COMPARISON OF QPE AND QSIM AS QUALITATIVE REASONING TECHNIQUES

BY

MICHAEL CHIEN

W. M. KIM RODDIS

A Report on Research Sponsored by

THE NATIONAL SCIENCE FOUNDATION

Research Grant

#IRI-8909196

UNIVERSITY OF KANSAS

LAWRENCE, KS

MAY 1991

ABSTRACT

Qualitative reasoning predicts and explains the behavior of physical systems using the system's structure through modeling and simulation. There are several approaches to qualitative reasoning. Two of the most prominent software implementations are QPE (Qualitative Process Engine) by Forbus and QSIM (Qualitative Simulation) by Kuipers. A comparison of the two systems is done on the basis of representation and reasoning ability of physical systems. The standard examples in qualitative reasoning and examples in fatigue and fracture in metals are used in the comparison. The fatigue and fracture domain of study can serve as a prototype for other related models of material behavior. A thorough comparison of QSIM and QPE identifies future directions of qualitative reasoning development.

ACKNOWLEDGEMENTS

This report is based on research performed by Michael Chien in partial fulfillment of the requirements for the degree of M.S.C.S. The research was supported by the National Science Foundation under NSF Grant #IRI-8909196 and by the University of Kansas General Research Fund.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

Chapter 1

Introduction

1.1 Qualitative Reasoning

Qualitative reasoning is the prediction and explanation of the behavior of physical systems using non-numeric information. This task consists of representation and reasoning of physical systems through modeling and simulation. The motivation for qualitative reasoning stems from observing the problem solving techniques of engineers and scientists. Their approach to problem solving is the simplification of difficult problems into easier problems through analysis. Careful approximations and abstraction of a problem into models simplifies the analysis of the problem. These steps are usually qualitative (e.g. since parameter A is greater than parameter B then behavior C results). These steps are repeated until all that remains is a relatively simple problem. Mathematics is used only when numerical precision is required but qualitative reasoning is used throughout the reasoning process. Thus, qualitative reasoning is a central part of problem solving.

The field of qualitative reasoning is relatively new but there are different points of view about the underlying direction of qualitative reasoning. These different views occur because qualitative reasoning is useful in solving two categories of problems. The first category is composed of the traditional engineering and science problems that already have extensive mathematical models. Qualitative reasoning is helpful and useful in this category because it can help guide selection of the proper quantitative model and because we would like to learn more about human intuition and reasoning.

The second category of problems includes those that do not have a mathematical model readily available for use. An example of a problem in this category is a leaky bucket filling with water from a faucet. Qualitative reasoning can predict the different behaviors that might occur dependent on the leak in the bucket and the rate of water falling from the faucet. This type of problem could be solved using mathematical models. However, this type of solution is undesirable or unsuitable for a variety of reasons. One reason is that the rate of water falling from the faucet might be unknown. Even though this piece of information is missing, a prediction of the possible behaviors is possible through qualitative reasoning but not through solving the equations of the mathematical model. Qualitative reasoning is useful in prediction of behaviors at a high level of abstraction through knowledge of the structure of the model (Fig.1-1).

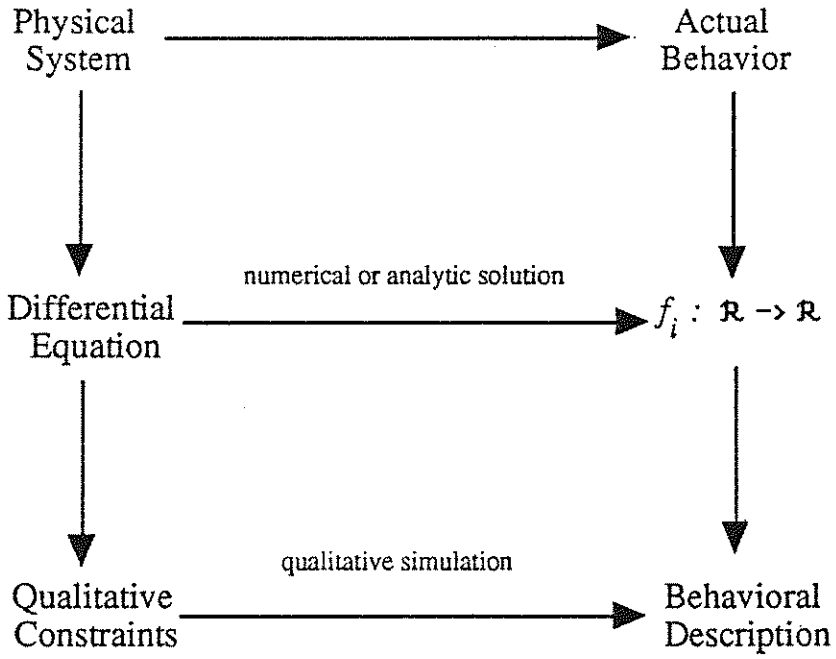

Fig. 1-1 Level of Qualitative Abstraction [Kuipers 1986]

Another reason that qualitative reasoning is useful in this category is the difficulty in building mathematical models. For instance, suppose that the faucet is turned on and off by an automatic controller. Building a mathematical model becomes somewhat tedious. Suppose there is an additional faucet filling the bucket, building the mathematical model becomes even more tedious. However, the solution of this problem is trivial compared to the time required to build a model. Qualitative reasoning provides a coarser level of modeling for creating a suitable representation for the situation. It is important to note that qualitative reasoning is not useful in problems in which the solution can not be derived from the structure of the problem (e.g. politics and the stock market).

