simula (Birtwistle *et al*, 1973) - an object-oriented programming language with features similar to frames and semantic nets in AI.

ISIM is restricted in scope to problems that can be modelled as discrete-event simulations where discrete objects (active and passive entities, resources, queues) have behaviors describable in terms scheduled events. All appropriate statistics are collected by the system and represented in frames.

Entities

An entity is an identifiable object in the simulated world. All entities have attributes and behaviors (represented in frames) that can be invoked by sending the entity the appropriate message. ISIM distinguishes active entities from passive ones. Active entities are those objects in the simulation that do most of the processing and whose behavior is of interest. In a bank, tellers could be modelled as active entities. Passive entities on the other hand flow through the system as the objects processed by the active entities. Customers in the bank can be viewed as passive entities. In the coal transportation system the coal is modelled as passive whereas mines, ports and trains as active.

Generic methods associated with active entities are inherited by any object described as an active entity. Specialized objects can define methods that either override their generic counterparts or augment them. Thus, a description of a mine may augment the generic "start-operations" method by adding an event on the event list to produce coal.

Resources

In addition to entities, ISIM defines resources as objects that are used by other objects and are available on a competitive basis. Objects requiring resources may be put on wait-queues until the resource becomes available. An example of a resource in the coal transportation scenario is the loader/unloader required at mines and ports for the trains. All resources have methods for dealing with requests like "acquire", "release", "put-on-wait-queue" and so on. Resources also have wait-queues and can break down.

Queues and distributions

There are two types of queues defined in ISIM to augment the event-lists already available in KC: first-in-first-out and last-in-first-out. Methods associated with queues include "add-to-queue" and "remove-from-queue". Queues can have a maximum length and current length specified as local variables.

ISIM implements various statistical distributions with appropriate methods: get-value, initialize, reset. Examples of distributions include: poisson, negative-exponential, uniform, constant.

Monitor System

The monitor system in ISIM automatically collects statistics at three levels: system, group and member. System stats describe the behavior of the system as a whole. This may include the mean throughput time for passive entities, the total number of passive entities processed by the system, mean queue length, overall utilization factor for the system and so on. Group stats describe groups of entities. For instance, a set of statistics can be collected for all mines in the system, another set for all trains, one for all coal units to name a few. Statistics interesting at this level include mean idle, active and down time, utilization, total number of passive entities served, average queue length, number of zero-length queues and so on. Statistics at the member level

include the member counterparts of the group statistics: total idle, active and down time, total queue length, utilization factors.

All statistics are stored in separate frames accessible by the expert system. When frames are made instances of simulation objects (like queues, entities, resources), the appropriate statistics modules are automatically attached to the correct slots. The behavior specification for entities thus need not include any extra overhead for managing statistics.

Expert System

This module is the least developed of the three. There are three programming tools available for use here: Ops5, Prolog, and contexts. We speculate on the use of these mechanisms for achieving the two goals outlined above.

Expert system for running the simulation

When considering ways of integrating AI and simulation technologies, using an expert system that runs the simulation is the obvious one. When a real-world problem cannot be solved analytically, it may be amenable to a simulation approach. Human experts run the simulation (which has presumably been validated against actual system performance), manipulating one or more of its parameters, to create the desired behavior. Recommendations are then made about the real world problem based on the simulation parameters. For instance, in our case study, the simulation of the coal transportation system may be run over 1000 hours of simulated time to find bottlenecks in the system or underutilized resources. The knowledge used by these experts can, in principal, be encoded in an expert system. Such a system would incorporate knowledge about simulation experiments in general with domain specific rules to control and interpret the simulation model.

We are in the process of eliciting heuristic rules from simulation experts to test this general notion. So far, a simple expert system module has been implemented in ISIM to maintain coal stockpiles at a constant threshold by using a simple set of rules to speed-up or slow-down mine production depending on current conditions. The rules test the current stockpile levels and train destinations. If the stockpiles are below threshold but there are no trains headed for the mine, production is not affected. If trains are expected to arrive soon, the expert system increases production to raise the threshold.

The forward chaining mechanism in Ops5 is well suited for interpreting the behavior of the simulation. Prolog's backward chaining mechanism can be used for performing experiments aimed at producing the desired results. One can imagine a prolog program running the simulation to find the operational parameters that can maximize utilization of the trains while, at the same time, minimizing cost increases.

An exploratory expert system for finding relationships between simulation components

The expert system here is also goal directed with the crucial difference that the goal is to find relationships between different parts of the system using sensitivity analysis techniques on the variables in the model. Given the set of parameters that the expert system can manipulate (speed/capacity of trains, production schedules of mines, stockpile thresholds), the expert system will experiment with these parameters and find statistically significant effects. At one end of the spectrum, the experimentation can be seen as random permutations of the possible parameters while at the other end the experimentation can proceed in an orderly manner, guided by general rules provided by an expert. Success in this project will be gauged on whether the expert system can find "interesting" rules in the transportation model like

fast, small capacity trains are more efficient than slow, large capacity trains
an intermittent operating schedule for mines closer to the port is better for maintaining a constant coal supply

Whereas the first expert system can be seen as a "black box", this approach is more like an advisory system capable of suggesting relationships to the human expert that may not be immediately apparent from examining the model.

Birtwistle, G. M., O-J. Dahl, B. Myhrhaug and K. Nygaard. (1973). *Simula Begin*.. Studentlitteratur, Lund, Sweden.

Birtwistle, G. M. (1979). *Discrete-event modelling on Simula*. MacMillan Press, LTD, London.

Pepper, J. & G. Kahn. (1986). "KnowledgeCraft: An environment for rapid prototyping of expert systems." Proc: Society of Mechanical Engineers Conference on Artificial intelligence for the automotive industry, Detroit, Michigan. March 12-13, 1986.

[36]

Model Generation from Specifications for Machining

Arie Ben-David*, Leon Sterling* and Joseph A. Kovach**

* Center for Automation and Intelligent Systems Research
and Department of Computer Engineering and Science
Case Institute of Technology
Case Western Reserve University
Cleveland, Ohio, 44106, USA.
Phone: (216) 368-5278. CSNET: leon@case

** Ex-Cell-O Corp. M/S 2456
Manager, Machining Technology
23555 Euclid Ave.
Cleveland, Ohio 44117
Phone: (216) 692-5278

Abstract.

We describe an implementation of an expert system capable of successfully performing the tasks of process design optimization and troubleshooting superalloys grinding operations from process specifications.

The system, named DT, partitions a design by automatically generating and solving several empirical models. Later it synthesizes the partial solutions by generating and solving a non linear optimization model. Several feedback paths are available and corrective actions are taken by the system if necessary.

DT incorporates various types of domain knowledge: Physical principles, state of the art empirical mathematical models, heuristics and textbook data. The process design optimizer and the troubleshooter are tightly coupled. Each modules cannot perform its task properly without activating the other. The troubleshooter includes knowledge which is beyond the scope of the empirical models and is capable of resolving any combination of multiple faults in any degree of severity. During its operation the system assigns values to fifteen process variables.

Synthesis of the design is done as follows: A unique model is generated for each process specification, taking into account the propagated partial solutions. In order to avoid, as much as possible, being stuck in a local optimum, the system generates an approximate solution for the optimization model. An initial guess is generated using heuristics and textbook data. Usually this guess violates some constraints and a description of the resulting problems is sent to the troubleshooter which resolves the conflicts with respect to all the constraints. Conflicts resolution is done using generate and test strategy and the 'best' point is selected as an initial search point for optimization. Both generation and evaluation of a new point are done by applying domain knowledge. The optimization model, some hints regarding adequate problem solving techniques and the approximate solution are sent to a commercially available optimization package. The final solution is checked using heuristics and corrective actions are taken should it not satisfy the criteria.

About 120 real world cases of surface and feed creep grinding have been tested already. The results of more than 95 percent of them were very good in the sense that experts would have chosen the same initial machine settings. The system is written in Prolog. The conflicts resolution requires about 20 iterations, and a typical process design takes about 120 seconds (excluding process specification input) on an 0.6M IBM AT. Using the same machine, models and optimization package, it takes the domain experts (who have developed the empirical models) about 3 hours per design to generate and solve 80 percent of the cases without using DT, mainly due to the time consuming tasks of manually generating the proper models and estimating an acceptable initial search point by trail and error.

The approach taken in DT seems to be adequate for solving similar problems in other machining domains like turning, ECM etc. Further enhancements of the system will be focused on giving it some ability to learn from its own failures by correcting the empirical models it builds.

References

[1] Davis,R. Reasoning from First Principle in Electronic Troubleshooting. International Journal of Man-Machine Studies, Sept. 1983, 403-423.

[2] Fink, P.K. Control and Integration of Diverse Knowledge in a Diagnostic Expert System. proc. IJCAI 1985, 426-431.

[3] Forbus K. D. Qualitative Process Theory. Qualitative Reasoning about Physical Systems. (edited by Bobrow D.G), MIT Press, 1985, 85-168.

[4] Herrod R.A. and Rickel J. Knowledge Based Simulation of a Glass Annealing Process. An AI Application in the Glass Industry. proc. AAAI 1986, 800-804.

[5] Koton, P.H. Empirical and Model-Based Reasoning in Expert Systems. proc. IJCAI 1985, 297-299.

[6] Kovach J.A. Thermally Induced Grinding Damage in Cast Equiaxed Nickel-Based Superalloys. Ph.D. Thesis. Mechanical Eng. Dept. Case Western Reserve University, May,1986.

[7] Malkin, S. Grinding of Metals, Theory and Application. Journal of Applied Metalworking, Jan. 1984, 95-109.

[8] Pan, J.Y. and Tenenbaum, M.J. P.I.E.S An Engineer's " Do-It-Yourself " Knowledge System for Interpretation of Parametric Test Data. proc. AAAI 1986, 836-843.

[9] Pao, Yoh-Han. Some Views on Analytic and Artificial Intelligence Approaches. Proc. IEEE Workshop on Intelligent Control, August, 1985.

[10] Siemens, R.W., Golden M. and Ferguson J.C. StarPlan II: Evolution of san Expert System. proc. AAAI 1986, 844-849.

[38]

A Simulation-Based Prediction Mechanism for An Expert Factory Control System

Szu-Yung David Wu
Department of Industrial Engineering
Lehigh University, Bethlehem, PA 18015

ABSTRACT

Expert systems are currently being touted as a mean of resolving manufacturing scheduling problems. However, most of the existing design philosophies for expert systems do not lend themselves to real-time control environments. Hence that the expert systems developed to date are neither generic nor are they responsive enough to be used for on-line system control.

In this paper, an architecture is outlined which takes advantage of both expert system technology and discrete event simulation. The simulation is used as a prediction mechanism to evaluate several possible control alternatives provided by the expert system. A performance measure is obtained from the simulation for each of the suggested alternatives, the control decisions are then made based on the measure. This performance measure is worth a great deal of domain-specific knowledge that otherwise would have to be included in the knowledge base.

The integration of the expert system, the simulation, and the control effectors forms a system called "Multi-Pass Expert Control System" (MPECS). MPECS is designed to control automated manufacturing systems. Key elements of MPECS include:

1. An Expert System, which generates potential scheduling alternatives based on real-time shop information and scheduling knowledge.
2. A simulation-based predication mechanism, which allows the system to evaluate alternative schedules based on the system performance (i.e., use the simulation model as a source of feedback for system decision making).
3. A real-time control mechanism, which affects the control on a variety of automated machining systems.

In this paper, the frame work of MPECS is defined and the simulation-based prediction mechanism of MPECS is described in detail.

REFERENCES

1. Beenhakker, H. L., Development of Alternate Criteria for Optimality in the Machine Sequencing Problem, unpublished Ph.D. dissertation, Purdue University, 1963.
2. Conway, Richard W., Johnson, Bruce M., and Maxwell, Willian L., "An Experimental Investigation of Priority Dispatching," The Journal of Industrial Engineering, Vol. 11, P. 221, 1960.
3. Cutkosky, Mark R., Fussell, Paul S. and Milligna, Robert Jr., "Precision Flexible Machining Cells within a Manufacturing System," CMU-RI-TR-84-12, June, 1984.
4. Dar-El, E. M. and Wysk, R. A., "Job Shop Scheduling--A Systematic Approach," Journal of Manufacturing Systems, Vol. 1, No. 1, 1982.
5. Kumara, S. R. T., Joshi, S., Kashyap, R. L., Moodie, C. L. and Chang, T. C., "Expert Systems in Industrial Engineering," IJPR to appear.
6. Nilsson, N. J., Learning Machines, McGraw-Hill, New York, 1965.
7. Panwalker, S. S. and Iskander, W., "A Survey of Scheduling Rules," OR, Vol. 25, P. 45-61, 1977.
8. Stecke, K. E. and J. J. Solberg, "Scheduling of Operation in a Computerized Manufacturing System," NSF Grant No. APR74 15256, Report No. 10, 1977.
9. Wu, Szu-Yung D., An Expert System Approach for the Control and Scheduling of Flexible Manufacturing Cells, Unpublished Ph.D. dissertation, The Pennsylvania State University, 1987.
10. Wysk, R. A., Wu, S. D., Yang, N. S., "A Multi-Pass Expert Control System (MPECS) for Flexible Manufacturing Systems," Bound Volume of the Symposium on Integrated and Intelligent Manufacturing (ASME), December, 1986.

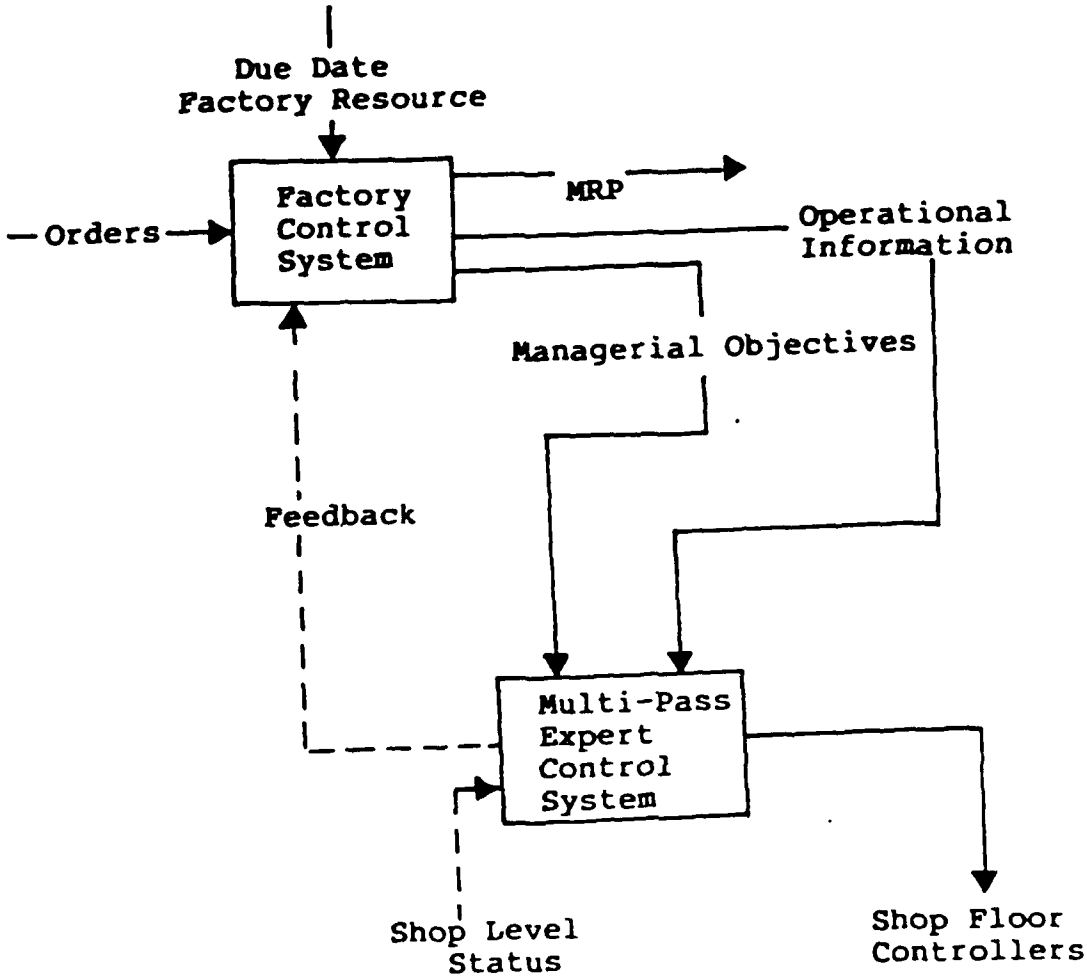


Figure 1: An Overview of MPECS

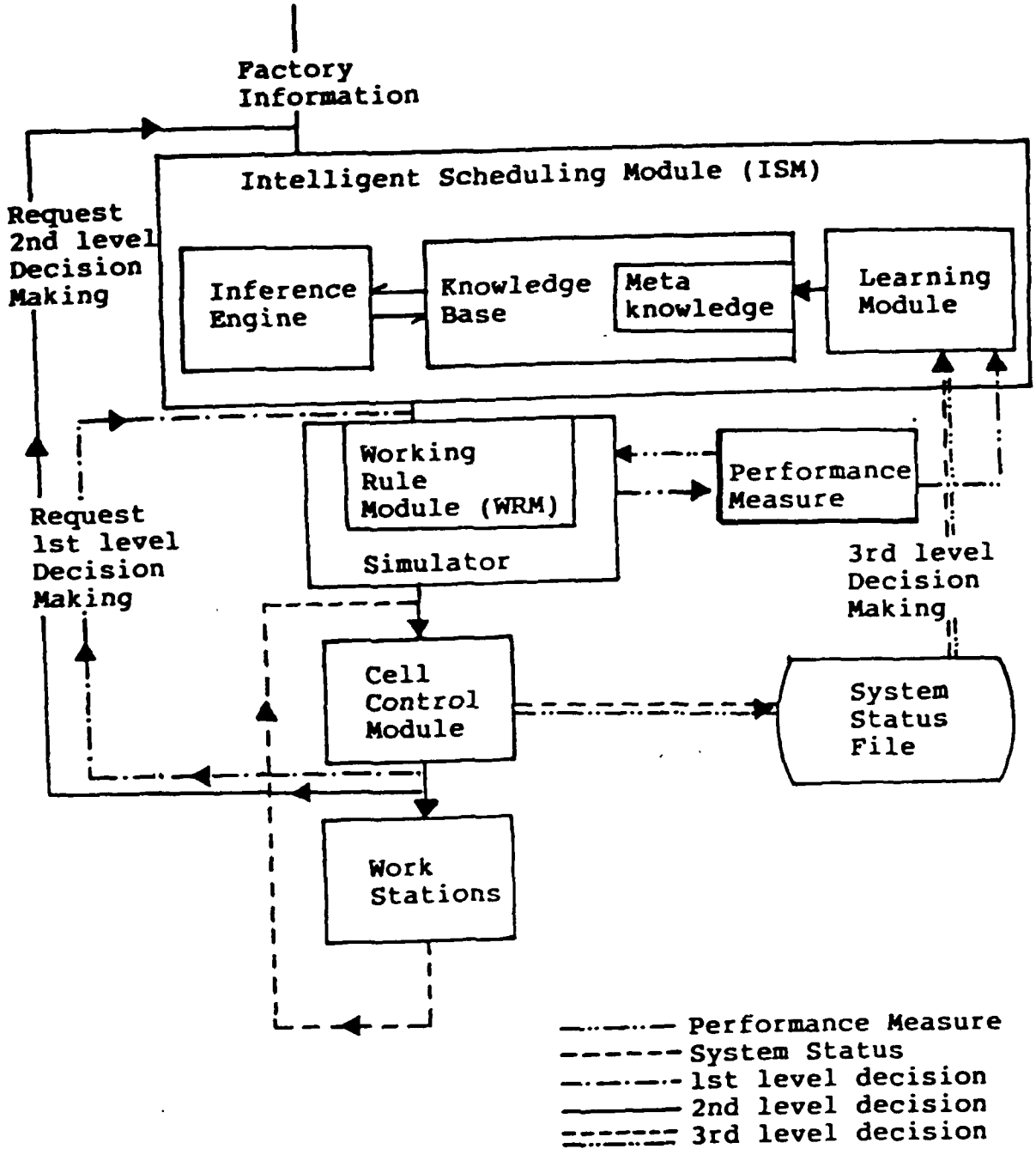


Figure 2: The General Scheme of MPECS

[39]

MINIWORLD SIMULATOR FOR MACHINE LEARNING SYSTEMS

Dan W. Patterson

Computer Science Department
University of Texas
El Paso, TX 79968

Abstract

The problem of building special training systems and environments for machine learning and other knowledge-based systems has, to date, received little attention. To gain more insight into this problem a MiniWorld Simulator is now being developed. This system will serve as a research tool in providing training scenarios for autonomous learning systems and will assist in evaluating the performance of various learning paradigms.

The area of machine learning has taken on increased importance as an automated method of collecting or eliciting expert knowledge. Autonomous learning is particularly useful when a large body of knowledge is needed or when systems must function in dynamic environments.

A number of impressive systems have demonstrated an ability for some degree of autonomous learning such as INDUCE [Hof-83], CLUSTER [Mic-83], ID3 [Qui-83], LEX [Mit-83], GIDES [Pat-87], and Meta-DENDRAL [Buc-78] to name a few. All of these systems require some form of "training" as well as feedback on the quality and extent of the knowledge "learned." In the past these functions have been provided manually by the designer or through the use of programs that have been developed specifically to test a given paradigm.

Figure 1 illustrates the major components of the MiniWorld Simulator and a typical learning system. The Simulator can provide a variety of training knowledge to a learner system. The training information is passed to the learner in incremental chunks or as a complete file. Depending on the scenario chosen, the information can be random or well-ordered, labelled or not, positive examples only or a mixture of both positive and negative examples. The examples are generated from predefined attribute domains and relations specified among the attributes or from given functions evaluated on specified domains.

For example, concept learning can be accomplished through the evaluation of descriptions of positive (or positive and negative) examples of the concept. Thus, to learn the concept "expensive objects", the simulator should generate specific examples (e.g. descriptions of gold coins, sleek cars, diamond rings) or non-examples (bread, water, potatoes, etc.) of the concept. For this, the Simulator requires a "definition" of the target concept

plus attribute descriptions for training objects, not necessarily all relevant.

If the learning task is predictive (e.g. predicting the trajectory of a target), the Simulator may be required to provide time/position examples, possibly corrupted with noise. In this case, the Simulator must be given the target "concept" and other parameters related to the environment such as the trajectory function(s) and noise desired.

The system currently has the ability to provide a variety of training examples and to evaluate the learning performance of many inductive learning systems. It can also be used to create background domain knowledge for a knowledge base. Further capabilities now under development include: (1) the storage and indexed recall of complete descriptions of objects, (2) problem solutions for analogical learning systems [Car-86], (3) general problem solutions (step-by-step examples), (4) training data for cooperative learning paradigms, and (5) the generation of simulated stimuli for many system environments.

It is also expected that training information comparable to that found in instruction textbooks can eventually be generated in a uniform format for learner systems. All of the knowledge, whether training examples or domain knowledge, is stored in a flexible frame-like structure.

The MiniWorld Simulator has already proven itself useful in testing and evaluating certain types of inductive learner systems, and it is believed the added capabilities now under development will add much to our understanding of training vehicles for a variety of other autonomous learning systems.

REFERENCES

[Buc-78] Buchanan,B.G. and E.A.Feigenbaum, DENDRAL and Meta-DENDRAL: Their Applications and Dimension, Artificial Intelligence, Vol 11, 1978.

[Car-86] Carbonell, J.G., Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition,in Michalski,R.S., J.G.Carbonell and T.M.Mitchell(eds), Machine Learning, Volumn II, Morgan Kaufmann Publishers,Inc.,Los Altos, CA, 1986.

[Hof-83] Hoff, W., Michalski, R.S.,and Stepp R.,"INDUCE/2: A Program for Learning Structural Descriptions from Examples," Technical Report No. UIUCDCS-F-83-904, Department of Computer Science, University of Illinois at Urbana-Champaign, 1983.

[Mic-83] Michalski,R.S. and R.E.Stepp, Learning from Observation: Conceptual Clustering, in Michalski,R.S., J.G.Carbonell and T.M.Mitchell(eds), Machine Learning, Tioga Publishing Co., Palo Alto, CA, 1983.

[Mit-83] Mitchell,T.M., P.E.Utgoff and R.Banerji, Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics in Michalski,R.S., J.G. Carbonell and T.M. Mitchell (eds), Machine Learning, Tioga Publishing Co.,Palo Alto, CA, 1983.

[Pat-87] Patterson,D.W., Learning to Identify Objects in Expert Systems, Proceedings of the Sixth International Phoenix Conference on Computers and Communications, Phoenix, AZ, 1987.

[Qui-83] Quinlan,J.R., Learning Efficient Classification Procedures and Their Application to Chess End Games in Michalski,R.S., J.G. Carbonell and T.M. Mitchell (eds), Machine Learning, Tioga Publishing Co., Palo Alto, CA,1983.

MINIWORLD SIMULATOR FOR MACHINE LEARNING SYSTEMS

Dan W. Patterson

Computer Science Department
University of Texas
El Paso, TX 79968

Abstract

The problem of building special training systems and environments for machine learning and other knowledge-based systems has, to date, received little attention. To gain more insight into this problem a MiniWorld Simulator is now being developed. This system will serve as a research tool in providing training scenarios for autonomous learning systems and will assist in evaluating the performance of various learning paradigms.

The area of machine learning has taken on increased importance as an automated method of collecting or eliciting expert knowledge. Autonomous learning is particularly useful when a large body of knowledge is needed or when systems must function in dynamic environments.

A number of impressive systems have demonstrated an ability for some degree of autonomous learning such as INDUCE [Hof-83], CLUSTER [Mic-83], ID3 [Qui-83], LEX [Mit-83], GIDES [Pat-87], and Meta-DENDRAL [Buc-78] to name a few. All of these systems require some form of "training" as well as feedback on the quality and extent of the knowledge "learned." In the past these

functions have been provided manually by the designer or through the use of programs that have been developed specifically to test a given paradigm.

Figure 1 illustrates the major components of the MiniWorld Simulator and a typical learning system. The Simulator can provide a variety of training knowledge to a learner system. The training information is passed to the learner in incremental chunks or as a complete file. Depending on the scenario chosen, the information can be random or well-ordered, labelled or not, positive examples only or a mixture of both positive and negative examples. The examples are generated from predefined attribute domains and relations specified among the attributes or from given functions evaluated on specified domains.

For example, concept learning can be accomplished through the evaluation of descriptions of positive (or positive and negative) examples of the concept. Thus, to learn the concept "expensive objects", the simulator should generate specific examples (e.g. descriptions of gold coins, sleek cars, diamond rings) or non-examples (bread, water, potatoes, etc.) of the concept. For this, the Simulator requires a "definition" of the target concept plus attribute descriptions for training objects, not necessarily all relevant.

If the learning task is predictive (e.g. predicting the trajectory of a target), the Simulator may be required to provide time/position examples, possibly corrupted with noise. In this case, the Simulator must be given the target "concept" and other parameters related to the environment such as the trajectory

function(s) and noise desired.

The system currently has the ability to provide a variety of training examples and to evaluate the learning performance of many inductive learning systems. It can also be used to create background domain knowledge for a knowledge base. Further capabilities now under development include: (1) the storage and indexed recall of complete descriptions of objects, (2) problem solutions for analogical learning systems [Car-86], (3) general problem solutions (step-by-step examples), (4) training data for cooperative learning paradigms, and (5) the generation of simulated stimuli for many system environments.

It is also expected that training information comparable to that found in instruction textbooks can eventually be generated in a uniform format for learner systems. All of the knowledge, whether training examples or domain knowledge, is stored in a flexible frame-like structure.

The MiniWorld Simulator has already proven itself useful in testing and evaluating certain types of inductive learner systems, and it is believed the added capabilities now under development will add much to our understanding of training vehicles for a variety of other autonomous learning systems.

REFERENCES

[Buc-78] Buchanan,B.G. and E.A.Feigenbaum, DENDRAL and Meta-DENDRAL: Their Applications and Dimension, Artificial Intelligence, Vol 11, 1978.

[Car-86] Carbonell, J.G., Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition,in

Michalski,R.S., J.G.Carbonell and T.M.Mitchell(eds), Machine Learning, Volumn II, Morgan Kaufmann Publishers,Inc.,Los Altos, CA, 1986.

[Hof-83] Hoff, W., Michalski, R.S.,and Stepp R.,"INDUCE/2: A Program for Learning Structural Descriptions from Examples," Technical Report No. UIUCDCS-F-83-904, Department of Computer Science, University of Illinois at Urbana-Champaign, 1983.

[Mic-83] Michalski,R.S. and R.E.Stepp, Learning from Observation: Conceptual Clustering, in Michalski,R.S., J.G.Carbonell and T.M.Mitchell(eds), Machine Learning, Tioga Publishing Co., Palo Alto, CA, 1983.

[Mit-83] Mitchell,T.M., P.E.Utgoff and R.Banerji, Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics in Michalski,R.S., J.G. Carbonell and T.M. Mitchell (eds), Machine Learning, Tioga Publishing Co.,Palo Alto, CA, 1983.

[Pat-87] Patterson,D.W., Learning to Identify Objects in Expert Systems, Proceedings of the Sixth International Phoenix Conference on Computers and Communications, Phoenix, AZ, 1987.

[Qui-83] Quinlan,J.R., Learning Efficient Classification Procedures and Their Application to Chess End Games in Michalski,R.S., J.G. Carbonell and T.M. Mitchell (eds), Machine Learning, Tioga Publishing Co., Palo Alto, CA,1983.

A Distributed Simulation Environment - the Intelligent Vehicle Workstation

[43]

Kevin J. Lehnert

Daniel M. Donahue
Stanley K. Hill
Marion Lineberry
Michael Sullivan

Texas Instruments, Incorporated
Artifical Intelligence Laboratory
PO Box 226015, M S 238
Dallas, Texas 75266

Abstract:

The goal of the Intelligent Vehicle Workstation (IVW) is to provide a realistic, distributed simulation environment for the prototyping of intelligent vehicle systems. By simulating vehicle behavior, the user can develop systems of superior fidelity, thus reducing the difficulty of the transition from simulated to actual systems. An object-oriented approach to simulation has been adopted to achieve these goals. The currently operational system (see Donahue86, Sullivan87) uses a time-driven simulation paradigm. Conversion to discrete-event simulation (see Lehnert87) is underway.

The underlying physical architecture of the simulation system consists of a network of software modules interfacing to a central communication system (See Figure 1). The specific structure of the simulation system design is mapped onto this architecture (See Figure 2). Each box in Figure 2 corresponds to a module instance in the physical domain. The flexibility of the physical system is such that very few constraints module are imposed on the internal structure of the simulation subsystems. A single simulation could contain modules implemented in a variety of ways, from FORTRAN subroutines to knowledge-based expert system shells. A simple interfacing procedure translating messages passed over the communication system to function calls in the target module is all that must be provided.

Every simulation environment consists of a world, a set of platforms, and a simulation manager controlling interactions (See Figure 3). The world is a set of data representing the true state of the simulation milieu as terrain surfaces, weather conditions, and static and dynamic world objects. A platform is any object in the world capable of independent action, that is, capable of posting events. A vehicle is one type of platform. The interactions between separate platforms, as well as those between a platform and the world, are constrained by physical principles and uncertainty. The role of the simulation manager is to enforce real world interaction constraints. The interactions take the form of time-stamped

events. All interactions are reduced to one of a small set of generic event types. Coordination of multiple platforms running in a distributed system is achieved by chronological processing of events. A time horizon is used to determine an interval in which no new events will be posted for processing events. This technique is similar to that described in Misra and means that the simulation appears continuous as the maximum allowable event duration approaches zero.

Multiple levels of system development are supported by IVW. Shells are provided for the user at every level of development so that user contribution can be limited to those facets of the domain which are of particular interest. Libraries of individual vehicle subsystems and procedures for creating new subsystems are available for the internal configuration of a vehicle. The existing IVW has been extended to provide the simulation of multiple platforms over a series of machines. See Figure 4. This extension has allowed us to examine issues involved with physically distributing the simulated agents, such as communication packet frequency and size, performance improvements, and synchronous control in a unified world.

References

Donahue, D. *An Autonomous Vehicle Workstation*. AAAI Workshop on AI and Simulation. August 1986.

Lehnert, K., Lineberry, M. *A Simulation Environment for the Development of Intelligent Vehicle Systems*, to appear in the Proceedings of the National Aerospace and Electronics Conference, May 1987.

Misra, J. *Distributed Discrete Event Simulation*. ACM Computing Surveys, Volume 18, Number 1, March 1986, pp 39-64.

Sullivan, M., Donahue, D., Hull, S. *Simulation of Multiple Platforms Using an Intelligent Vehicle Workstation*, to appear in the Proceedings of the National Aerospace and Electronics Conference, May 1987.

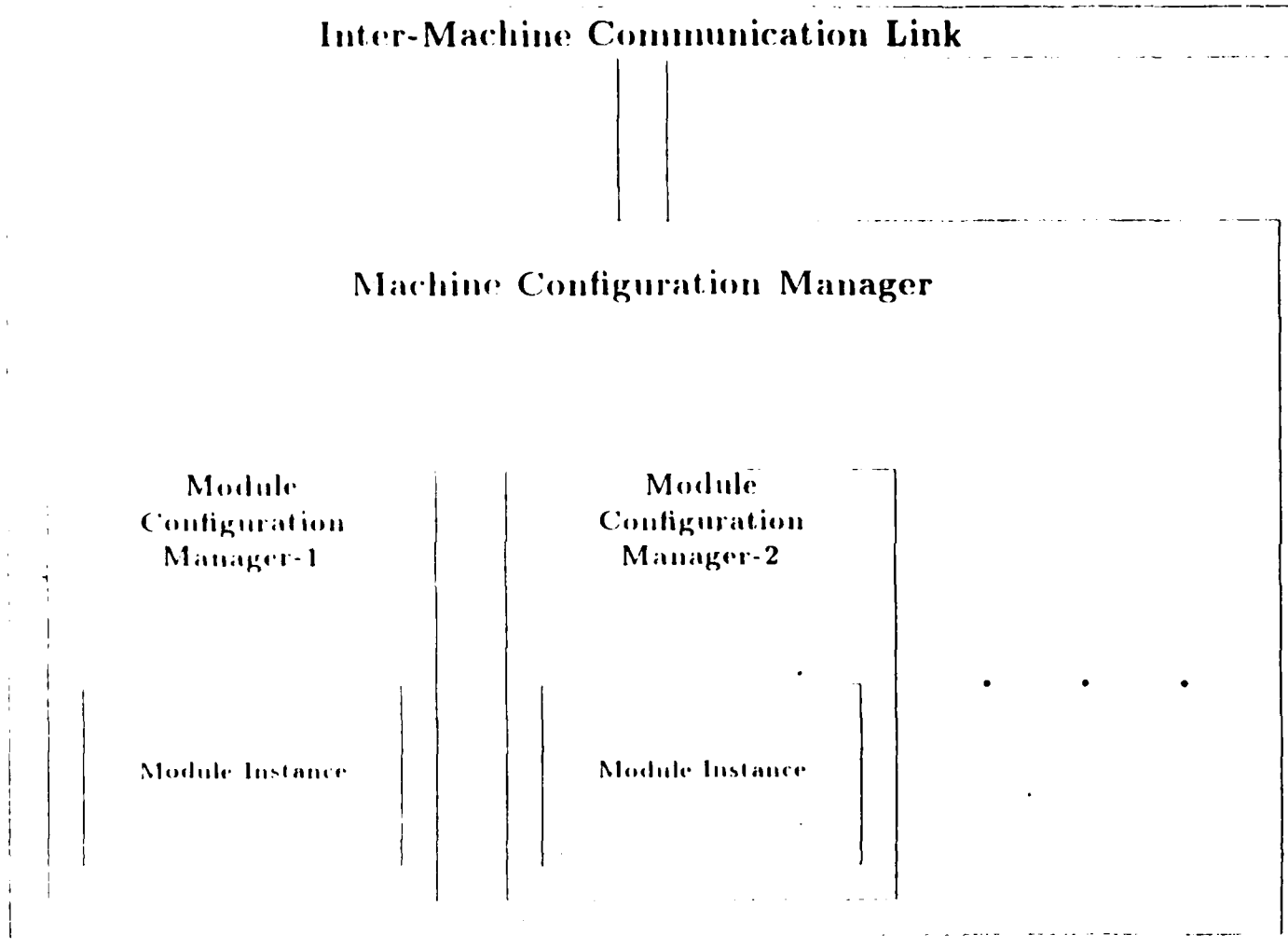


Figure 1. Physical Communication Hierarchy

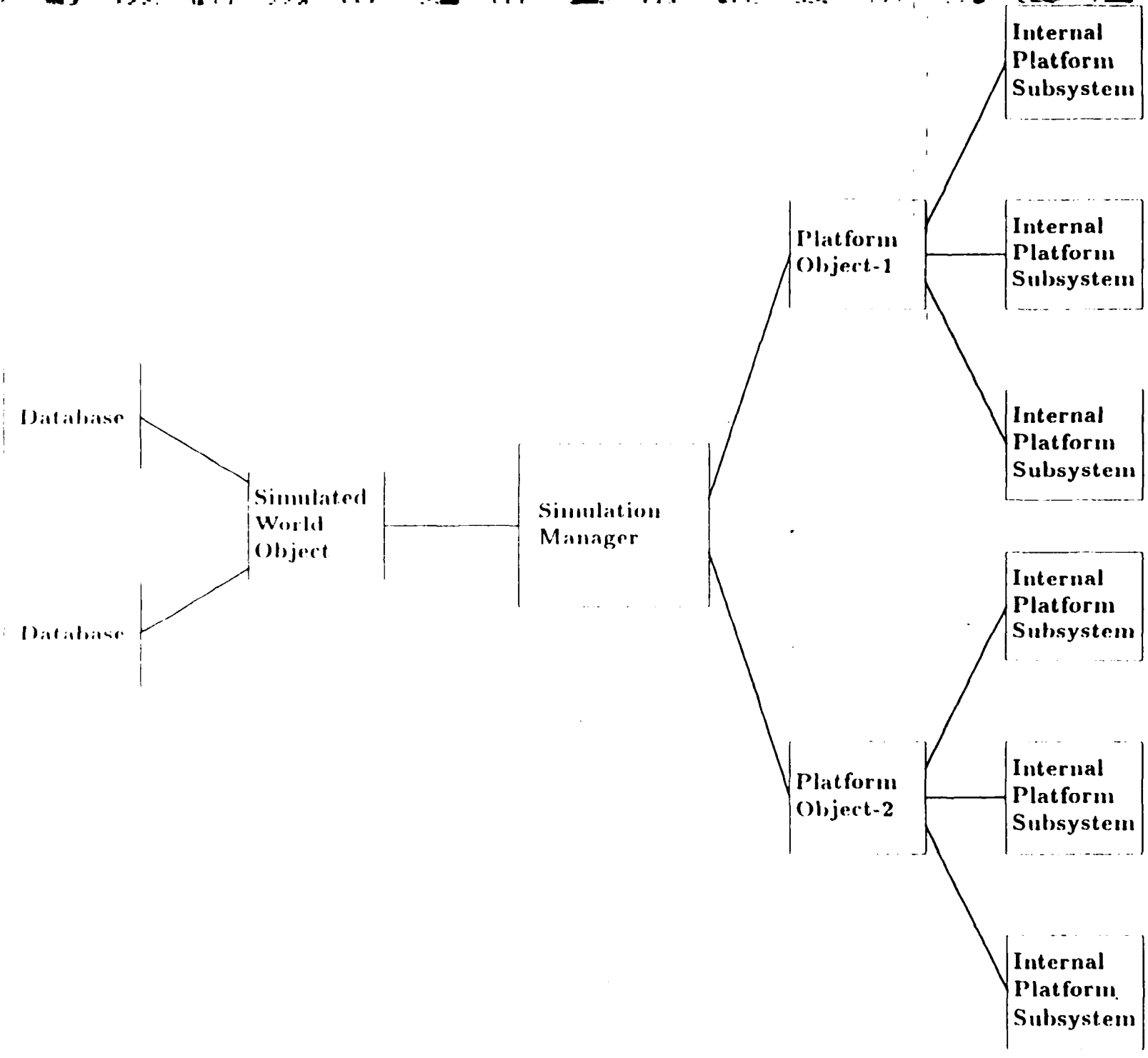


Figure 2. Functional Communication Hierarchy

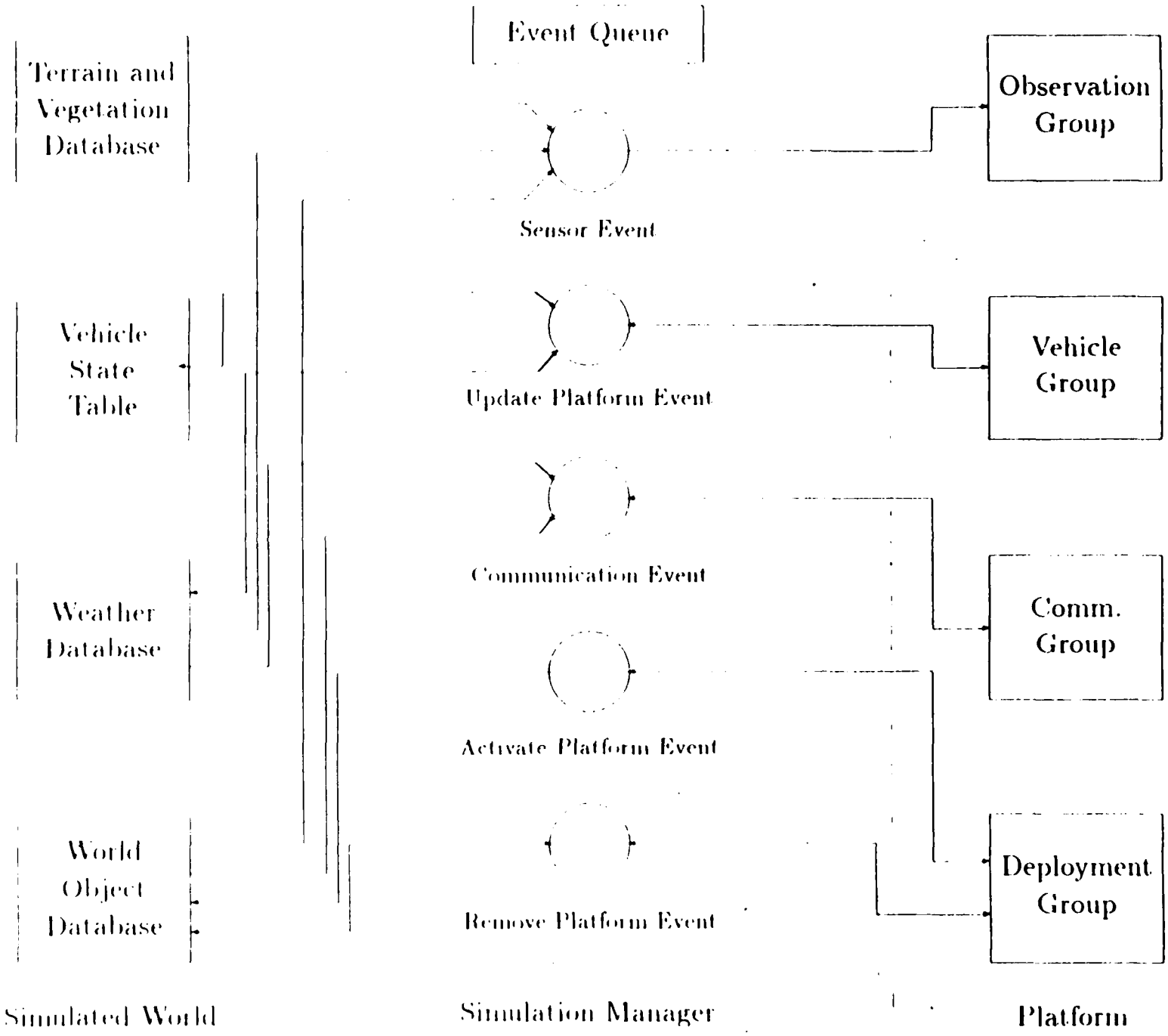


Figure 3. Overall Simulator Structure

Sensors
Current Sensor
Add SENSOR
Shift Sensors

ON current sensor
OFF current sensor
Sensor Attitude
EDIT current sensor
EDIT sensor