Traditional modeling and simulation techniques also fail to address problems concerned with user and software interaction. In quantitative reasoning, it is necessary to make some assumptions about the bounds of simulation. People tend to ignore low probability states in a simulation and do not set up "stupid" initial conditions. Ignoring these states can cause critical behaviors to be pruned from the results. The generation of a description of all possible behaviors in qualitative reasoning is called an envisionment. Envisionments predict all possible behaviors which solves the simulation bounds problem. The last problem concerns the interpretation of standard simulation results. Simulation results are often only a range of numbers for the parameters of the model. It is still necessary to make correct assessments of the results.

Qualitative reasoning helps bridge the gap between user and results. The prediction and explanation of results is a central part of qualitative results. Causality describes the function of the model and not just the behavior of the model. The results

of a qualitative simulation show what can happen and to some extent what *causes* the results. In the leaky bucket example with two faucets, a possible result is that the bucket overflows. This is useful information but we would like to identify the processes that cause this behavior. If it is determined that the first faucet caused the overflow, then an adjustment can be made on the water flow rate of the faucet.

Qualitative reasoning solves many of the traditional problems associated with quantitative modeling and simulation. Expert systems used as intelligent front ends for simulation software also address many of the same issues. Rule-based expert systems have been the most prevalent use of artificial intelligence technology. However, there are a common set of problems associated with traditional expert systems. The most serious criticism of expert systems is their lack of common sense. [Forbus 1988] They do not contain a range of solution techniques, using simpler ones to solve simple problems with less work and applying more complicated techniques only when necessary. Solving problems about physical systems requires a wide range of solution techniques. Another problem is that the boundaries of the domain coverage are not well represented in today's expert systems. If the expert system does not have enough information or is addressed with a problem outside of its domain, the expert system does not degrade gracefully.

One of the most important considerations in developing intelligent tools is the reusability of software tools for other tasks. Expert systems are very good at solving very specific problems but development and maintenance are an ongoing process spanning years (e.g. R1 for VAX configuration). If building intelligent systems requires many years to accumulate and represent knowledge, then previous work must be reuseable from different points of view. The development of a reuseable library of

diverse domains requires a common framework. One of the goals of qualitative reasoning is to provide a uniform framework in which diverse domains and many types of knowledge can be integrated.

1.2 Survey of Qualitative Reasoning

Two major research works guided and formulated qualitative reasoning. The first work is the Naive Physics Manifesto [Hayes 1985]. Many of the ideas in this paper have become central ideas in qualitative reasoning. The domain of liquids is used to illustrate the problems of common sense reasoning about physical situations. One problem is the need of multiple ontologies for fluids. The use of multiple but interrelated model views in reasoning methods is often needed in problem solving.

The most influential idea on qualitative reasoning from this paper is the idea of a history. A history contains an event that is unrestricted temporally but that is restricted spatially. This temporal extension allows qualitative reasoning to focus on the event in the history instead of the representation of the event. An example of a bucket (ignore the leak for simplicity) filling with water will illustrate this point. A view of this event is that there are two objects of interest (bucket and water; we must determine which objects interact in this event) and we order this event by equal increments of time. Suppose that we want to know if the bucket fills, then the use of equal increments of time is irrelevant in this event. The results of this ordering: there is no water in the bucket, there is some water in the bucket, there is more water in the bucket, there is even more water in the bucket, the bucket is full. Histories allow time to be ordered by time points of interest. The history is: there is no water in the bucket, there is some water in the bucket, the bucket is full. Histories allow us to order events temporally

(that are of interest) and not by equal increments of time (that are not of interest).

The second influential piece of work is the NEWTON program [de Kleer 1977]. NEWTON solves textbook physics problems by creating an envisionment, a representation of all of the different possible behaviors. Envisionments organize the multiple next states in a qualitative simulation that are not present in standard quantitative simulation. Envisionments and NEWTON's ability to solve problems established the basis for future qualitative simulators.

There is a large amount of diverse research in the field of qualitative reasoning. There is a substantial amount of research in the areas of mathematical aspects of qualitative reasoning, automated modeling and multiple ontologies, integration of qualitative and quantitative reasoning techniques, causality in qualitative reasoning, qualitative kinematics, qualitative simulation, and other styles of qualitative reasoning [Weld 1988; Davis 1987]. This progress significantly contributes to the overall goals of qualitative reasoning. The testing of these ideas through software implementations can verify their feasibility and worthiness.