UPDATE current sensor
UPDATE sensor
Draw sensor footprint

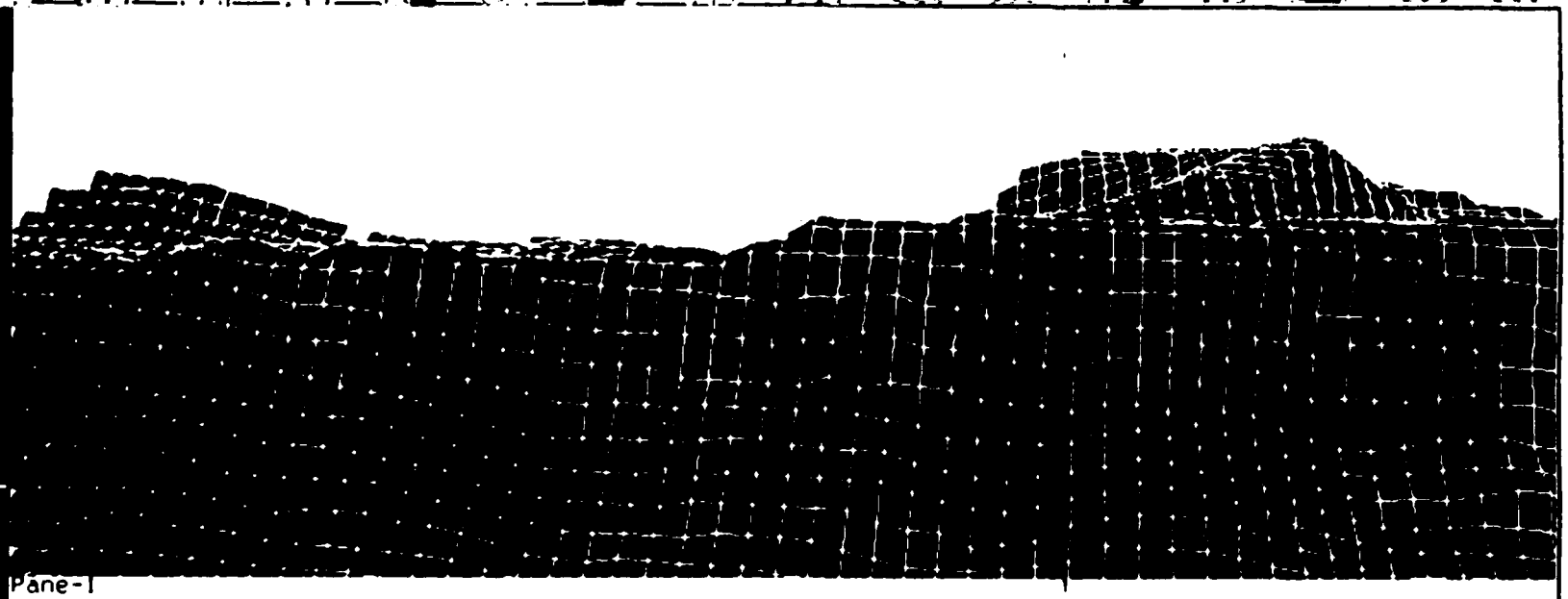
Platform menu
Object Definition Menu

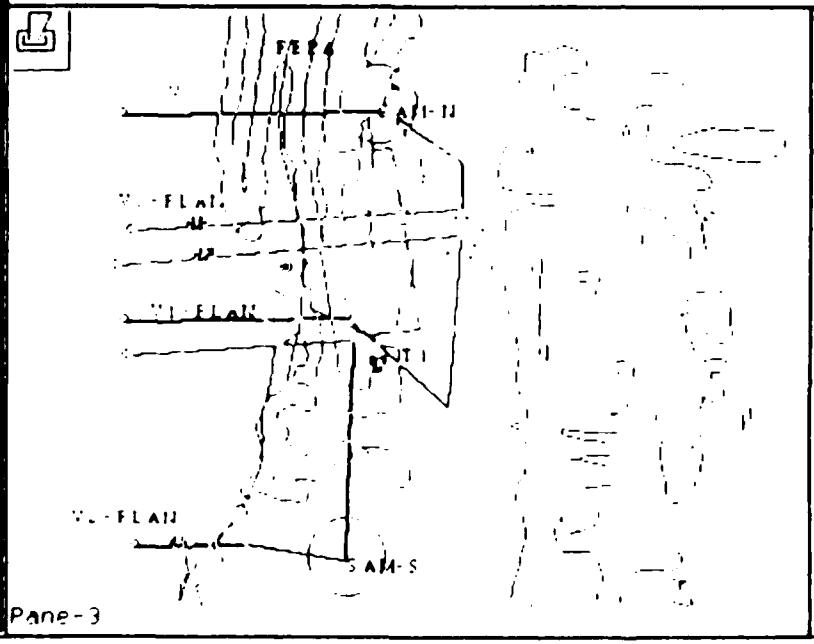
LINDY
Select a new vehicle
Choose Configuration
Previous menu selections

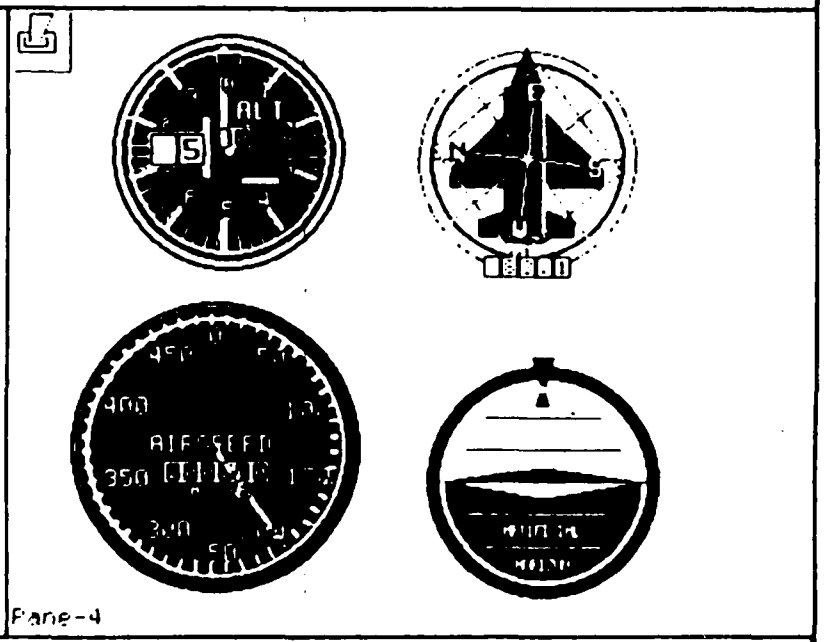
Break Process menu
Configuration menu
Color menu
Eye menu
Global Planner menu
Observer menu
Pilot menu
Platform menu
Save/Restore menu
World Object(s) menu

INITIALIZE
CONTINUOUS
SINGLE STEP
N STEPS

EDIT Time







OFF

Yo

Lisp Listener 3

Main | S/R | Platform | Cockpit | Planner | Pilot | Eye

Fig 41

[44]

QSOPS: An Integrated Knowledge-Based Environment for the Qualitative Simulation of Physical Systems

Alfred D. Round
Knowledge Systems Laboratory
Stanford University
701 Welch Road, Building C
Palo Alto, California 94304 U.S.A.
Telephone: 415-725-3849
E-mail Address: round&sumex-aim.stanford.edu

Abstract

Much of the recent work in the qualitative simulation of physical systems focuses on the prediction of future states by applying constraints or "influence relationships" between objects, or by running "propagation/prediction cycles" on the current system state[1,2,3]. Implicit in these paradigms is the notion that a simple physical system can be described in terms of qualitatively distinct stages. The traditional approach to simulation of more complex physical systems is through mathematical modeling, often resulting in a set of differential equations that is difficult to solve and whose parameters are not intuitively related to the system being simulated. The major research issue addressed here is how to combine the best features of both approaches so that the states of complex physical systems are continuously variable and expressible in precise quantitative terms, while the relationships that generate these states retain a simple, heuristic, intuitive flavor.

The QSOPS system addresses this issue by providing an intelligent environment for the qualitative simulation of processes in the domain of bacterial molecular genetics. Using QSOPS, a molecular biologist can interactively describe the processes and objects of interest

and predict the future states of the resulting molecular systems by simulating these processes. QSOPS consists of three components whose integration provides a unified simulation environment:

- A Description component that allows the user to tell the computer about the processes to be simulated, the objects that these processes act on, and global information about the state of the system. The user description of a process is converted into a representation that captures the specific ways that objects interact while undergoing the process, the temporal relationships among sub-processes, and the number of simultaneous occurrences of the process. The representation of an object reflects the spatial relationship of its sub-objects as well as attributes such as concentration and lifetime; several different graphical representation types for objects are also supported.

- A Simulation component that predicts future states of objects when the processes that act on them are simulated. The theory of simulation used by QSOPS is a simple one. A given process creates new objects and/or destroys old objects at a certain frequency. In addition, each object has a (usually finite) lifetime. An object is thus created by a process, and is either destroyed by a process or expires at the end of its lifetime. The simulation of a process then consists of tracking the creation and destruction of the input and output objects of the process based on the process frequency and the object lifetimes. The process frequency and object lifetimes are determined by rules which dynamically compute them as functions of object attribute values.

- An Animation component that dynamically renders the motion of molecules undergoing a process, using object images and interaction types specified in the Description component and process frequencies computed in the Simulation component.

A set of experiments using the QSOPS environment has been conducted to predict the bacterial concentration of the amino acid tryptophan over time by simulating the transcription, translation, and repression of the genes responsible for its synthesis and regulation. Simulation of these processes under normal conditions as well as with perturbations such as mutations yields results which are consistent with bacterial molecular genetic theory[4,5].

References:

1. deKleer, J. and J.S. Brown, "A Qualitative Physics Based on Confluences," in "Formal Theories of the Common-Sense World", Ablex, 1983.

2. Forbus, K.D., "Qualitative Process Theory," Artificial Intelligence Laboratory, AIM-664, Cambridge: M.I.T., 1982.

3. Kuipers, B., "Commonsense Reasoning about Causality: Deriving Behavior from Structure," Tufts University Working Papers in Cognitive Science No. 18, May, 1982.

4. Friedland, P. and Y. Iwasaki, "The Concept and Implementation of Skeletal Plans," Journal of Automated Reasoning 1(1985), pp. 161-208.

5. Koton, P., "Towards a Problem Solving System for Molecular Genetics," Technical Report 338, MIT Laboratory for Computer Science, Cambridge, Massachussetts, 1985.

[45]

Simulation and Knowledge Systems

Michael Greenberg
Paul Cohen

Experimental Knowledge Systems Laboratory
Department of Computer and Information Science
Lederle Graduate Research Center
University of Massachusetts
Amherst MA, 01003

April 30, 1987

Simulation can be used to assist problem solving processes in knowledge systems. Three important roles for simulation are prediction, providing a source of "deep knowledge," and evaluation. For example, in the context of fighting forest fires, simulation can predict the movement of a fire and the effects of building a fire break. As a source of deep knowledge, simulation can be used to determine how actions interact and find unanticipated side effects. As a tool for evaluation, simulation can provide many test cases. A forest fires' behavior can be easily represented as a simulation, but not easily with rules—especially when knowledge about specific terrain is taken into account.

We are exploring how knowledge that can be easily represented in a simulation can be used by a knowledge system. To this end we are building a simulation-based forest fire fighting system. The task of this system is to plan and execute actions for fighting a forest fire. Simulation is used to

- model the actual fire and to keep track of the effects forest fire fighting efforts (i.e. fire breaks and location of crews),
- provide the planner with predictions about the fires' behavior,
- anticipate the effects of proposed actions,
- provide a test-bed for different forest fire fighting strategies and tactics, and
- generate test cases for evaluating performance.

As part of our overall goal of understanding the role of simulation in knowledge systems, we are currently exploring several research issues.

- How can simulation technology integrated with knowledge system technology? What does a system which combines the two look like?
- What functionality should the simulation provide? How much control of the actual simulation is necessary? Can the knowledge system control the granularity of the simulation?
- What knowledge about the simulator must the knowledge system have? Does it have to know about the strengths and weaknesses of the simulation? Does it need to know about the disparity between the simulation model and the real world?
- How does the quality of the simulation affect the performance of the overall system? Is there enough knowledge in a simple simulation to benefit a knowledge system?
- Is there less work involved in implementing a knowledge system plus a simulator, or a knowledge system alone? If simulations are easy to implement, we may find that knowledge system development time is reduced.

During this presentation, we will discuss the roles of in knowledge systems and the of our simulation-based forest fighting system.

[46]

CONFIG Project: A Generic Tool for Qualitative-Simulation-Based Reasoning about Configurations of Engineered Systems

Jane T. Malin
Bryan D. Basham
Systems Development and Simulation Division
Engineering Directorate
NASA-Lyndon B. Johnson Space Center
Houston, TX 77058

Rick Harris
MITRE Corporation
1120 NASA Rd. 1
Houston, TX 77058

Designing, testing, and operating engineered devices requires analysis of the effects of failures and procedures as they propagate through device configurations. Such analysis is required in development of failure management expert systems, and in failure modes and effects analysis (FMEA). Information about device configuration and operating modes is used to predict effects of local changes in components on the device as a whole, and to plan how to diagnose failures and recover from them [1]. Early in design and testing, many of these analyses are performed mentally by engineers on the basis of qualitative information.

The purpose of the CONFIG project is to develop a generic device modelling tool for use in discrete event simulation, to support such analyses. The tool permits engineers to graphically model device configuration (components and connections) and qualitatively specify local operating modes of components. Computation and specification requirements are reduced by focussing the level of component description on operating modes and failure modes, and specifying qualitative ranges of component variables relative to mode transition boundaries. Using an approach similar to Pan [2], discrete events are defined at the level of changes in operating modes. A time-step approach is avoided, and simulation processing need occur only when modes change or variables cross qualitative boundaries.

As in the work of Towne et al [3], the tool supports the development of graphical libraries of component models, and permits an author to build device models graphically. A model is built by using component objects from a library, and connecting them at ports with graphical relations that define data flow between components.

The core of a component model is its state-transition diagram, which graphically specifies modes of operation (both normal and failed) and the transitions among them. State transitions and within-state variable transformations are specified by process statements. Process statements have three parts: invocations (preconditions for effects execution), effects (executed if all invocations are satisfied), and delays corresponding to each effect (effect completions scheduled at increments to the current time).

A key capability is a process language constructor and interpreter that permits process statements to be written with an array of qualitative and

quantitative syntaxes, including table lookups. Component variables can be specified quantitatively or qualitatively, but a small number of qualitative ranges is desirable (e.g., abnormal-low, low, medium, high, abnormal-high).

The event structure controls the propagation of behavior changes among components. Scheduled events pass data between ports, change variable values, and make state transitions. The primary event is the update of a component, which is triggered by a change in an input variable, local variable, or component state. In such an event, appropriate processes are inspected, and the effects of invoked processes are scheduled with corresponding delays. Updates originating from many components can be scheduled at the same time on the discrete event clock.

CONFIG is implemented on a LISP machine, using a discrete event simulator and object-oriented knowledge engineering software. CONFIG has been used to build device models in two domains, digital circuits and thermal systems.

References

Malin, J. T., and N. Lance, Processes in Construction of Failure Management Expert Systems from Device Design Information. Submitted for review, 1987.

Pan, J. Y., Qualitative Reasoning with Deep-level Mechanism Models for Diagnoses of Mechanism Failures. In Proc. First Conference Art. Int. Applications, Denver, CO, Dec. 1984, pp. 295-301.

Towne, D. M., A. Munro, Q. A. Pizzini, and D. S. Surmon, Representing System Behaviors and Expert Behaviors for Intelligent Tutoring. Univ. So. Calif. Behavioral Technology Laboratories, Tech. Report No. 108, Feb. 1987.

EXAMPLES OF PROCESSES

State Independent

The CONDENSER.262 Unit in DRUMMAN2 Knowledge Base

Own slot FEEDBACK from CONDENSER.262
Inheritance OVERRIDE.VALUES
Effects FEEDBACK 1.EFFECT in THERMAL
Type PROCESS
Values UNKNOWN

Own slot FLOW.PRESSURE.MAJOR.DIFFERENTIAL from COMMON.PROCESSES
Inheritance OVERRIDE.VALUES
Invocation PRESSURE 1.MAJOR.DIFFERENTIAL.INVOCATION in THERMAL
Effects FLOW 1.EFFECT in THERMAL,
PRESSURE 1.CONSTANT.EFFECT in THERMAL,
CLEAR.AHEAD? 1.TRUE.EFFECT in THERMAL,
PRESSURE 1.NOMINAL.EFFECT in THERMAL
Delay 0, 0, 0, 1
Type PROCESS
Values UNKNOWN

Own slot FLOW.PRESSURE.MINOR.DIFFERENTIAL from COMMON.PROCESSES
Inheritance OVERRIDE.VALUES
Invocation PRESSURE 1.MINOR.DIFFERENTIAL.INVOCATION in THERMAL
Effects FLOW 1.EFFECT in THERMAL,
CLEAR.AHEAD? 1.TRUE.EFFECT in THERMAL,
PRESSURE 1.NOMINAL.EFFECT in THERMAL
Type PROCESS
Values UNKNOWN

State Dependent

(Heat Sink Failure State of Condenser)

(Output) The CONDENSER.HEAT.SINK.FAILURE Unit in THERMAL Knowledge

Member slot: PRESSURE.FLOW.CLEAR from CONDENSER.HEAT.SINK.FAILURE
Inheritance: OVERRIDE.VALUES
Invocation: FLOW 1.INVOCATION, CLEAR.AHEAD? 1.INVOCATION,
PRESSURE 1.NO.DIFFERENTIAL.INVOCATION
Effects: PRESSURE 1.CONSTANT.EFFECT
Type: PROCESS
Values: UNKNOWN

Member slot: PRESSURE.FLOW.NOT.CLEAR from CONDENSER.HEAT.SINK.FAILURE
Inheritance: OVERRIDE.VALUES
Invocation: FLOW 1.INVOCATION, CLEAR.AHEAD? 1.F.INVOCATION,
PRESSURE 1.NO.DIFFERENTIAL.INVOCATION
Effects: PRESSURE 1.HIGH.EFFECT
Type: PROCESS
Values: UNKNOWN

Member slot: PRESSURE.NO.FLOW.CLEAR from CONDENSER.HEAT.SINK.FAILURE
Inheritance: OVERRIDE.VALUES
Invocation: FLOW 1.NO.INVOCATION, CLEAR.AHEAD? 1.INVOCATION,
PRESSURE 1.NO.DIFFERENTIAL.INVOCATION
Effects: PRESSURE 1.LOW.EFFECT
Type: PROCESS
Values: UNKNOWN

State Transition

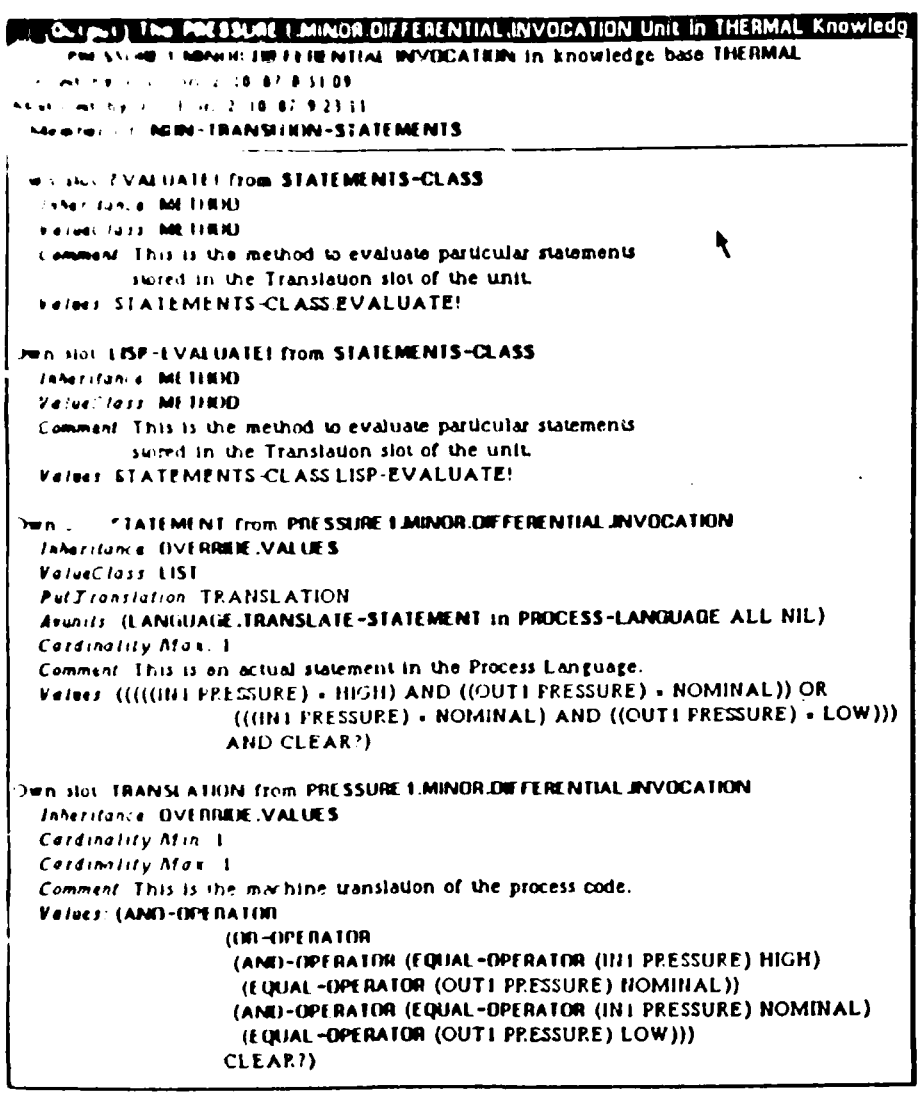
(Takes the Condenser from a Heat Sink Failure State to its Nominal State)

The CONDENSER.HEAT.SINK.FAILURE Unit in THERMAL Knowledge Base

Member slot: CONDENSER.HEAT.SINK.FAILURE.TO.CONDENSER.NOMINAL from CONDENSER.HEAT.SINK.FAILURE
Inheritance: OVERRIDE.VALUES
Invocation: CONDENSER.HEAT.SINK.FAILURE.TO.NOMINAL.INVOCATION
Effects: CONDENSER.TO.NOMINAL.EFFECT
Type: PROCESS
Values: UNKNOWN

PROCESS LANGUAGE STATEMENTS

Example of an INVOCATION



(Output) The PRESSURE.1.MINOR.DIFFERENTIAL.INVOCATION Unit in THERMAL Knowledg

PRESSURE.1.MINOR.DIFFERENTIAL.INVOCATION in knowledge base THERMAL

Member Of: BEGIN-TRANSITION-STATEMENTS

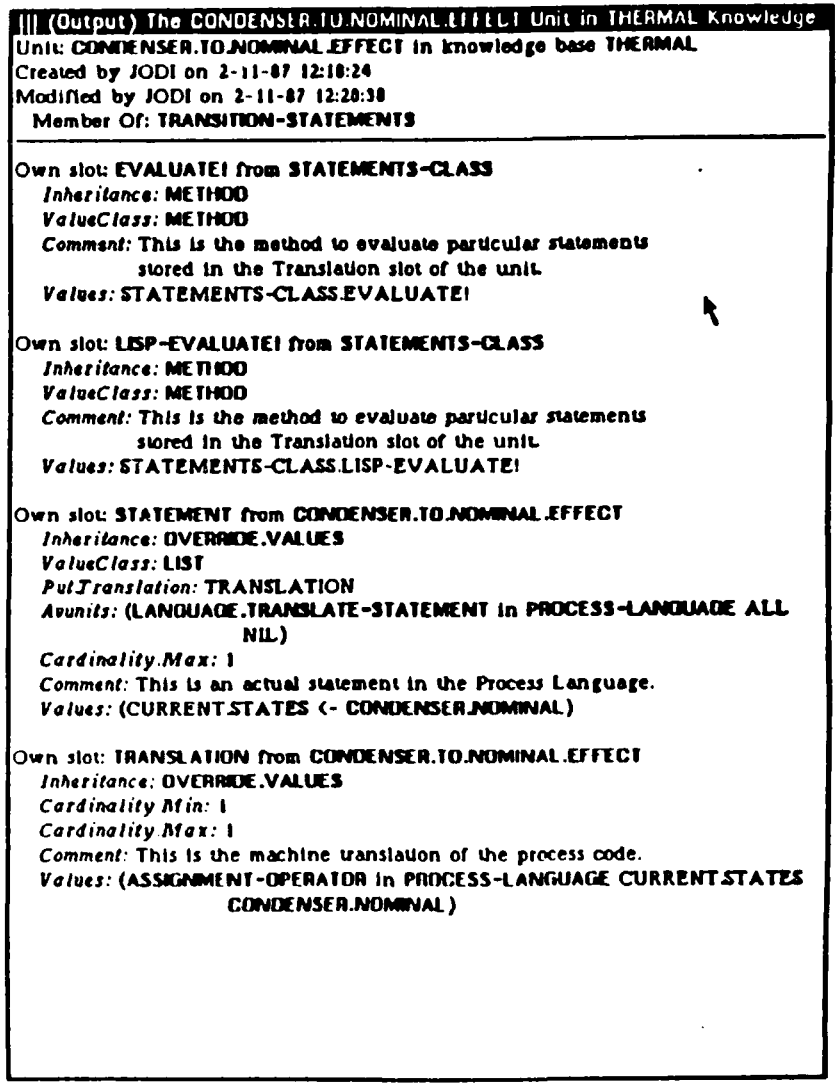
Own slot: EVALUATE! from STATEMENTS-CLASS
Inheritance: METHOD
ValueClass: METHOD
Comment: This is the method to evaluate particular statements stored in the Translation slot of the unit.
Values: STATEMENTS-CLASS.EVALUATE!

Own slot: LISP-EVALUATE! from STATEMENTS-CLASS
Inheritance: METHOD
ValueClass: METHOD
Comment: This is the method to evaluate particular statements stored in the Translation slot of the unit.
Values: STATEMENTS-CLASS.LISP-EVALUATE!

Own slot: STATEMENT from PRESSURE.1.MINOR.DIFFERENTIAL.INVOCATION
Inheritance: OVERRIDE.VALUES
ValueClass: LIST
PutTranslation: TRANSLATION
Avunits: (LANGUAGE.TRANSLATE-STATEMENT in PROCESS-LANGUAGE ALL NIL)
Cardinality.Max: 1
Comment: This is an actual statement in the Process Language.
Values: ((((IN1 PRESSURE) = HIGH) AND ((OUT1 PRESSURE) = NOMINAL)) OR (((IN1 PRESSURE) = NOMINAL) AND ((OUT1 PRESSURE) = LOW))) AND CLEAR?)

Own slot: TRANSLATION from PRESSURE.1.MINOR.DIFFERENTIAL.INVOCATION
Inheritance: OVERRIDE.VALUES
Cardinality.Min: 1
Cardinality.Max: 1
Comment: This is the machine translation of the process code.
Values: (AND-OPERATOR
((OR-OPERATOR
(AND-OPERATOR (EQUAL-OPERATOR (IN1 PRESSURE) HIGH)
(EQUAL-OPERATOR (OUT1 PRESSURE) NOMINAL))
(AND-OPERATOR (EQUAL-OPERATOR (IN1 PRESSURE) NOMINAL)
(EQUAL-OPERATOR (OUT1 PRESSURE) LOW)))
CLEAR?)

Example of an EFFECT (State Transition)



(Output) The CONDENSER.TO.NOMINAL.EFFECT Unit in THERMAL Knowledge

Unit: CONDENSER.TO.NOMINAL.EFFECT in knowledge base THERMAL
Created by JODI on 2-11-87 12:18:24
Modified by JODI on 2-11-87 12:20:38
Member Of: TRANSITION-STATEMENTS

Own slot: EVALUATE! from STATEMENTS-CLASS
Inheritance: METHOD
ValueClass: METHOD
Comment: This is the method to evaluate particular statements stored in the Translation slot of the unit.
Values: STATEMENTS-CLASS.EVALUATE!

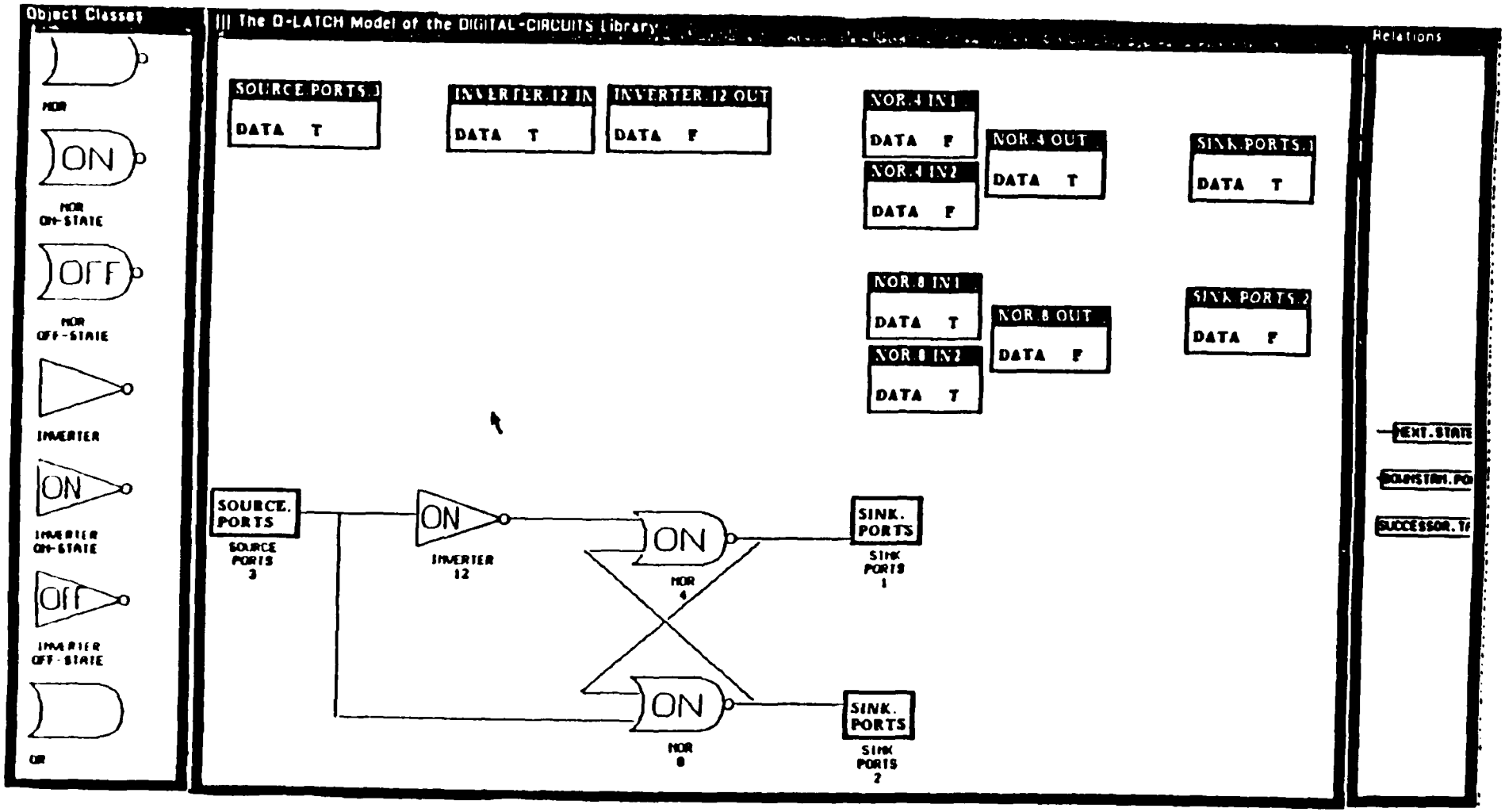
Own slot: LISP-EVALUATE! from STATEMENTS-CLASS
Inheritance: METHOD
ValueClass: METHOD
Comment: This is the method to evaluate particular statements stored in the Translation slot of the unit.
Values: STATEMENTS-CLASS.LISP-EVALUATE!

Own slot: STATEMENT from CONDENSER.TO.NOMINAL.EFFECT
Inheritance: OVERRIDE.VALUES
ValueClass: LIST
PutTranslation: TRANSLATION
Avunits: (LANGUAGE.TRANSLATE-STATEMENT in PROCESS-LANGUAGE ALL NIL)
Cardinality.Max: 1
Comment: This is an actual statement in the Process Language.
Values: (CURRENT.STATES <- CONDENSER.NOMINAL)

Own slot: TRANSLATION from CONDENSER.TO.NOMINAL.EFFECT
Inheritance: OVERRIDE.VALUES
Cardinality.Min: 1
Cardinality.Max: 1
Comment: This is the machine translation of the process code.
Values: (ASSIGNMENT-OPERATOR in PROCESS-LANGUAGE CURRENT.STATES CONDENSER.NOMINAL)

D-LATCH MODEL

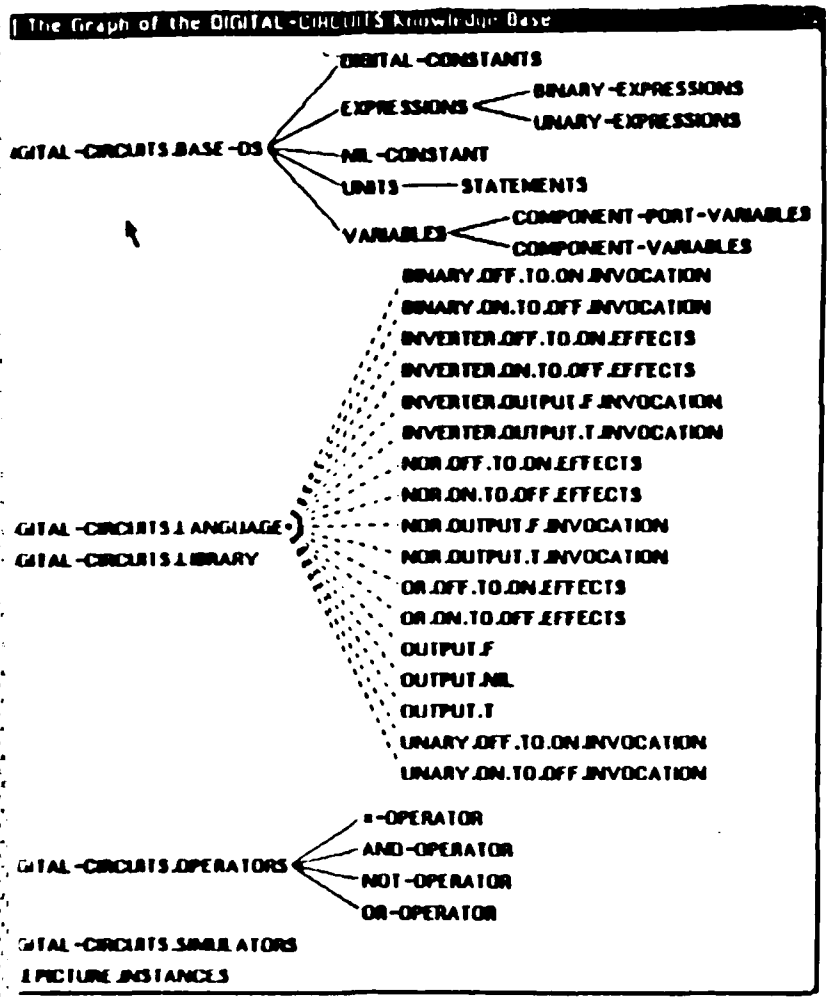
(under Digital Circuits Library)

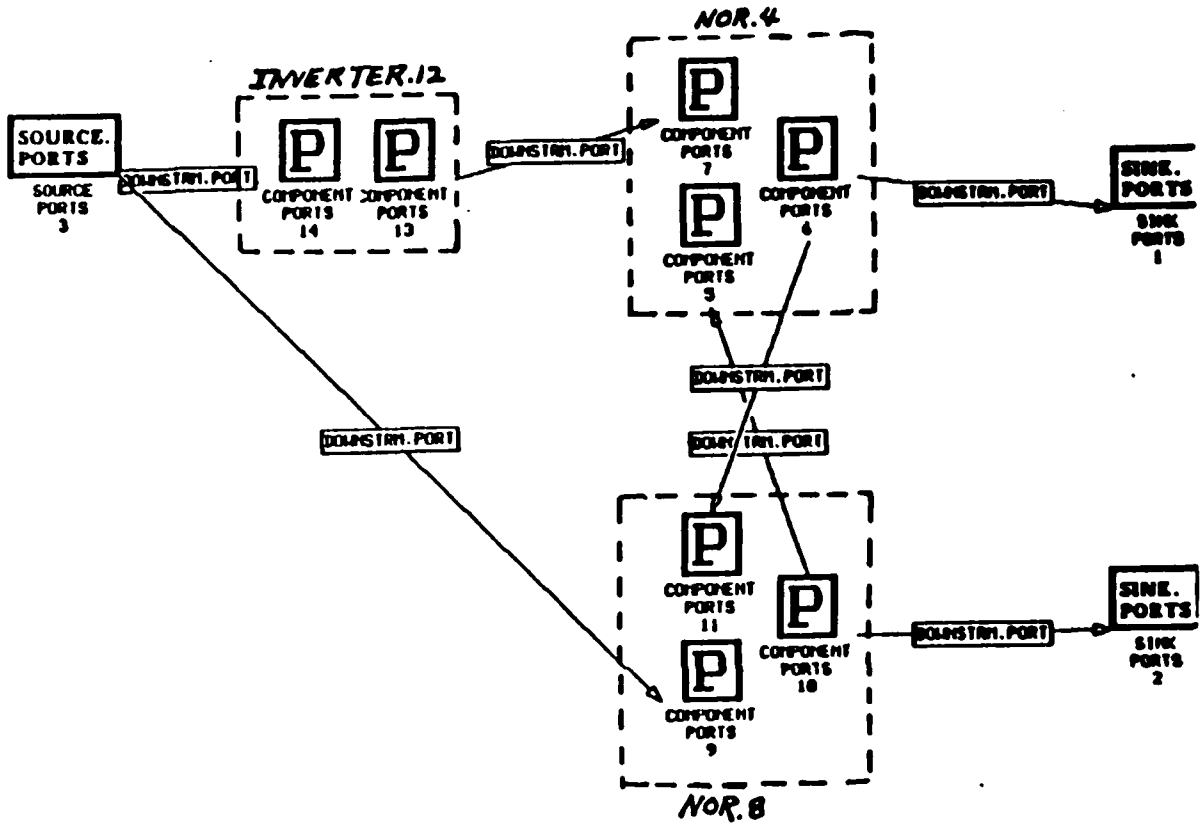


Decomposition of D-LATCH Components

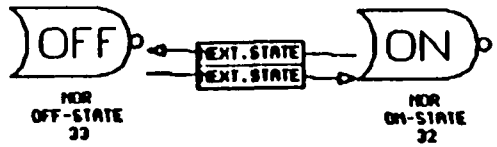
Digital Circuits Library

(Process Language Statements)



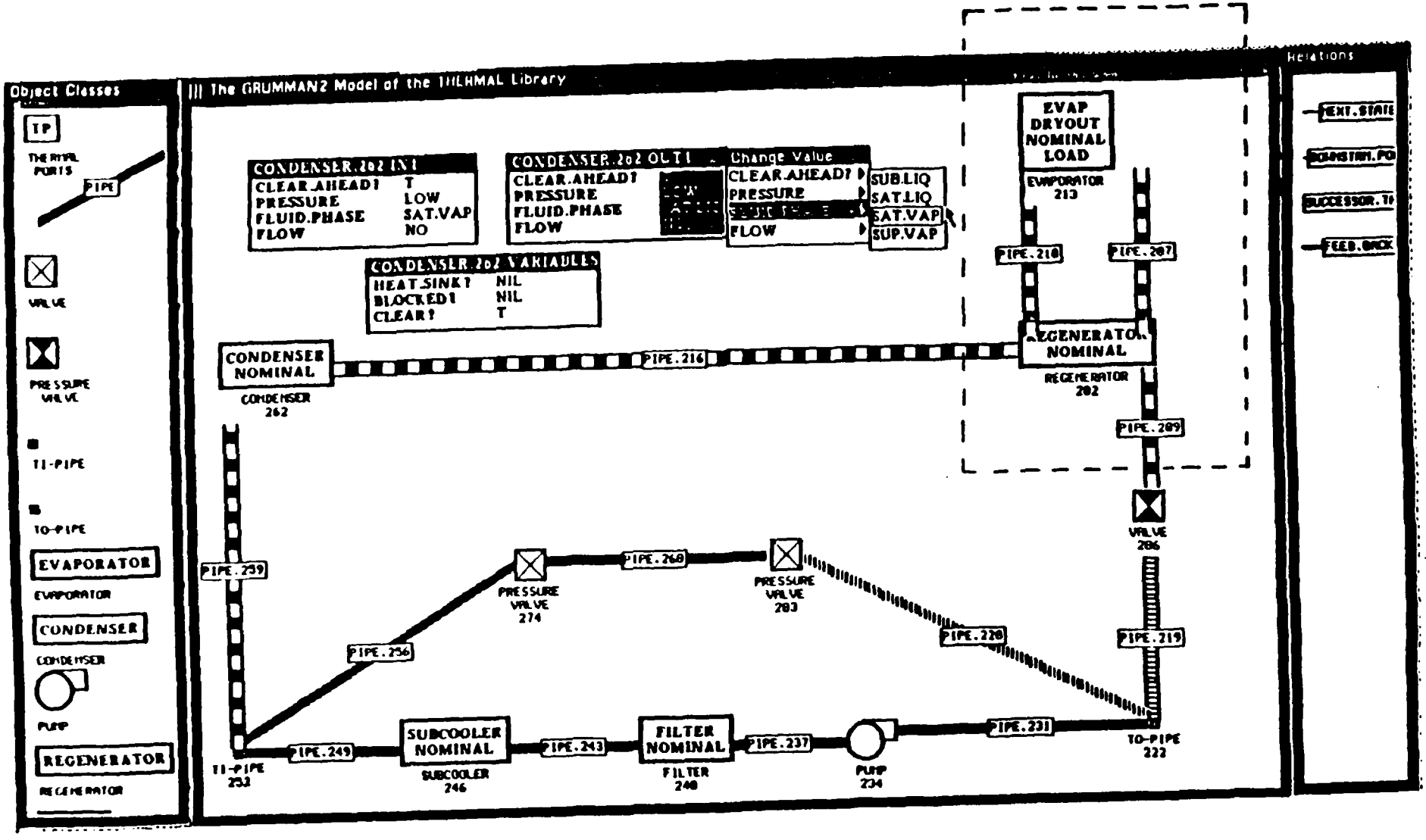


State Diagram for Nor-Gate

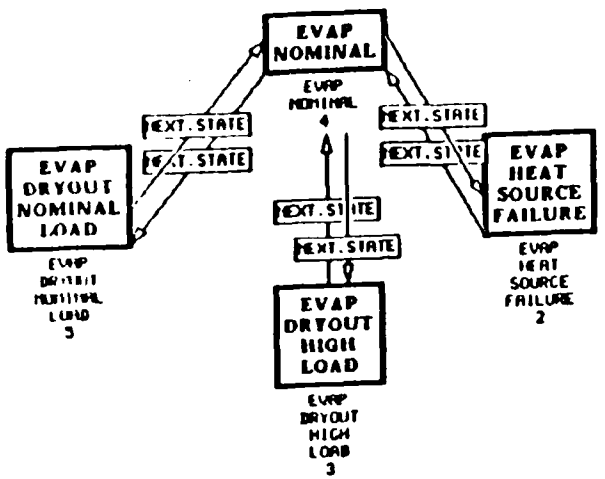


GRUMMAN BUS MODEL

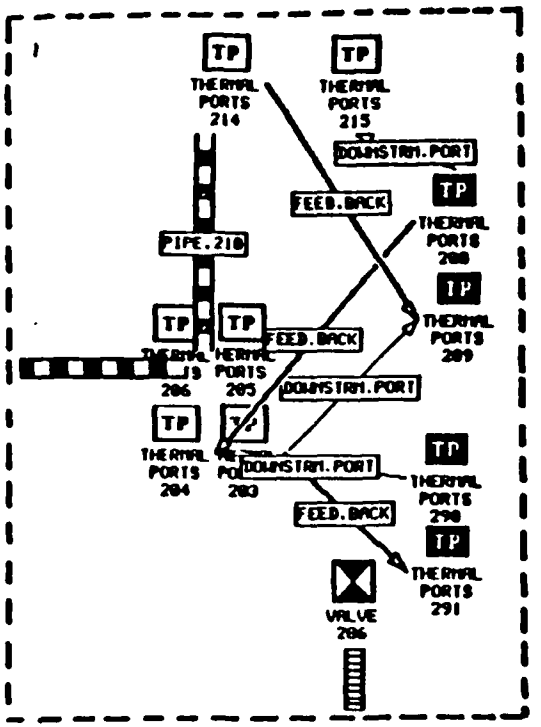
(under Thermal Library)



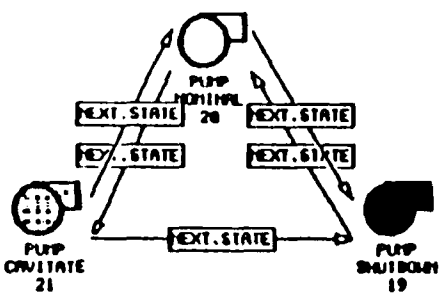
State Diagram for Evaporators



Decomposition of Evaporator.213, Pipe.207 Regenerator.202, and Pipe.209



State Diagram for Pumps



[48]

KNOWLEDGE-BASED SIMULATION OF TACTICAL ADVERSARIES

Yee-yeen Chu, Ph.D.
Azad M. Madni, Ph.D.

Perceptronics, Incorporated
21111 Erwin Street
Woodland Hills, CA 91367
(818)884-7572

ABSTRACT*

Current implementations of simulated tactical adversary do not offer the modeling flexibility or contain sufficiently deep knowledge to produce realistic adversary decision making behavior. To address this deficiency, knowledge-based expert systems technology is proposed to model and emulate the behavior of an adversary.

The tactical environment under consideration is characterized by time-varying data and event-driven tactical strategies and decisions. Such a tactical decision making environment can be characterized as a real-time problem that requires continuous reasoning in the face of time-varying, uncertain data, with changing constraints and evolving objectives.

To address these characteristics, a cognitive model was developed in support of the simulation. A key feature of the model is to simulate tactical decisions and actions with a group of active "specialists" function. These specialists conduct, as their special assignments, the situation assessment, objective formulation, planning, and control functions. In response to changes in tactical environment, each specialist monitors a variety of events and takes warranted actions. Given the real-time coordination and communication constraints, the critical problem in this environment is orchestrating the execution of the various specialists that perform distinct but related subtasks.

---

*Submitted to the Second Workshop on AI and Simulation, AAAI-87 Conference.

To this end, a "cooperating expert" architecture is developed that allows the specialists (in the form of knowledge sources) to communicate via a common knowledge base that includes the adversary's global objectives and constraints. The architecture as shown in Figure 1 is based on the blackboard model (Erman, et al, 1980; Hayes-Roth, 1983) framework. On top of this framework is the execution and scheduling module which, under the purview of a Modified Petri Net (Madni, et al., 1984), models the task execution at the level of abstraction where there is a high degree of concurrent task-related activities and tactical decisions.

Based on the model described above, a prototype intelligent adversary simulation was developed (Chu, et al, 1985; Chu and Shane, 1986), which was implemented on the Symbolics 3670. The prototype simulation system has demonstrated some basic tactical simulation and control capabilities including those summarized in Table 1. Specifically, the automated adversary has total control of its tactics, maneuvers and the various subsystems including sensors, weapons, and countermeasures. The manner in which the adversary controls these assets can be modified interactively by an experimenter/operator in the knowledge base browsing/editing mode. It thus appears that the rule-based representation of specialist functions has provided a convenient basis for constructing the interface for modifying tactical rules. This interface provides the necessary flexibility for the experimenter to modify the expertise and rules used by the simulated tactical adversary.

The prototype intelligent adversary system is upward compatible to future simulation. Future research will involve additional work in regards to three major areas: the simulation of multiple coordinated adversaries, the simulation of supporting teams, and the graphical interface that experimenter/operator were to specify, modify, and manipulate tactical targets. At present, the prototype knowledge-based adversary simulation system provides a flexible experimental tool for assessing the suitability and capabilities of expert system techniques in the tactical simulation environment.

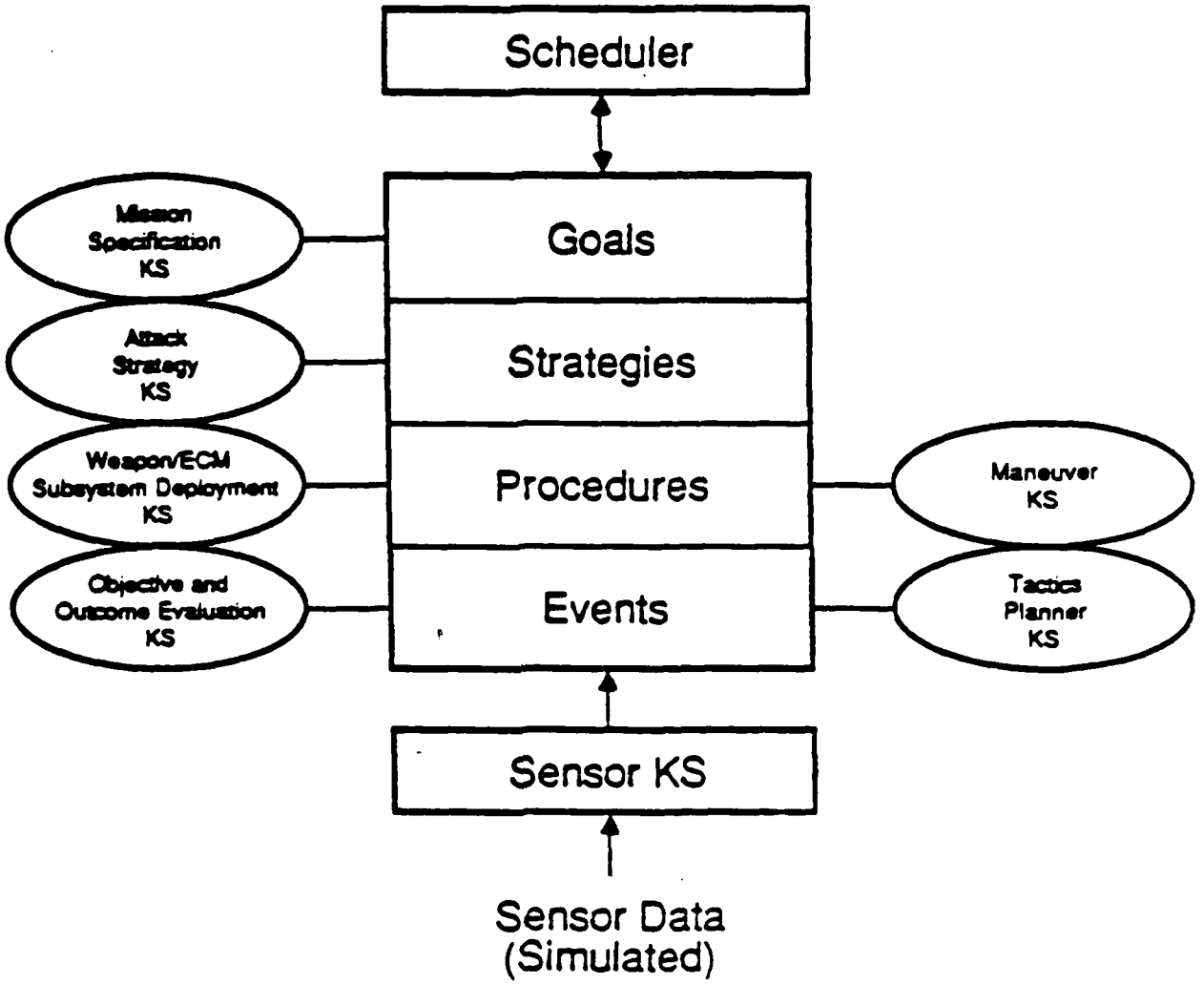


FIGURE 1.
BLACKBOARD MODEL-BASED INTELLIGENT OPPONENT
SIMULATION ARCHITECTURE

TABLE 1

CURRENT CAPABILITIES OF INTELLIGENT ADVERSARY SIMULATION

- o User Controls
  - Tactics
  - Maneuvers
  - Subsystems, including sensors, weapons, and countermeasures
- o User Monitors
  - Geopolitical Situation Display
  - Subsystem Status
  - Command Messages
- o Computerized Adversary Controls
  - Tactics
  - Maneuvers
  - Subsystems, including sensors, weapons, and countermeasures
- o Tactical Environment Simulates
  - Effects of tactical moves, maneuvers, and deployment of assets
  - Effects of decisions and non-decisions
- o Experimenter Controls
  - Operation modes, including browse/edit/execute and simulation modes
  - Situation variables and initial conditions
  - Active rule sets
- o Experimenter Monitors
  - Audit trails for the user's and adversary's decisions and actions
  - Rationale and explanation related to the adversary's moves

References

Chu, Y. and A.M. Madni. Network-Based Human Behavior Modeling for Adaptive Training. Proceedings of 1985 International Conference on Systems, Man, and Cybernetics, November, 1985.

Chu, Y., D. Shane and A. Chen. Adaptive Training for Tactical Decision Making Using Intelligent Opponent Simulation, Technical Report, Perceptronics, Inc., Woodland Hills, CA, June 1985.

Chu, Y. and D. Shane. Intelligent Opponent Simulation for Tactical Decision Making Training. Final Technical Report, PFTR-1120-86-12, Perceptronics, Inc., Woodland Hills, CA, December 1986.

Erman, L.D., F. Hayes-Roth, V.R. Lesser and D.R. Reddy. The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. ACM Computing Survey, 12:213-250, 1980.

Hayes-Roth, B. The Blackboard Architecture: A General Framework for Problem Solving. Technical Report, HPP-83-30, Heuristic Programming Project, Computer Science Department, Stanford University, May 1983.

Madni, A.M., Y. Chu, D. Purcell, and M. Brenner. Design for Maintainability with Modified Petri Nets (MPNs), Technical Report, PFTR-1125-84-11, Perceptronics, Inc., Woodland Hills, CA, November 1984.

[49]

Transformational Approach to Automated Causal Model Generation

C. N. Lee, P. Liu, M. Y. Chiu and S. J. Clark

Siemens Research and Technology Laboratories
105 College Rd. East, Princeton N.J. 08540

ABSTRACT

Causal models have been widely used in qualitative reasoning [1] [2] to explain the behavior of physical systems. Such applications of causal models have limitations due to the static nature of the models. Once models are designed considering the teleological aspects and knowledge representation of a model, the reasoning capability of model is limited by the modeling scheme to certain problem domains and to certain levels of abstraction in reasoning.

This research studies a ***transformational approach to automated causal model generation***. This approach relaxes the static nature of models by dynamically constructing them and thus facilitates different reasoning schemes for different levels of abstraction [3]. The causal modeling process is analyzed to see how the reasoning capability is related to model representation [4] and what variables should be considered in the modeling process to enhance reasoning capabilities. In this paradigm, the qualitative model for causal reasoning is regarded as a sequentially transformed object from the functional hierarchy of a physical system. The part of the system functional hierarchy, which is related to observed symptoms, is transformed into an initial causal model and the model is modified by successive transformations. Each transformation is matched to a functional level change of the system hierarchy and an abstraction level change in reasoning. The variables are qualitatively transformed, keeping the causal relations in the model transformations.

This strategy is successfully applied to our robotic assembly cell diagnosis system [5]. The functional hierarchy of a robot cell is obtained from the cell design information, i.e. by automatically translating the cell control program and functionally representing the physical devices in the robot cell. For cell diagnosis, part of the cell functional hierarchy is transformed as a basic causal model with the aid of cell failure symptoms. The causal network is automatically constructed from the system functional hierarchy and causal reasoning is performed on the causal network. During the reasoning process, backtracking may occur through the causal relation of functional hierarchy, i.e. a diagnostic test is performed to verify the suspected functional failure and proves the diagnosis to be incorrect. If diagnosis fails, then the symptom list is updated with the test result and a new causal model is generated.

The results show that this strategy can be generalized for physical system diagnosis and that various reasoning schemes can be supported for diagnosis of complex physical systems. The functional hierarchy of the system automatically makes the diagnostic knowledge base hierarchically arranged and reveals functional causal relations clearly for diagnostic system design. However, since the modeling boundary is defined by the transformational operators instead of model itself, the initial functional hierarchy should be generated carefully and the design of appropriate transformational operators is required.

References

[1] R. Davis, "Diagnosis via Causal Reasoning: Paths of Interaction and the Locality Principle," AAAI-83 Proceedings, Washington, D.C., 88-94, 1983

[2] *Qualitative Reasoning about Physical Systems*, Edited by D. G. Bobrow, The MIT Press, Cambridge, Massachusetts, 1985

[3] H. J. Levesque and R. J. Brachman, "A Fundamental Tradeoff in Knowledge Representation and Reasoning" Redaings in Knowledge Representation, Edited by R. Brachman and H. Levesque, M. Kaufmann Publishers, Los Altos, California, PP 42-70, 1985

[4] M. J. Stefik "Planning with Constraints," Stanford Heuristic Programming Project Memo HPP-80-2, 1980

[5] P. Liu, M. Chiu, S. J. Clark and C. N. Lee, "Knowledge Based Diagnosis of Robotic Workcells," Proceedings of Robots 11, Chicago, Ill., April 1987.

[52]

Goal Directed Simulation

Marc R. Halley and Daniel Pliske

Center for Machine Intelligence
TASC - The Analytic Sciences Corporation
McLean, Virginia 22124

ABSTRACT:

For the past three years, we have been involved in research on a combination of simulation and artificial intelligence that we call goal directed simulation. Many researchers have noted that simulations have been limited in their usefulness because they are only able to answer "what would happen if " questions (Rothenberg 1986, Halley 1987). Our research interests lie in expanding simulations so that they are able to answer questions about goals. A goal directed simulation is given a desired goal and produces the parameters necessary to satisfy that goal.

Goal directed simulation is both pragmatic and theoretical. Pragmatically, many previous simulations have been rejected by operational users. The simulations, supposedly useful for enhancing decision making, went unused because they were too difficult to modify, needed an intermediary to run the simulation, and provided answers that had to be translated into operational terms. Goal directed simulation would allow the operator to enter his goal and would prescribe a recommended way to achieve it, eliminating much time consuming analysis.

Theoretically, goal directed simulation is an area to study the relationship between "predictive" simulation and "prescriptive" knowledge based decision systems (Reddy 1987).

We have built three goal directed simulation environments : the CM ADVISOR , SONAR PLEXUS, and the QUEUEING LAB. CM ADVISOR and SONAR PLEXUS are command and control object oriented simulations , while the QUEUEING LAB is a workflow simulation system. The CM ADVISOR has been installed in an operational facility and provides daily resource allocation recommendations (Halley, 1987). SONAR PLEXUS is an experimental simulation system for underwater surveillance systems which enhances a conventional object oriented simulation with reasoning (Halley et al. 1987). Operators enter system surveillance goals and SONAR PLEXUS produces recommended resource allocations using simulation and reasoning.

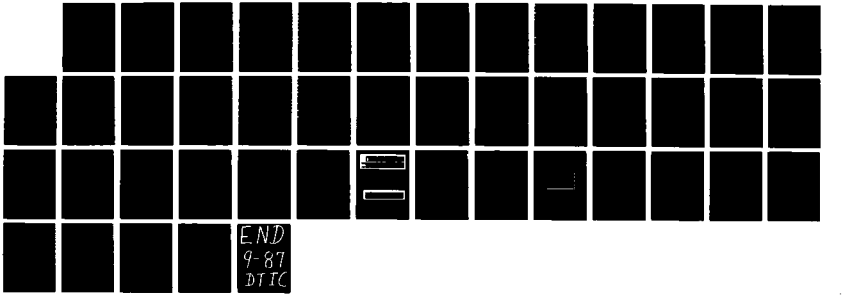
The QUEUEING LAB is a system for simulating workflow problems which are common in computer communications and factory design (Pliske 1987). Artificial intelligence techniques of direct manipulation interfaces and knowledge representation are used to explain and represent the system model. QUEUEING LAB, however, uses queueing theory to calculate system statistics rather than discrete event simulation. Research is now underway to turn this conventional simulation system into a goal directed simulation. Instead of entering changes in transaction arrival rates or station service times to answer questions about the system, the operator will enter a system production goal and the simulation will determine the operating parameters necessary to reach the goal.

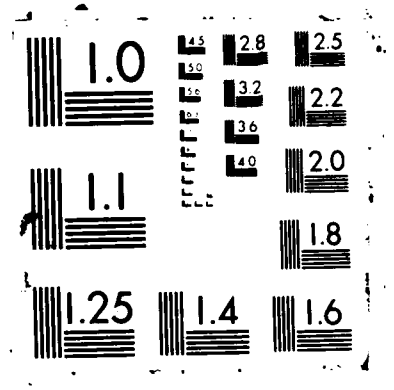
The next two sections will give brief overviews of goal directed simulation as developed in SONAR PLEXUS and as researched in the QUEUEING LAB.

AD-A183 736 WORKSHOP ON AI (ARTIFICIAL INTELLIGENCE) AND SIMULATION (2ND) HELD IN SEA..(U) AMERICAN ASSOCIATION FOR ARTIFICIAL INTELLIGENCE MENLO PARK C..

2\2

UNCLASSIFIED P A FISHWICK ET AL. JUL 87 F/G 12/9 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

SONAR PLEXUS: A Goal Directed Command and Control Simulation

We have developed a simulation environment, called SONAR PLEXUS, whose operational goal is to solve a real world command and control problem and whose research goal is to extend simulation technology. Specifically, SONAR PLEXUS extends the "what would happen if" simulation paradigm to answer 3 questions:

1. What should be done to accomplish a goal?
2. What states cannot exist?
3. What alternatives might improve the outcome?

SONAR PLEXUS simulates the command and control situation in a hypothetical ocean surveillance scenario which was developed from information from the open literature (Tucker 1985). A human operator is in charge of configuring the network of hydrophone arrays in the North Atlantic Ocean so that the hydrophone arrays are collecting signatures from the highest priority ships in the area. SONAR PLEXUS simulates the collection environment and provides the operator with configuration change recommendations.

SONAR PLEXUS includes four basic modules plus specialized interfaces for each module (Figure 1). The environment simulation holds and displays the current status of all resources, connections, and ships under surveillance. The resource model has an explic.. description of the characteristics and constraints of the resources. The reasoner contains rules and procedures for reconfiguration decision making. Finally, the ship database has a large set of ships and characteristics with a gatekeeper which services database queries.

The surveillance system is modeled as a combination of a frame hierarchy and a database. The frame hierarchy

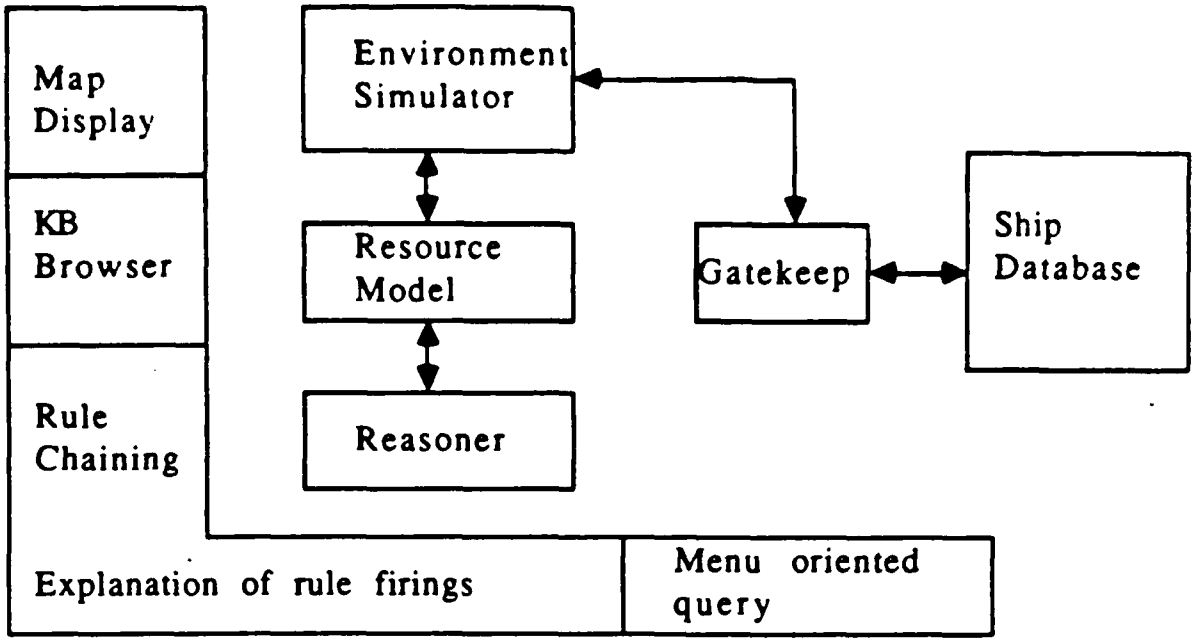


Figure 1. The Sonar Plexus environment consists of a simulator, a resource model, a reasoner, and a ship database with an access gatekeeper. A separate interface was designed to reflect the functionality of each module.

contains an explicit description of the surveillance resources and the network of connections between them. Each hydrophone array, relay site, processing station and processing channel is represented as a frame in a taxonomy of resources.
Each frame's attributes include the capabilities and constaints of the resources, the linkages to other resources, and the current configuration. The characteristics of the ships to be monitored by the surveillance resources are modeled in a standard record database.

Sonar Plexus' reasoning module can determine alternatives and recommendations on how to reach collection goals. This determination is based on the hydrogeography of the area between the ship and the resource, the current network configuration, and the priority of the ships already in the area. The solution proposed by each resource , calculated by a combination of rules and procedures, may range from a retuning of the array to a reconfiguration of the network.

The QUEUEING LAB - Goal directed simulation of workflows

The QUEUEING LAB is a queueing system model developed on a Xerox 1109 lisp machine which combines artificial intelligence and queueing theory to produce an interactive queueing model environment for an analyst designing factories or computer communications networks. The interface is similar to other AI based simulation systems in that objects are directly represented and manipulated on the screen and the underlying model is contained in a frame taxonomy (Figure 2) . QUEUEING LAB differs, however, in that the calculations of workflows and statistics are determined by a closed system queueing model rather than discrete event simulation. In design analysis, order of magnitude results are acceptable when design parameters can only be guessed. In this situation, queueing theory offers tremendous advantages in speed but slightly less model detail than offered by a full simulation language. QUEUEING LAB has been used to identify bottlenecks and design flaws in the initial designs

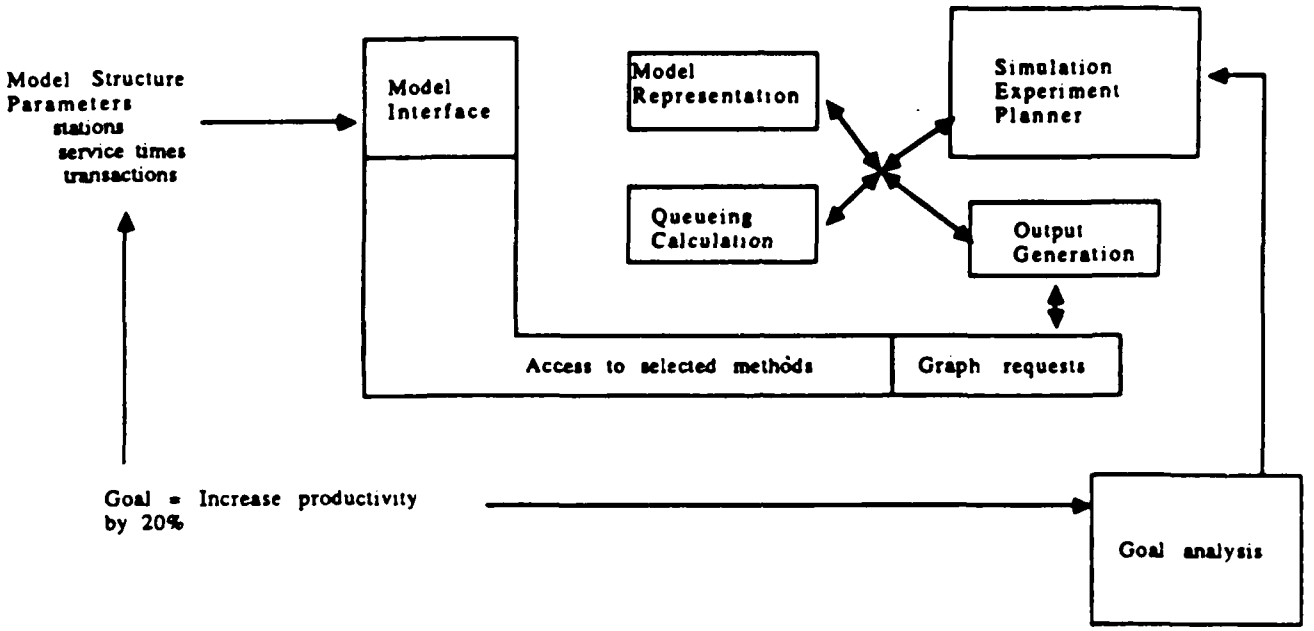


Figure 2. Queueing Lab is a goal directed environment for analysis of workflows. Models are entered using the mouse on a Xerox 1109 lisp machine. Workflow models are stored in a semantic networks which contain the objects and their interconnections. Queueing calculations are performed to obtain workflow statistics. Two research modules are now being added which will allow the operator to enter a goal and have the simulation analyze and plan the way to achieve the goal.

has been used to identify bottlenecks and design flaws in the initial designs for a factory of the future project at TASC (Pliske, 1987).

Research has just been initiated to enhance QUEUEING LAB into a goal directed simulation. We are investigating the possibility of developing a planning module that would be capable of accepting a goal specification consisting of a set of constraints or performance requirements. Based on this specification, and a problem specification consisting of a generalized description of the system in question, the simulation would generate a core model at an appropriate level of detail. Using the core model, along with goal information, the simulation would develop a plan for achieving the stated goal. The plan would entail the construction of a search space of parameterized models, each model being a plausible variation on the core model. Given this plan, the simulator would proceed to execute selected variants in the search space, and finally recommend a variant that satisfies the goal specification.

Currently, the research is focused on the most appropriate way to structure the plans generated by the simulator. Even given a simple system, there are several general strategies one could take in specifying its parameters to achieve a particular goal. At the same time, interactions between elements of the problem frequently exist, and must be dealt with. Finally, the plan must contain the detailed knowledge needed for execution.

REFERENCES

Fox, M. "The Intelligent Management System: An Overview", in Processes and Tools for Decision Support, ed. H.G.Sol Amsterdam, The Netherlands: North Holland.

Halley, M. 1987. "User Reactions to Artificial Intelligence Paradigms", TASC Internal Report, McLean, Va.

Halley, M., Miller, T., Hougum, C. and W. Mosenthal. 1987. "Sonar Plexus - Enhancing a Command and Control Simulation with Reasoning". 1987 SCS Eastern Simulation Symposium, Orlando, Fla.

Pliske, D. 1987. "Using Queueing Theory and Artificial Intelligence to Resolve Design Issues", TASC Internal Report, McLean, VA.

Reddy, R. 1987. "The Epistemology of Knowledge Based Simulation", Simulation 48:162-170.

Rothenberg, J. 1986. ""Knowledge-Based Simulation". AAAI Workshop on AI and Simulation, Philadelphia, Pa.

Tucker, J. 1985. "Cold War in the Ocean Depths", High Technology, Vol 5, No. 7.

Parallel Marker Propagation System Construction Set

Howard Schneider
Department of Psychiatry
Cité de la Santé de Laval
Laval, Québec H7M 3L9 CANADA

ABSTRACT

Since described by Quillian (1968) parallel marker propagation systems, also known as semantic networks, have interested AI researchers. The parallel marker propagation system allows knowledge to be represented in such a way that a database can be quickly searched for items with desired qualities regardless of the size of the database.

Many different types of parallel marker propagation systems are possible. For example, what constitutes an intersection? Should nodes pass on all markers? How should the links propagate markers? Since the hardware to build systems of large numbers of interconnected nodes is not readily available much of the design and analysis of parallel marker propagation systems has been done by intuition, ie, using those features that seemed to do the job. However, over the last few years formalizations of these systems have been developed and it has been shown that intuition is not always correct, eg, Etherington and Reiter (1983), Touretzky (1986).

Another approach towards the development of parallel marker propagation systems is simulation. The experimentation allowed by the simulation of various types of parallel marker propagation systems is useful in both the development of new classes of such systems as well as in assessing the validity of various theoretical approaches. While previous parallel marker propagation system simulators have tended to simulate only very specific systems, eg, Fahlman's (1979) LORIS simulator, or a narrow range of systems, eg, Levenick 's (1985) NAPS simulator, a simulator I have been working on, the Parallel Marker Propagation System Construction Set or PMPSCS, is designed to allow simulation of a large variety of marker propagation systems.

The key to building a simulator capable of simulating a wide range of parallel marker propagation systems is the adoption of a suitable classification scheme for such systems. The classification scheme used by the PMPSCS specifies parallel marker propagation systems by node type, link type and external controller type. A low-level functional, almost structural, basis is chosen for the classification of various nodes, links and external controllers. Some examples of these are shown in Figure 1.

The PMPSCS (Parallel Marker Propagation System Construction Set) is given the specification for a desired parallel marker propagation system. From this information the selected propagation system is constructed and the PMPSCS then awaits input data. Input entries cause simulated nodes to become activated and cause simulated markers to be propagated from one node to another. A sample run from a simulation of a parallel marker propagation system consisting of the node architecture shown in Figure 2, simple passive links and a general type of external controller with several predefined 'behavior' rules is shown in Figure 3.

The PMPSCS is coded in ADA. A library of ADA packages represents different components which in varying combinations can be used to construct a specific type of node, link or external controller. By filling in certain parameters in a given ADA package, various sub-types of the desired node, link or controller are created. The PMPSCS essentially translates the specifications for a given parallel marker propagation system into a combination of ADA packages.

The two limiting factors in performing simulations of parallel marker propagation systems have been memory and time. In almost all simulations performed the number of links tends to be much greater than the number of nodes so that storage requirements and processing time can be expressed in terms of the total number of links. For simulations involving less than one million nodes approximately four bytes of memory are required per link. It is interesting to note that the simulation time is less than linear with respect to the number of links in the system. Apparently as there occur more links and nodes in the system, the probability that a given link will not be used during a set of marker propagation cycles increases.

References

Etherington, D.W. & Reiter, R. 1983. "On Inheritance Hierarchies with Exceptions." Proc. AAAI-83, Pp. 104-108. Available from Morgan Kaufmann, Los Altos, CA.

Fahlman, S.E. 1979. NETL: A System for Representing and Using Real World Knowledge. Cambridge, MA: MIT Press.

Levenick, J. 1985. Knowledge Representation and Intelligent Systems: From Semantic Networks to Cognitive Maps, doctoral dissertation, University of Michigan.

Quillian, M.R. 1968. "Semantic Memory. In Minsky, M. (ed.) Semantic Information Processing, Cambridge, MA: MIT Press. Pp. 227-270.

Touretzky, D.S. 1986. The Mathematics of Inheritance Systems, Los Altos, CA: Morgan Kaufmann.

Figure 1 - Classification of Parallel Marker Propagation Systems

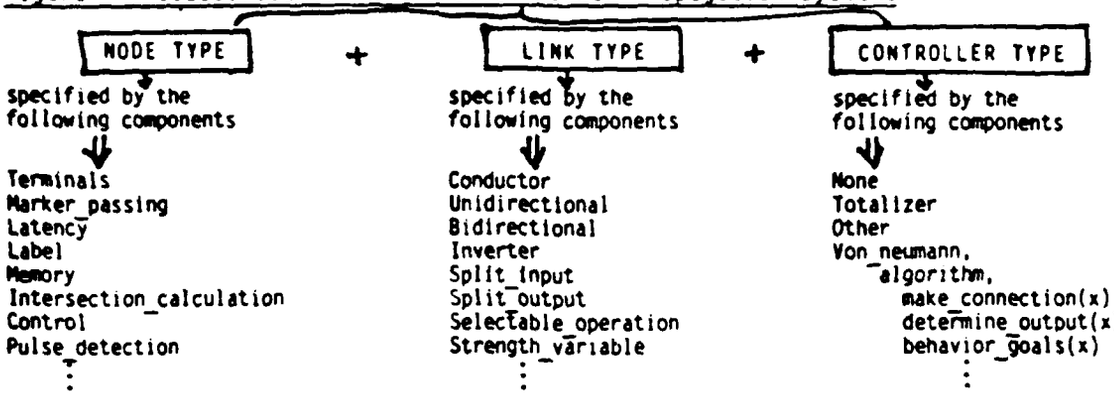


Figure 2 - Specification of a Node Type

```
Terminals(IN/OUT(simple))
Marker_passing(simple)
Latency(out(1))
Label(none)
Memory(permanent(one_write))
Intersection_calculation(
 (total(external))  OR
 (calculated)            )
Control(external)
```

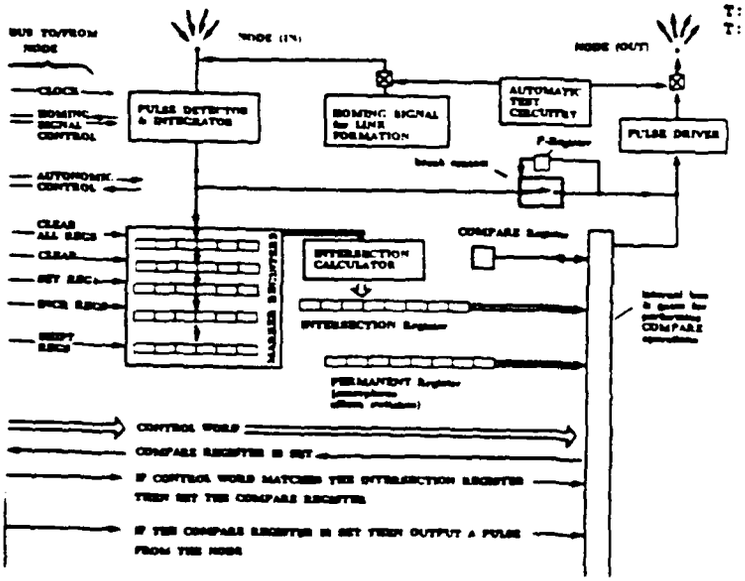


Figure 3 - PMPSCS Simulation Run

```
Node type: as per Fig. 2
Link type: Conductor (markers pass unchanged)
Controller type: Von_neumann( (..) )

T: teacher      C: computer
```

T: HELLO	C: HELLO
T: GOOD	C: noted
.	.
T: RED EAT ROUND	C: APPLE
T: GOOD	C: noted
T: BUY	C: TREE
T: BAD	C: SUPERMARKET
T: VERYGOOD	C: noted
T: WALK_SUPERMARKET	C: linked
T: RESTAURANT_STEAK	C: linked
T: CAR_RESTAURANT	C: linked
T: GASOLINE	C: CAR
.	.
T: HUNGRY CAR	C: STEAK
T: BROKEN CAR	C: GARAGE
T: HUNGRY	C: APPLE

[59]

Simulations in a model-world based on an algebraic approach*

Charles Chiu

Department of Physics and Artificial Intelligence Laboratory

University of Texas 78712

April 28, 1987

[Submitted to the second workshop on AI and simulation.]

Keywords: AI-based simulation tools, abstractions at qualitative and quantitative levels, expert reasoning in simulation, automatic analysis of simulation results.

ABSTRACT: A useful way to demonstrate the working of a set of physics principles is to incorporate them into a model-world and to simulate processes in this world. In a previous work we considered an algebraic approach to simulate processes, such as collisions, recoils and explosions in a world governed by the principle of conservation of momentum [Chiu 87a, 87b]. Presently we investigate the world of constraint motion to further develope our approach. Examples here are motions along a roller coaster track, motions of an object attached to a string etc. For details of present work see [Chiu 87c].

Various components in our framework and their interrelationships are depicted in Figure 1.

1. **Define-world:** It states basic principles of the model-world. For present case they are,

   - the conservation of energy,
   - the presence of centrifugal acceleration in any curved motion,
   - the presence of a toward acceleration pressing against the path, which ensures that the motion stays on the constraint path.

*This research was supported in part by the National Science Foundation through grants DCR 8417934 and DCR-8512779

2. **Define-model:** It contains those dynamical and geometric statements specific to the simulation system. A typical roller coaster model is shown in Figure 2.

   These principles and statements of the model are expressed in terms of simple algebraic relations. These relations are directly passed onto the Math Handler, which interfaces the simulation system with a Math Package.

3. **A simulation reasoner:** It controls simulations and justifications.

   In constrast to the conventional numerical simulation and the qualitative envisionment as in [de Kleer 77], we do an **algebra-based simulation**. For the roller coaster model,

   - the reasoner first instantiates the algebraic expressions of velocity and the toward acceleration in a given segment of the track.
   - It then examines the instantiated expressions based on the properties of the trig-functions and inequality relations and deduces the appropriate motion for the next step.
   - The reasoner provides a **qualitative** description of the simulation. See Figure 3.
   - Upon request, the reasoner will also furnish an explanation to justify the simulated behavior at any point. This justification shows that the local **quantitative** behavior is precisely what one deduces from basic principles.

4. **Analytic reasoners:** They facilitate abstractions of the next level results.

   An example would be the determination of the range of the initial height in S1, so that when a ball begins from this range, it will travel through the entire track. One could determine this by trial and error, i.e. making many runs. However, a more powerful approach is through algebraic consideration. Present tools for the analytic reasoners are:

   - The algebraic solver: It separates out intermediate variables from unknowns, and with the aid of Math handler it proceeds to solve for the unknowns.
   - The critical point locator: It determines parameter values which lead to those features specified by the user.
   - The monotonic function verifier. For instance this tool makes inspection on some interval in order to determine whether the corresponding end points are extremum points.

Some typical questions for the present analytic reasoners are shown in Figure 4.

References

1. C. Chiu. 1987a. Qualitative physics based on exact physical principle (accepted by AAAI Qualitative Physics Workshop, May 1987).

2. C. Chiu. 1987b. Algebra-based construction of a model-world where momentum is conserved. University of Texas Computer Science TR-xx, forthcoming.

3. C. Chiu. 1987c. Algebra-based simulations in a model world of constraint motion. University of Texas Computer Science TR-xx, forthcoming.

4. J. de Kleer. Multiple representations of knowledge in a mechanics problem solver. IJCAI 1977, p. 299.

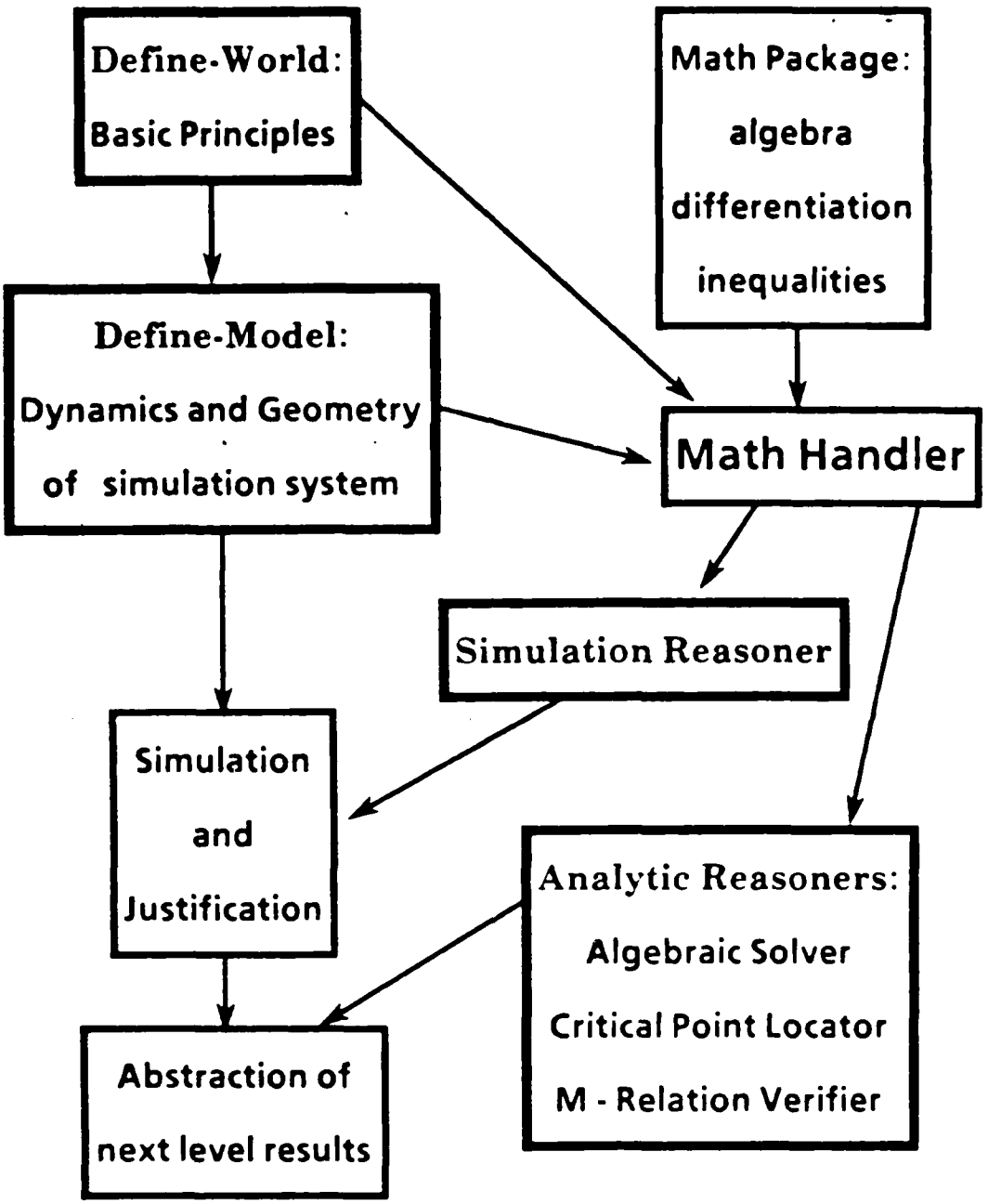


Figure 1: **Algebra based simulation:** Various components and their interrelationships

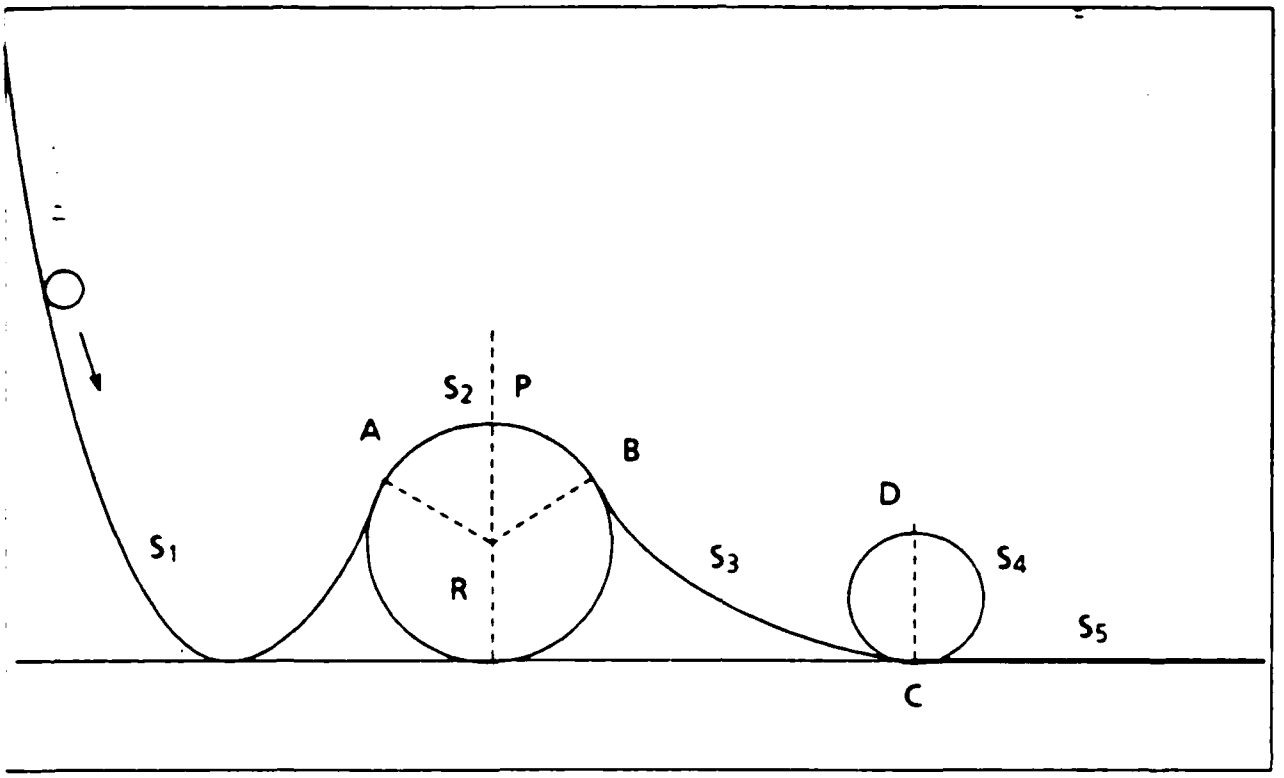


Figure 2. A typical roller coaster model.

Specifications of the roller coaster track for the sample runs given:

- The junction between segments S1 and S2 is at A, where the vertical coordinate $y = 1.8R$
- Segment S2 is a circular arc having a radius R. The peak of S2 is at P, where $y=2R$
- The junction between S2 and S3 is at B, where $y = 1.8R$.
- S4 is a circular loop with a radius R/2.
- S5 is a horizontal segment

```
Q>  (outcome? (stimes 3 R) S1)
Begin in S1 from rest moving to right, initial height
3•R
Pass through path  S1
Fly away at start of  S2  ••••••End of run
Q>  (outcome? (stimes 2 4 R) S1)
Begin in S1 from rest moving to right, initial height
2 4•R
Pass through path  S1
Fly away at start of  S2  ••••••End of run
Q>  (outcome? (stimes 2 2 R) S1)
Begin in S1 from rest moving to right, initial height
2 2•R
Pass through path  S1
Pass through path  S2
Pass through path  S3
Pass through path  S4
Pass through path  S5  ••••••End-of-Track••••••
Q>  (outcome? (stimes 1 8 R) S1)
Begin in S1 from rest moving to right, initial height
1 8•R
Pass through path  S1
Slide back in  S2 ••••••End-of-run
Q>  (outcome? (stimes 1 5 R) S1)
Begin in S1 from rest moving to right, initial height
1 5•R
Slide back in  S1 ••••••End-of-run
Q>  (outcome? (stimes 0 3 R) S3)
Begin in S3 from rest moving to right, initial height
0 3•R
Pass through path  S3
Slide back in  S4 ••••••End-of-run
Q>  (outcome? (stimes 1 R) S3)
Begin in S3 from rest moving to right, initial height
1•R
Pass through path  S3
Fall off in  S4  ••••••End-of-run
```

Figure 3: Sample simulation runs. Syntax: Q> (Outcome? initial-height path-name).

A list of question categories and sample questions:

1. The *What – is?* category:

   - What is the initial height in S1 if the fall-off occurs at $y = 3R/4$ in S4?
   - What is the toward acceleration when the ball begins to slide back at $y = R/3$ in S4?

2. The monotonic function category:

   - Verify a negative monotonic function relation between the initial height in S1 and the toward acceleration at the junction between S1 and S2.
   - Verify a positive monotonic function relation between the toward acceleration and the height in S2 for a given initial height in S1.

3. The critical point category:

   - Find the range of the initial height in S1, such that the ball will travel through the entire track.
   - Find the range of the initial height in S3, such that the ball will fall off in S4 region before it reaches the height $y = 3R/4$.

Figure 4: Typical questions for the analytic reasoners.

[60]

Title: Dynamic Planning Under Uncertainty Using Automated Model Construction and Risk Analysis

Authors: Louis Anthony Cox, Jr., Ph.D. and Richard Blumenthal

Affiliation: US WEST Advanced Technologies

Address: 6200 S. Quebec St. Englewood, CO 80111, Suite 170

EXTENDED ABSTRACT

1. Introduction and Problem Statement

This paper describes the results of research on automatic model building and analysis being carried out at US WEST Advanced Technologies as part of a larger effort aimed at integrating A.I. and Operations Research methods for solving reliability, optimization, and planning problems. An important variety of applications, ranging from on-line management of research and development (R&D) projects to intrusion by an intelligent attacker into a secure data processing system or facility, can be formulated as stochastic logic-network optimization problems where an intelligently planning agent seeks to reach a goal while learning along the way whether he is able to successfully complete certain activities that can help him reach it. Formally, a set of nodes, representing states, activities, or events, is partially ordered by logical precedence constraints, representable (possibly after the introduction of extra nodes) as labelled arcs between nodes. Each node has associated with it both a vector of expected resource costs (e.g., time and money) for attempting to resolve it -- expected costs that may depend on whether the attempt is successful -- and a conditional probability of being successfully completed, given that it has been reached. In the resulting probabilistic AND-OR graph, the planner's goal is to reach and complete a

goal activity (or to discover that it can not be reached) at minimum expected total cost. Within the context of this optimization problem, the goal of an advisory system is to dynamically identify optimal strategies for proceeding as uncertainties are resolved, and to assess when to give up.

2. Solving the Problem: A Hybrid A.I./Simulation Optimization Program

We will describe a methodology that has been developed to automatically formulate and solve stochastic logic-network optimization problems by combining techniques of knowledge representation, A.I. problem solving via network abstraction (using a modular decomposition heuristic) and conventional Monte Carlo simulation. An early version of this methodology has been applied to security risk assessment in the Air Force. Substantially enhanced versions for reliability analysis of complex engineering systems (e.g., communication networks, industrial facilities) have been rapidly prototyped and are now being developed further at US WEST Advanced Technologies.

The program uses artificial intelligence in two places: to automatically model a system or situation in terms of a stochastic logic-network (along with an associated fact-base of instantiated predicates), based on a dialogue with the user; and (ii) to simulate the choice behavior of a "rational" planning agent trying to dynamically solve his planning problem over time. This problem-solving routine is then embedded within a Monte-Carlo simulation to generate a "risk profile" (cumulative probability distribution function) of the remaining resource expenditures required to reach the goal (or to show that it is unreachable). The

decision whether to continue is based on this risk profile. The (heuristically) "best" next node to try, if a decision to continue is made, is automatically identified by an efficient heuristic path-search algorithm. The agent then tries the prescribed node, discovering whether he succeeds or fails at it. This information is fed back into the program, which dynamically "replans" in response to it. It turns out that no replanning is necessary until a failure is encountered at some node, since solving the problem of which node to try next automatically generates a partial strategy (a "best path") for reaching the goal node that should be followed as long as possible.

We will discuss (i) the knowledge-acquisition and representation program (US WEST's proprietary "KRIMB" package) used to formulate stochastic logic-network models by intelligent questioning of the user; (ii) the design and implementation of the simulation portion of the program; (iii) the problem-solving subroutine for intelligent planning in probabilistic AND-OR graphs; and (iv) some example risk profiles generated by the intelligent simulation program for some example Air Force facilities, where it has been used to study the sensitivity of facility vulnerability to physical intrusion as a function of the presence of access controls, the frequency of patrols, etc.

Key References

Alexander, J.H., et al, "Knowledge Level Engineering: Ontological Analysis," Proceedings, AAAI-86, 963-968.

Alterman, R., "An Adaptive Planner," Proceedings, AAAI-86, Vol. 1, 65-69.

Ben-Dov, Y., "Optimal Testing Procedures for Special Structures of Coherent Systems," Management Science, 27, 12, 1981, 1410-1420.

Brachman, R., and Schmolze, J.G., "An Overview of the KL-ONE Knowledge Representation System," Cognitive Science, 9, 2, 1985, 171-216.

Cox, L.A., "A Probabilistic Risk Analysis Program for Analyzing Security Risks," Proceedings of the 1986 Society of Risk Analysis Annual Meeting, Plenum Press, 1987 (forthcoming.)

Cox, L.A., "Intruder Risk Assessment and R&D Planning in Stochastic Logic Networks," invited paper presented at Joint ORSA/TIMS Meeting, Miami Beach Fontainebleau Hilton, October 27-29, 1986. (Currently under review by Operations Research.)

Cox, L.A., ATAM Knowledge Base Methodology, Final Report to the Department of Transportation, Arthur D. Little, Inc., Cambridge, MA, December, 1986.

Kadane, J.B., and Simon, H.A., "Optimal Strategies for a Class of Constrained Sequential Problems," Annals of Statistics, 8, 2, 1977, 237-255.

Kaczmarek, T. et al, "Recent Developments in NIKL," Proceedings, AAAI-86, 978-985.

Sidney, J.B., and G. Steiner, "Optimal Sequencing by Modular Decomposition: Polynomial Algorithms," Operations Research, 34, 4, 1986, 606-612.

Simon, H.A., and J.B. Kadane, "Optimal Problem-Solving Search: All-or-None Solutions," Artificial Intelligence, 6, 1975, 235-247.

[61]

FAULT DIAGNOSIS BASED ON QUANTITATIVE MODELS

Stefan Feyock

Department of Computer Science
College of William and Mary
Williamsburg, VA 23185

Reasoning about physical systems from first principles, whether for purposes of explanation, prediction, or fault diagnosis, requires the reasoner to have available a model of the subject system; the model, in a sense, *is* the first principles.

The motivation for the research to be described was the requirement on the part of workers at NASA/Langley Research Center for model-based fault diagnosis techniques that could deal with physical systems such as jet engines and avionics subsystems. Such systems have a predominantly continuous character, and are thus best represented by continuous models; since they are usually too complex to be mathematically tractable, continuous simulation models (CSMs) are the representational vehicle of choice. The present paper describes the use of CSMs as the basis of an automated reasoning system, and in particular the application of such a reasoner to fault diagnosis. The justification for using a quantitative rather than a qualitative model lies in the fact that the quantitative description of the system of interest allows an innovative application of a technique familiar from the field of software engineering and program proving: the use of Dijkstra's concept of predicate transformers [Dijkstra] to establish weakest preconditions for the model to exhibit the behavior seen in the actual system.

The basis of our approach to model-based fault diagnosis is to observe the actual system in action, and then pose the question "how do we make the model behave like that?" It was noted that in most cases faults corresponded to unexpected changes in CSM model parameters. Thus, given a set of actual system behaviors, we ask what parameter values have to hold in the model to produce the observed results.

This formulation of the problem provides the link to the weakest precondition approach, which poses a similar question about programs: given any predicate, the program acts as a predicate transformer that maps the given predicate to the (weakest) predicate that had to be true before the program was executed, in order for the given predicate to hold after execution.

A number of problems arise as a result of attempting to apply predicate transformers to CSMs in the manner indicated. Simulation models are unlike non-simulation programs in that the latter engage in runtime behavior to produce a result; in a simulation, the runtime behavior *is* the result. A number of modifications to the weakest precondition approach were required to produce appropriate results.

The techniques described have been applied to a number of systems, including an Aerobee rocket control system, a relay servo, and an aircraft arresting cable mechanism; CSMs describing these systems were drawn from [Chu]. The models furnished to the reasoner are thus not based on if-then rules or any other sort of expert system or qualitative model, but rather are quantitative CSMs of the actual system. This fact has a number of interesting implications, including the integration of qualitative and quantitative simulation, and the extraction of information not only from the simulation output, but from the structure of the simulation model itself.

Chu, Yaohan, *Digital Simulation of Continuous Systems*, Mcgraw-Hill, New york, 1967.

Dijkstra, J. E., *A Discipline of Programming* Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[62]

Simulation of Chemical Processes in Preliminary Design: Model-Construction, Abstraction, and Control of Information-Flow

Submitted to The Second Workshop on AI and Simulation

by
Theodore Kritikos
Michael L. Mavrovouniotis
and **George Stephanopoulos**

Laboratory for Intelligent Systems in Process Engineering
M.I.T. Room 66-562
Department of Chemical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

28 April 1987

Simulation of Chemical Processes in Preliminary Design: Model-Construction, Abstraction, and Control of Information-Flow

by Theodore Kritikos, Michael L. Mavrovouniotis, and George Stephanopoulos

1. Preliminary Design

In the design of chemical processing systems, the designer must make a preliminary judgement on the profitability of a proposed process, based on some initial flowsheet. This flowsheet, which may be only partial or inconsistent, must be developed very quickly, and with very limited available data. Thus, the simulation needs during this phase are quite distinct from those at the detailed design stage. Extra flexibility and versatility in the models and the simulation methods is essential.

The Design-Kit graphic interface [2] (Figure 1) was developed on Symbolics 3640 and 3650 Lisp computers, and allows the construction and manipulation of process flowsheets, construction of flowsheet models, and preliminary steady-state simulation with simplified models. All these operation can be carried out through easy-to-use menus of different types. The facilities include Newton-Raphson equation solving, symbolic differentiation, and degrees-of-freedom analysis

In this work we discuss issues of design-oriented simulation and specific solutions we have developed within the Design-Kit framework.

2. Automatic Modeling of Equipment

The equipment units that can be used in the design are classified in a hierarchy. Units are classified based on characteristics such as input-output topology (e.g. single-input-double-output), internal homogeneity (e.g. homogeneous-composition equipment are well mixed internally), heat behavior (e.g. isothermal, adiabatic, or externally heated), pressure behavior (e.g. isobaric), etc.

To create a new type of unit, the designer may simply state what existing classes are superclasses of the new type. The inheritance mechanisms of the classification can then cooperate with special methods to construct *automatically* a data-model for the equipment unit, such that the implicit specifications imposed by the classification are met.

3. Connection Streams

When the units are connected, special *connection streams* are created. Thus, the input-output ports of two equipment pieces do not communicate directly, but rather through the connection streams. The simulation takes place in a constraint-propagation manner [1], with information passed from unit to unit via the connection streams. The presence of the connection streams caters to two particular needs of preliminary design simulation.

Control of Information Flow. Even though the connection would eventually represent material flow, so that all the properties of the fluid would have to match between the two connected equipment ports, it is often desirable to postpone this stage until the basic structure of the flowsheet has been established. The designer does not want to take into account the heat needed

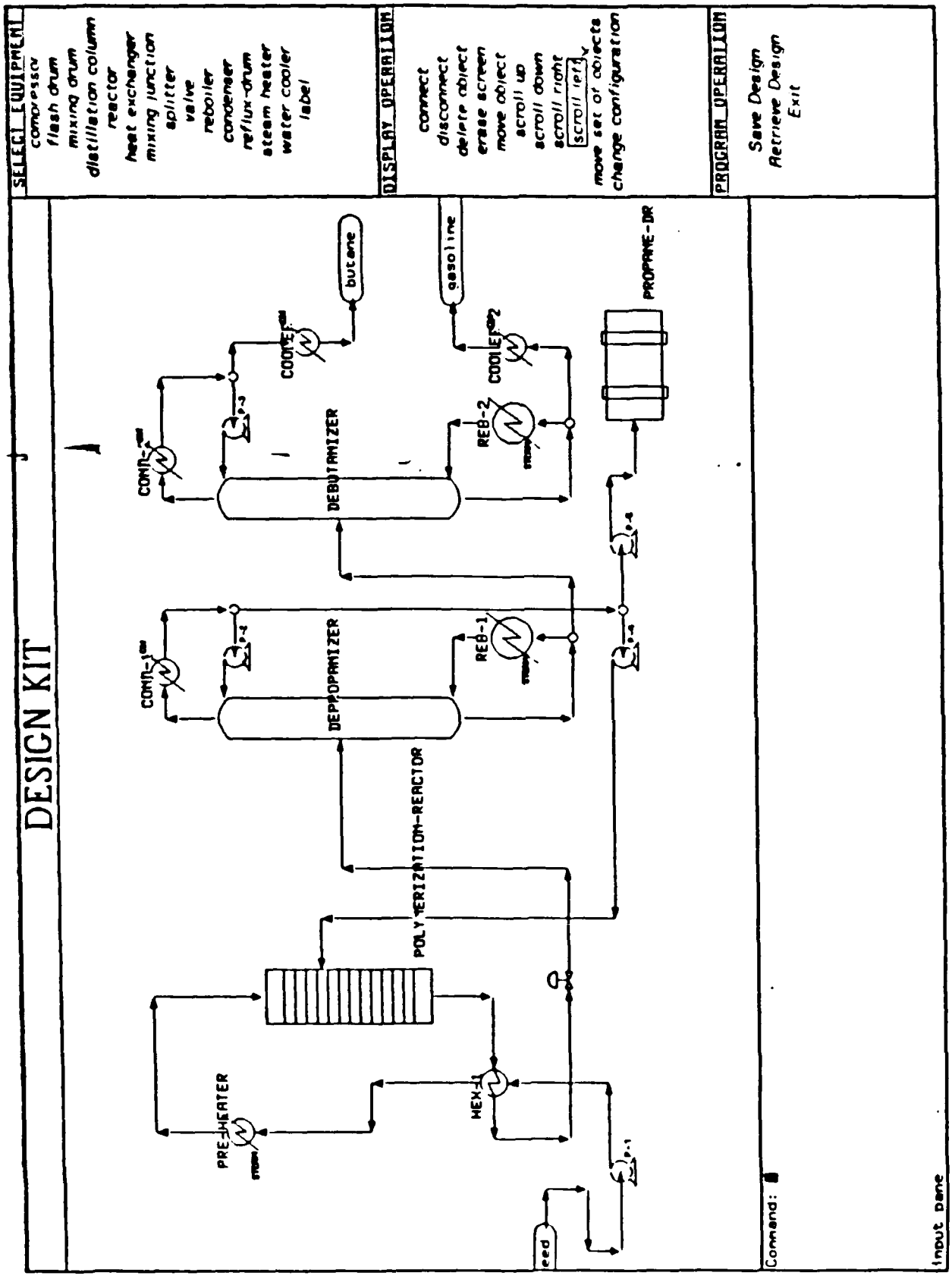


Figure 1: The Design-Kit graphic interface for the Preliminary Design of Process Flowsheets.

to raise the temperature of a stream between units, but rather postpone heat integration until a later detailed-design stage. As the connection is made, the designer simply specifies what kind of information is to be conveyed through the connection, and can thus explicitly control information flow for hierarchical design. For example, to avoid heat considerations, the designer excludes temperature from the "matched" properties, allowing the stream to have different temperatures at each end. The kind of information conveyed through the connection streams can be changed at any time.

Multiple Levels of Abstraction. The designer often views a certain section of an artifact at two different abstraction levels simultaneously [1, 3]. For example a series of distillation columns is viewed at both the detailed level and more abstracted as a "product recovery section" on which specifications may be posed. In order to accomplish simultaneous simulation at different levels of abstraction in our system, the designer need only construct the alternative views and then indicate their correspondence by stating pairs of connection streams as "overlapping", i.e. as being one and the same stream.

4. Strict Modelling Abstractions

Some of the models are too complex for the automatic modelling mechanism to handle because of the complexity and internal structure of the unit. In conjunction with the abstraction mechanism described above, we plan to provide facilities to construct such models from high-level specifications of the conceptual parts of the equipment unit and their interactions.

5. References

[1] Steele, G.L. Jr.
The Definition and Implementation of a Computer Programming Language Based on Constraints
Technical Report AI-TR-595, AI Lab., MIT, Cambridge, Massachusetts, August, 1980.

[2] Stephanopoulos, G., Joback, K., Johnston, J., Lakshmanan, R., Kritikos, T., Mavrovouniotis, M., and Siletti, C.
Design Kit: An Intelligent Interface and Database for Process Engineering
In *1987 Spring National Meeting* American Institute of Chemical Engineers, Houston, March, 1987.

[3] Sussman, G.J., and Steele, G.L. Jr.
CONSTRAINTS- A Language for Expressing Almost-Hierarchical Descriptions
Artificial Intelligence 14 1-39, 1980.

[65]

Simulating the Behavior of Complex Devices for Model-Based Troubleshooting

Walter Hamscher
MIT Artificial Intelligence Laboratory, Rm 834
545 Technology Square
Cambridge, MA 02139
(617)-253-5848
Netmail: hamscher@ht.ai.mit.edu

Troubleshooting programs based on models of structure and function [1,2,3,4] scale poorly. They perform adequately on simple devices without internal state, such as combinational circuits, but complex devices present a serious pragmatic drawback: the troubleshooter must be able to simulate the operation of the device being diagnosed, perhaps several times under different fault scenarios. Moreover, the troubleshooter must record many intermediate results that ordinary simulators discard. The difficulty and cost of making predictions from the device model swamps the troubleshooting task. Yet, ironically, many failures in complex devices have catastrophic effects, manifesting themselves in ways that should be predictable without recourse to detailed simulation at all. Since model-based troubleshooting otherwise has important advantages relative to more traditional approaches to automated diagnosis (e.g. Mycin, Internist [6]), we wish to model complex devices using abstractions that make troubleshooting feasible. In other words, we propose to avoid the drawbacks not by changing the way the troubleshooter works, but rather by modeling the target device with the ultimate goal of doing troubleshooting explicitly in mind. In particular, we wish to make even as complex a device as a computer board appear to behave like the kind of simple static device that the methodology handles best.

The important abstractions to make are those that reduce the number of distinguishable device states under consideration, i.e., that abstract away temporal detail. An example of such an abstraction would be to model only the rates at which events occur, rather than the individual events. This is quite natural for continuous systems; for example, the behavior of an oscillator can be expressed in

terms of a number of cycles per unit time, while it would be impractical to model its behavior cycle-by-cycle, event-by-event. The abstraction also applies to sequences of discrete events such as the transmission of characters over a serial line, where knowing only the aggregate rate may be sufficient for the task at hand. We propose to simulate a computer board using a representation that suppresses most of its components' internal states and instead models behavior in terms of the rates at which events occur. This simulation will then be used for troubleshooting the board.

The intuition that supports the use of such a model for troubleshooting is that humans can often tell a great deal about the possible faults in a misbehaving system merely by considering the rate at which events occur, without concerning themselves with the nature or timing of the individual events. For example, one expects the rate at which characters appear on the screen of a computer terminal to be very close to the rate at which characters are typed on its keyboard. Different distortions of that rate can suggest different kinds of failures - for example, if the output rate is much lower than expected but non-zero, the problem is probably in the host computer and not the terminal itself. For a model-based troubleshooting program to draw the same simple conclusion without undue effort, its behavior model of the host and terminal should be simple, too. The model should only make predictions about the expected rates of events; it should not be necessary to pay the overhead of modeling individual characters being transmitted across the wires. Such an abstract level of simulation will clearly not always be adequate; in particular, certain events may cause a system component to change its internal state in such a way that its subsequent responses are significantly different from before. A character stream sent to a terminal, for example, may contain an escape sequence that puts it into graphics mode. Hence, the simulator and the behavior models for the system must be able to make use of information at different levels of temporal detail and integrate them smoothly, in much the same way that mixed-mode logic simulators must integrate results derived at different levels of abstraction.

1 Brown, J. S., R. R Burton, and J. deKleer. Pedagogical, Natural Language, and Knowledge Engineering Techniques in SOPHIE I, II, and III. in: D. Sleeman and J. S. Brown (eds.), *Intelligent Tutoring Systems*, (Academic Press, New York, 1982) 227-282.

2 deKleer, J. and B. C. Williams. Diagnosing Multiple Faults. *Artificial Intelligence* 32 (1987) 97-130.

3 Davis, R Diagnosis Based on Structure and Function. *Artificial Intelligence* 24 (1984).

4 Hamscher, W. and R. Davis. Issues in Model Based Troubleshooting MIT Artificial Intelligence Laboratory Memo 893, 1987.

5 Szolovits, P. (ed.) Artificial Intelligence in Medicine Westview Press 1982.

Benchmarks for Research in Planning

Thomas Dean[1]
Department of Computer Science
Brown University
Box 1910, Providence, RI 02912

Abstract

The aspirations of researchers in planning have advanced significantly beyond the blocks world domain of the 1970s. The need to build robot planning systems capable of dealing with uncertainty and responding to situations in real-time has prompted a variety of new approaches to planning and control problems [1] [2] [3] [4] [5] [7]. Evaluating the relative merits of these approaches has, however, proved difficult. There are no real benchmarks for planning programs. Each program is tested on a separate robot or robot simulator thus making detailed comparisons impossible. We have set out to correct this deficiency by providing a complete simulated environment for developing and testing planning systems for applications in robotics. Our system, called BRIE[2], provides all of the tools, including a solid modeler and an agenda-based simulator, necessary to simulate a variety of industrial environments. We provide a one complete environment that models an appliance warehouse with one or more automated forklift trucks and various means of specifying the arrival times of incoming orders and deliveries (i.e., trucks to be loaded and unloaded). The important aspects of mechanical devices and their controllers are modeled by a discrete real-time simulator. Global parameters can be adjusted to vary the accuracy of sensors and servo operated machinery. A stereo depth profiler and an infrared range finder are but two of the simulated sensors provided. In the warehouse environment, all mechanical devices are controlled by direct memory access devices, but it is straightforward to simulate programmed I/O type devices and a speed controller is provided in the package as an example of how such devices can be implemented. Each simulated forklift truck is equipped with a variety of computational devices including a special CPU that controls all of the robot hardware and serves as an interface to the sensors. Lisp programs implementing planner routines are handled using standard real-time simulation techniques. We provide examples illustrating how to insert hooks into existing code to control timing and allow planning routines to handle interrupts and suspend and resume computations. The simulator allows for simple

[1]This work was a collaborative effort of the Brown University Planning Group whose members include John Arnold, Ken Basye, Gerry Boetje, Tone Engel, Timothy Good, Moises Lejter, and Scott Meeks

[2]BRIE stands for the Brown Robotics Implementation Environment

linear speedups or slowdowns of planning routines. The preferred method for controlling the simulated hardware involves downloading special programs to the CPU designed for controlling the hardware. A library of useful subroutines is provided for constructing these special programs. The solid modeling routines are used to keep track of the simulated environment, but they can also be used to support rudimentary spatial reasoning and path planning.

Our system provides a flexible environment for developing and testing planning programs that are designed to cope with uncertainty, spatial complexity, and real-time constraints. BRIE is as portable as Common Lisp is. We provide two options for graphical output: a set of routines built on the X-window system and a general scene rendering representation for animation [6]. The system is currently being used for research and educational purposes at Brown and it is our intention to make the code available to the general research community. Within the context of the warehouse domain, we will be developing a number of scenarios involving warehouse event sequences (deliveries and orders) and fixed forklift parameters. It is our hope that these scenarios will serve as benchmarks for researchers. We think that our work represents a viable approach to both assessing and furthering progress in the field.

References

1 Chapman, David, and Agre, Phil , Abstract Reasoning as Emergent from Concrete Activity, in Georgeff, Michael P and Lansky, Amy L (Eds), *The 1986 Workshop on Reasoning about Actions and Plans*, (Morgan-Kaufman 1987).

2 Brooks, Rodney A , A Robust Layered Control System for a Mobile Robot, A I Memo No 864, MIT AI Laboratory, 1985

3 Dean, Thomas, Intractability and Time-Dependent Planning, in Georgeff, Michael P and Lansky, Amy L (Eds), *The 1986 Workshop on Reasoning about Actions and Plans*, (Morgan-Kaufman 1987)

4 Miller, David P , Firby, R James, Dean, Thomas L., Deadlines, Travel Time, and Robot Problem Solving, *Proceedings IJCAI 9, Los Angeles, Ca.*, IJCAI, 1985.

5 Rosenschein Stanley, J and Kaebling, Leslie Pack, The Synthesis of Digital Machines with Provable Epistemic Properties, in Halpern, Joseph Y. (Ed.), *Theoretical Aspects of Reasoning about Knowledge, Proceedings of the 1986 Conference*, (Morgan-Kaufman 1987)

6 Strauss Paul S , *A Tutorial Guide to the SCEFO Language*, Technical Report, Brown University Department of Computer Science, 1986

7 Wilkins, David, Recovering from Execution Errors in Sipe, *Computational Intelligence* 1 (1985) 33-45

{67}

Simulation and Expert Systems for Finding Particle Beam Line Errors*

Lawrence Selig, Scott Clearwater,† Martin Lee,†† Robert Engelmore
Knowledge Systems Laboratory, Stanford University, Stanford, California 94305

Particle accelerators are complex facilities costing up to a billion dollars. A beam transport system of a particle accelerator is a lattice of electromagnets that bend and focus the beam through a vacuum pipe to a fixed target or another beam. Proper operation of the beam line depends on the correct alignment and calibration of these magnet elements. At thousands of dollars per hour just for electrical costs, there is a strong desire to quickly find any errors in the elements.

We have developed a knowledge based system, ABLE[1] [2] , to find errors in a beam line. ABLE couples rule based problem-solving knowledge with numerical simulation and optimization techniques to achieve expert-level performance. On dozens of test cases using simulated problem data, ABLE finds errors as well as an expert accelerator physicist.

Trajectory problems represent a large and important class of problems found in beam lines. Well-understood physical models[3] can predict a beam trajectory through a magnet lattice. ABLE interfaces with a Fortran modelling and optimization program, PLUS[4] , to generate a simulated beam trajectory (monitor readings) for a given set of magnet errors. Using the mouse-oriented, graphics interface (see figure 1), the expert can thus manually find errors by playing with the beam line and comparing the simulated beam trajectory with the reference beam trajectory.

---

* Work performed under the auspices of the U. S. Department of Energy and supported by the U. S. Army Strategic Defense Command, by the Department of Energy under contract number DE-AC-03-76SF00515, and by the Defense Advanced Research Project Agency under contract N00039-86C-0033.

† Los Alamos National Laboratory, Los Alamos, New Mexico USA 87545 and visiting scholar at the Knowledge Systems Lab, Stanford University.

†† Stanford Linear Accelerator Center, Stanford, California USA 94305.

The optimal value of an error can be found by conducting an optimization "experiment". The optimisation variables are the errors for a specific set of magnets and the fit points are the reference monitor readings. The results of the optimisation are numeric and require interpretation by the expert to determine if the hypothesized errors do in fact provide a good model of the problem. Our approach of coupling symbolic reasoning with numerical modelling and optimization may be applicable to processes other than accelerator control.

ABLE automates the expert's error-finding process. It plans and executes optimization experiments, then interprets the results to plan new experiments until the entire beam line section has been analyzed. (Figure 1 shows a solution found by ABLE superimposed on the input problem.) These experiments constitute an experiment tree, (see Figure 2), where the child experiment's monitors and magnets are supersets of those of its parent. ABLE also includes mixed initiative that allows the physicist to modify results, and delete, plan, or modify planned experiments as well as more usual capabilities such as plotting, explanation, etc.

The level of automation offered by ABLE is so far advanced over existing systems that the expert physicist can now rapidly formulate and test new problem solving strategies and augment his expertise. ABLE is helping develop new techniques which improve the state-of-the-art in finding errors in a beam line as well as automating these procedures. ABLE has been used to analyze real beam data and saved a great deal of the expert's time.

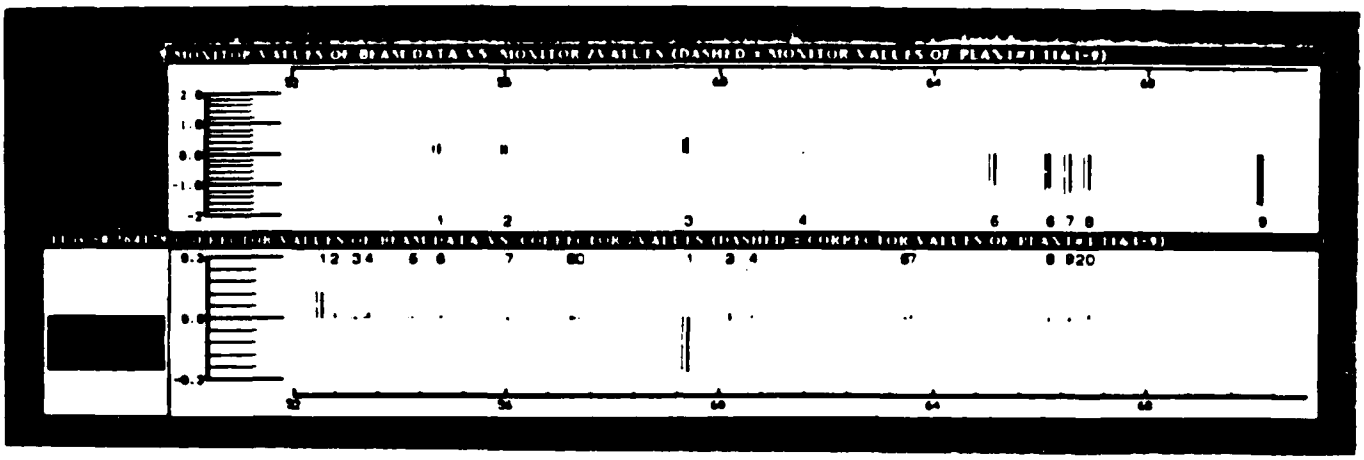


Figure 1 - Using Simulation in ABLE - The actuator bar at left was used to define large errors at elements (correctors) 1 and 11. The simulated monitor readings are plotted above. Also shown superimposed on the plots is the solution (monitor and error values) found by ABLE.

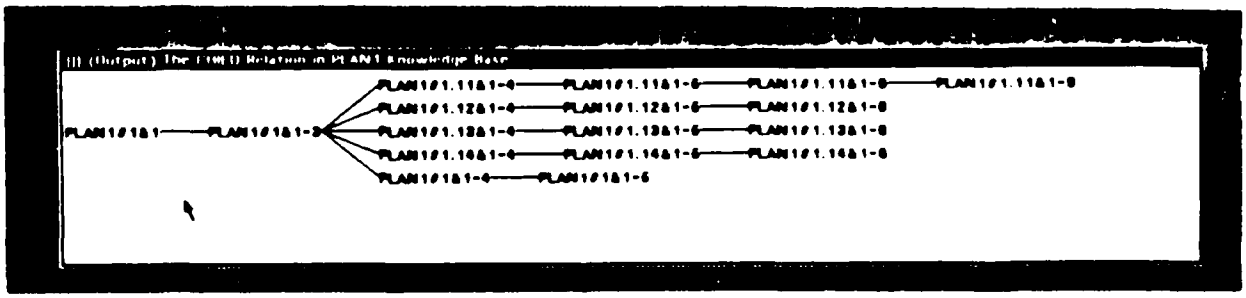


Figure 2 - Experiment Tree for Problem in Figure 1 - Experiments are named by their plan, variable elements and fit monitors. Plan1#1.11&1-9 means that elements 1 and 11 are used to fit monitors 1 throught 9. Plan1#1.11&1-9 is the only solution found for this problem.

REFERENCES

1. S. H. Clearwater and M. J. Lee, "Prototype Development of a Beam Line Expert System," 1987 Particle Accelerator Conference, Washington, DC.

2. M. J. Lee, S. H. Clearwater, S. D. Kleban, and L. J. Selig, "Error-Finding and Error-Correcting Methods for the Start-Up of the SLC," 1987 Particle Accelerator Conference, Washington, DC

3. M. D. Woodley, M. J. Lee, J. Jaeger and A. S. King, "COMFORT, Control Of Machine Functions OR Transport Systems", 1983 Particle Accelerator Conference, Santa Fe, New Mexico.

4. M. Lee, S. Clearwater, E. Theil, V. Paxson, "Modern Approaches to Accelerator Simulation and On-line Control", 1987 Particle Accelerator Conference, Washington, DC.

[69]

Causality in Simulation

Stefan Bernemann
Bernd Hellingrath

Fraunhofer Institut für
Transporttechnik und Warendistribution

Emil-Figge-Str. 75
D-4600 Dortmund50
FRG

Abstract:

In this paper we present some ideas on how and where techniques and methods of artificial intelligence can be used in the field of simulation. Based on the conceptual similarities of expert and simulation systems we describe our view of an integration of simulation and knowledge based systems, where the expressive power of modern representation languages can be used in the modelling process. We describe the features of such an integrated knowledge based simulation environment in the steps of modelling, simulation and analysis.Especially in the area of analysis a representation of causal/temporal relationships seems to be important. We describe two ways how to integrate such relationships into a simulation system.

Introduction

Both expert and simulation systems use model based problem solving. That means that the solution of the problem is not obtained by performing the steps of a predefined algorithm (a controlled - but fixed - sequence of steps or elementary actions). The solution is moreover derived by interpreting a model about the problem domain. However the kind of questions asked to such systems differ: In simulation the next state of the modelled system or the state after n time units is asked for whereas expert systems mainly deal with the task of interpreting a given system state.

Now if we consider to use an expert system to help us in performing a simulation study (How such a help could look like is described later) we see that the models underlying both systems overlap significantly. That's why we want to integrate a simulation and an expert system to work on *one comprehensive model* .

Integration of AI and Simulation

The idea of using a comprehensive model does not help much if we don't know how we want to represent it! We must use some formalism to describe the model so that for all the different questions the answers can be derived from that representation. In building expert systems there shows up a trend to use so called "hybrid systems". These are programming environments where different programming styles (rule based, object or frame oriented, procedural and logic) can

be used in an integrated manner. We feel that these hybrid representation languages are adequate for a modelling language that can be used for simulation.

In the following table we have listed up some formalisms, how they are used in expert systems and how they can be used in such a modelling language. Especially the object oriented approach has a long tradition in simulation; one of the first object oriented languages "Simula" is a language designed and used for simulation. In the literature other approaches have been presented, where Prolog [1] or rules [2] are favored to be used as a Simulation language. But using only one of these formalisms mostly leads to an inefficient realization or to a description

Formalism	Used in XPS	Used in Simulation
rules	heuristic and associative knowledge	complex control
frames/ objects	taxonomies and structural knowledge	objects of the simulation
procedures	knowledge about object behavior, elementary tasks	behavior of the simulation objects
relations (horn clauses)	relations between the objects and facts	relations between simulation objects

Table 1: The use of different formalisms

of the model that is not very readable. A detailed example of how these different formalisms can be used in describing a simple model of a transportation system can be found in [3]. For some of the commercially available representation tools that incorporate the use of different programming styles packages are already available that support the user in using these tools for simulation. (SimKit based on Kee from Intellicorp and Simulation Craft based on Knowledge Craft from Carnegie Group.)

The main novelty of these packages is that the knowledge engineer can use the power of the underlying representation tool to build an expert system that has access to and uses the structures of the simulation system. In the KBS project at the Robotics Institute [4], based on the Language SRL (whose heir is CRL), it was tried to build such an intelligent support. Simulation is an *objective-driven process* . In general the goal is the optimization of one or several system parameter or to detect relations between system parameters. That goal plays an important part in the process of modelling, how the experiments are set up and how to analyze the results.

The following list of some features of an Integrated Knowledge Based Simulation Environment is ordered according to the steps modelling, performing the experiments and analysis:

Modelling

- Modelling language close to expert's way of thinking (e.g. concepts like "throughput" should be part of the language!)
- Design model according to objectives of simulation study
- Storing models, parts of models, and domain knowledge in a data (knowledge) base
- Support the user by giving him guided access to this data (knowledge) base
- Interactive modelling by using an elaborate user interface (e.g. windows, mouse, icons, menus, "intelligent" editors for the modelling language and the graphical representation of the models, help facilities, on-line documentation, ...)
- Static analysis of the model, using heuristics to detect weak points which (might) cause missing the objective. These heuristics are gathered from experts of the problem domain.

Simulation

Designing Experiments

- Propose instrumentation of the model according to the objective of the simulation
- Propose which data to gather during the experiment
- Propose initial parameter values of the model according to experiences of experts and performed experiments

Running Experiments

- Interactive execution of the experiment, giving the possibility to stop the execution, inspect and change parameters of the model or of the experiment
- Visual feedback of important parameter values and evaluated data
- Hiding of *unimportant* data (that means a qualitative interpretation of the data!)
- Notification of *interesting* events relative to objective

Analysis

- Supporting the expert in analyzing the gathered simulation data, to determine whether the objectives are met
- Making suggestions how to change the simulation parameters, to meet objectives

Causal Models

In examining the task of analysis of simulation data one realizes that very often knowledge about causal/temporal relationships between events and simulation objects is required. Experts analyzing simulation data not only use heuristics but also very often a causal model of the simulated system to detect e.g. the reasons for a system breakdown through a causal line of objects and events. Let's make this clear with an example out of our working area:

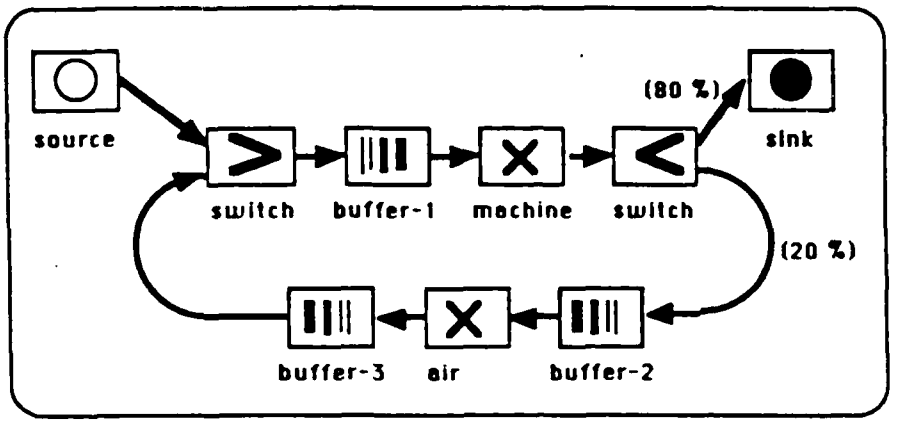


Fig. 1 A simple model

In Fig. 1 a simple model is shown where items enter the system from the source, go through a switch into a buffer, are processed by a single-server machine and leave the system in the sink behind the switch. 20% of the objects however are lead back through a buffered repair cycle to the switch after the source. We monitor the number of objects in the buffer and see in Fig. 2 a sudden rise of the buffer utilization.

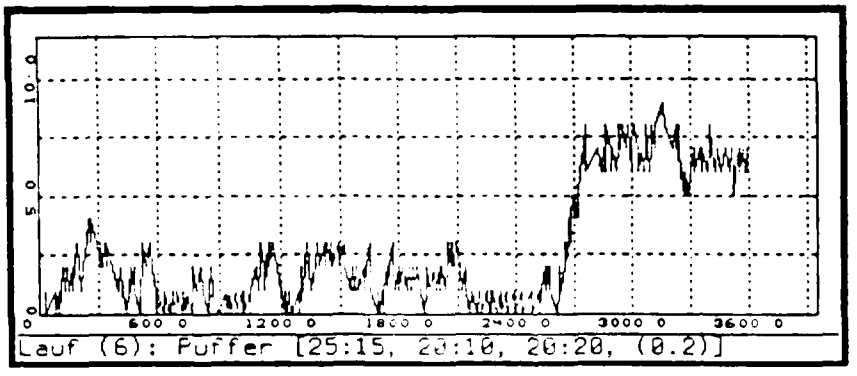


Fig. 2 Curve showing the number of items in buffer-1 over time

Our question is: Why does that happen? Was the cause of inhomogenity the distribution of the source or of the switch behind the machine?

Therefore it seems to be very useful to make causal relationships between objects explicit in the simulation model and to have a 'history' of the events and effects that have happened during the simulation, if the analysis of simulation data should be supported by a knowledge based system.

For a few years now an area of AI research, 'Qualitative Reasoning' is concentrated on developing methods to incorporate causal/temporal knowledge in AI systems to enlarge their area of application and to make them more robust. We have tried to adopt two approaches to get an explicit representation of causal/temporal relationships into a simulation.

Structured Events

In this approach we are using a system which works with an object-oriented description of the simulation system as described above. The general method for executing the simulation is event-scheduling. Normally the scheduling of new events is part of the state transition function, which describes what state changes in the system occur after a specific event. In our system we have tried to make these implicit relations between the events more explicit by tracing them back to state changes of the system. Therefore a set of rules is specified which describe under what conditions a new event will be scheduled. Through this the scheduling of new events can be described more explicitly independent of the state transitions caused by an event. The system takes care for an evaluation of the rules as soon as the state of a simulation component has changed. With such an event scheduling it is possible to record the temporal course of the simulation and the causal relationships on the level of events.

The resulting history network looks like this:

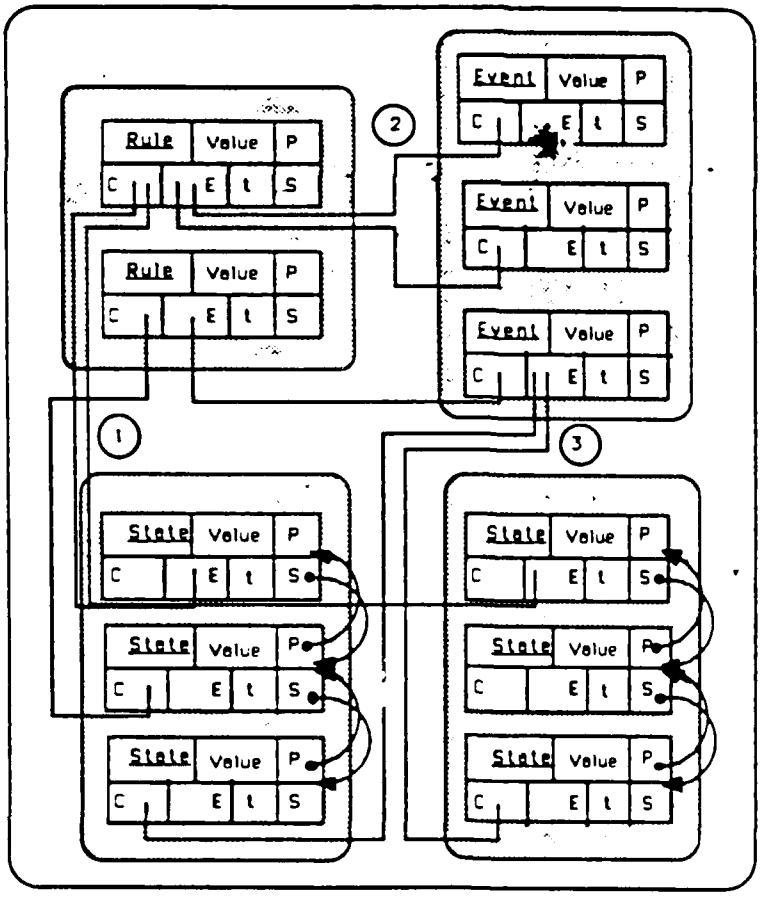


Fig. 3: The causal network

In out network (Fig. 3) we distinguish three types of nodes: state nodes, rule nodes and event nodes. These nodes are connected by two types of links:

Temporal links (Predecessor and Successor) connect rule- and event nodes according to their creation time. A similar list sorted by creation time is built for each state variable in the components and consists of state nodes, which represent the history of state changes at that particular state variable. These temporal links are shown in Fig. 3 by the arrows in the P- and S-fields.

Causal links connect nodes of different types by Cause and Effect relations, which are inverse to each other. These links are built up as follows:

A new state node is created whenever a state variable changes its value. Because of this change in state, the set of event rules is applied to the component to check if there are any events to be scheduled. If a rule fires a rule node is created and linked to the state node (the link labeled (1) in Fig. 3). The action part of the rule will lead to the scheduling of one or more events, which in turn are represented by new event linked to the rule node (labeled (2) in Fig. 3). When the event fires, it causes one ore more state changes leading to new state nodes (links (3) in Fig. 3) and so forth.

By tracing back these causal links one can show for example that the sudden rise in Fig. 2 is caused mainly by the distribution of the source and not by many items from the repair loop.

Processes

This approach is closely related to Forbus' 'Qualitative Process Theory' [5]. The simulated system is seen as a collection of passive objects with specified properties and a number of processes, which cause every change in the system. An example for a process in a material flow system would be:

Process Transition

Objects:
From.Component
To Component
Preconditions
From.Component,
Exiting.Object ≠ nil
To.Component
State = idle
Changes:
Start:
From.Component
Exiting.Object := nil
Number.Objects := Number.Objects - 1
State := idle
To.Component
Number.Objects := Number.Objects + 1
State := busy
Arriving.Object := Object.x
End:
To.Component
Arriving.Object := nil
Content := Content + Object.x

This description is similar to Weld's discrete processes [6]. Because of the complexity of the simulated systems it is often not sufficient to use a qualitative simulation, an event-oriented simulation is still necessary. In this case events are generated only by processes to continue their activity or because the preconditions of a process are met. Similar to the QPT a history can be built to gather the information about temporal/causal relations between object states and processes. With this history it is possible to answer questions about the reasons for the behavior of the system.

Shortcomings

The main problem within these approaches is the microscopic view of the system dynamics. To obtain useful results for analysis purposes it is necessary to generate abstractions. In [6] Weld described the use of aggregation techniques in a similar application, but he also pointed out some limitations. The question is if these techniques can support knowledge based analysis of complex models. Future work in our group will be directed to this problem.

References

[1] Cleary J. and Goh K.-S. and Unger B., "Discrete event simulation in Prolog", Proc. of SCS Conference on Artificial Intelligence, Graphics and Simulation, San Diego, Ca, pp. 42-47, (January 1985)

[2] O'Keefe R., "Simulation and expert systems - A taxonomy and some examples", Simulation, 46:1, pp. 10-16, January 1986

[3] Bernemann S. and Hellingrath B. and Joemann J., (1986), "How And Where Can AI Contribute To Simulation?", Proc. of the 2nd International Conference on Simulation in Manufacturing, Chicago, ILL, June 1986

[4] Reddy Y.V. and Fox M.S., (1982), "KBS: An Artificial Intelligence Approach to Flexible Simulation", Technical Report CMU-RI-TR-82-1, Robotics Insitute, Carnegie-Mellon University, Pittsburgh, PA, September 1982

[5] Forbus, K. D., "Qualitative Process Theory", Artificial Intelligence 24, pp. 85-168, 1984

[6] Weld, D.S., "The Use of Aggregation in Causal Simulation", Artificial Intelligence 30, pp. 1-34, 1986

PROBLEM SOLVING COUPLING INTERPRETATION OF NAIVE PHYSICS SIMULATIONS BASED ON ANALOGICAL REPRESENTATION AND LOGICAL INFERENCES

Francesco Gardin* & Hadley Taylor*

Department of Computer Science

University of Milan

Via Moretto da Brescia, Milan, Italy

*most of the research reported in this article was carried out at the JRC-Ispra A.I. and Robotics Laboratory, Ispra, Italy

Abstract

The ability to reason about the physical world requires a model of the world itself. The model proposed is based on Naive Physics using an analogical representation formalism, in a fashion similar to mental images. Such a model is used to run simulations whose results, collected via an interpreter module, are passed to a reasoning system used as feed-back for the simulator in a continous loop. The use of the interpreter allows also the description of a somehow quantitative simulation in qualitative terms. The interpreter is described in detail as well as its role within the problem solving system formed by the simulator, the reasoning system and the interpreter itself. Four applications implemented on a Symbolics 3600 are presented in the area of Robotics, Economy, Process Control and Diagnosis.

1.0 Introduction

A large number of problem solving tasks requires the capability of reasoning about physical processes. This implies the existance within the problem-solver of some model of the physical process which can be used to make predictions, deduce functionalities, understand misbehaviours, etc.

For such a model, although ordinary physics, employing equations can be used, there are nevertheless serious limitations to this way of modelling the physical world, the most important ones being: the need for quantitative values of the variables used in the equations, the capability to solve equations when analytic methods fail, the capability to express boundary conditions (especially spatial ones). (also use of abstract rather than common sense concepts).

A number of qualitative models have been proposed (**De Kleer 84**), (**Forbus 81**), (**Forbus**

New Knowledge Representations for Object Oriented Simulation

Steven C. Bankes
The RAND Corporation

This presentation will discuss knowledge representation issues that are critical for the application of knowledge-based/object-oriented simulation to a variety of domains. At RAND we are interested primarily in systems used for warfare modeling, however the same issues are important for other problem areas (e.g. civilian command and control situations such as fighting large forest fires).

We will begin by describing the critical shortcomings of current simulation techniques and how the RAND research program is addressing these problems through the application of emerging technologies. Among the problems being investigated at RAND are: varying the aggregation level within a simulation, inferencing over objects with multiple relations, improved support for sensitivity analysis, the use of interactive graphics, and improving simulation performance through concurrent processing.

In this presentation we will focus on one problem area which we have come to view as being of major importance in making object based simulation technology useful to a wide class of potential users. This problem is the representation of various real world phenomena which are not modeled well using current knowledge-based/object-oriented paradigms. Terrain, roads, rivers, political boundaries, weather and smoke defy easy representation as objects. It is known that some objects require special consideration in discrete event simulations in order to faithfully model continuous events such as movement or sensing that have spatial, temporal, and other physical characteristics. Phenomena such as terrain, roads, and rivers exhibit spatial continuity with respect to their graphic representation. They are difficult to represent semantically because they have many characteristics that may vary with the level of resolution required in different contexts.

Modeling these phenomena in a natural way requires the introduction of knowledge representation methods not provided in current languages and systems. What is needed is not only ways of representing these kinds of "phenomena" but also clean interfaces between various representational forms. Many entities need to be accessed in both object-like and non-object-like fashion. For example, one may wish to treat a road both as an object which can be queried about traffic which is on it, and as part of a map which has elevations at each point along its length. Even an object such as an airplane can require information about its geographical relation to other objects and thus need to be treated as embedded in a map. The fashion in which these problems are

handled has implications for both the execution-time performance of the simulation and the interface to supporting databases.

In this presentation we will report our analysis of user's needs in this area, describe the approaches we are taking in addressing these issues, and hopefully engage others in an exchange of views.

END

9-87

DTIC