There are many different software implementations of qualitative reasoning techniques. Three of the most prominent general purpose qualitative reasoning systems are ENVISION, QSIM, and QPE. The device-centered confluence simulator, ENVISION [de Kleer & Brown 1985], is one of the first robust general-purpose qualitative simulators. ENVISION is based on confluences, i.e. qualitative differential equations, and provides a thorough exploration of qualitative calculus. This simulator uses a modeling view known as the device-centered ontology. Each device of a system is modeled individually and is related to other devices through confluences. The predicted behaviors of the system are dependent on the interaction between the devices.

QSIM (Qualitative Simulation) [Kuipers 1986], is also based on qualitative differential equations (QDE). The modeling view is not the device-centered approach but is based directly on the QDE's. The main distinctions of this simulator are first the creation of landmarks (important magnitudes) and second the representation of the predicted behaviors as direct history generations rather than an envisionment. The results of an envisionment are temporally generic whereas direct history generation is more temporally specific. An envisionment shows the possibilities of what the possible states are and which transitions are possible. A history corresponds to the selection of a path of transitions through the qualitative states.

Another different approach to qualitative modeling and simulation is QPE (Qualitative Process Engine) [Forbus 1986]. The modeling view is through processes. Processes are the cause of change to objects over time. This modeling view is known as the process-centered ontology. The simulator is built around an assumption-based truth maintenance system [de Kleer 1986].

1.3 Thesis Goals and Organization

The goal of this thesis is to compare and contrast QPE and QSIM as qualitative reasoning techniques to encourage further research in qualitative reasoning. A comparison of these two systems is useful for new researchers in qualitative reasoning. Both techniques have successfully modeled non-trivial situations and provided insights into the advantages and disadvantages of their respective approaches to qualitative reasoning. These two systems are chosen for comparison because of their different approaches to qualitative reasoning, the recent progress of research based around these techniques, and the availability of these systems for researchers. There are some brief

comparisons of these two systems [Crawford, Farquhar, Kuipers 1990; Forbus 1990a]. However, these comparisons are only concerned with certain aspects of the systems. A comprehensive comparison of all aspects of the two systems has not been done by researchers that are not associated with one system or the other.

This comparison of qualitative reasoning techniques consists of two main areas. The first area of study is the qualitative modeling of a physical situation. Qualitative modeling is judged by the ability to model different kinds of physical situations and the ease in which models can be built. The second area of study is the mechanism of qualitative simulation. A comparison of the results that are produced by QSIM and QPE is the main consideration. The comparison also includes information about the portability of the two systems.

The library of qualitative examples included with the implementations is tested to ensure correctness of the ported versions and to provide a standard criteria for comparison. A fair comparison of these two techniques requires the study of a domain that has not been influenced by previous work. The choice of domain for the present study of qualitative reasoning is fatigue and fracture in metals. Engineering domains such as fatigue and fracture are useful research areas for qualitative reasoning because of the complexity and broad range of knowledge necessary for problem solving. Also, the results of engineering domains are easily verifiable due to well-tested theory and wide experience of engineers in the domain of study.

Fatigue and fracture are of concern in many engineering situations including design and maintenance of bridges, tanks, piping, and other structures subject to cyclic load or temperature variations. Qualitative reasoning is used to express many of the causal relationships between crack size, crack direction, fracture toughness, and applied

stress. The ability to predict possible behaviors without numeric data or with uncertain data causes qualitative reasoning to be useful in the different contexts of failure analysis, diagnosis and prescription, and prediction. The development of a qualitative model for fatigue and fracture can also serve as a prototype for other related models of material behavior.

This thesis consists of five chapters. Chapter 2 describes the theoretical background of QSIM and QPE and the use of the implementation as tools. The basic capabilities and limitations of the systems are explored in this section as well as portability issues concerning the respective systems. Chapter 3 describes the qualitative modeling capabilities and limitations of QPE and QSIM. The ability to model the domain of fatigue and fracture is the main consideration. Chapter 4 consists of the study of the mechanism and results of qualitative simulation. The summary of findings and future research areas are in Chapter 5.

Chapter 2

QPE and QSIM Concepts

2.1 Theoretical background

Both QPE and QSIM are qualitative reasoning techniques that have been successful in modeling a variety of non-trivial physical situations. However, there are fundamental differences in their modeling and simulation techniques and also in their initial goals of research. To make an informed comparison, it is important to consider the previous intentions of the systems and the future capabilities of both systems in the modeling of more complex physical situations. An overview of the basic concepts of both systems and an overview of the implementations will be explored in the following sections. Detailed examples are covered in Chapter 3 and Chapter 4.

2.1.1 QSIM Concepts

QSIM (Qualitative Simulation) [Kuipers 1986] is a qualitative simulation technique that produces all possible qualitative behaviors of a physical situation. The original effort of this research was primarily concerned with proving correctness and completeness of the task of qualitative simulation and also with acquiring a better understanding of qualitative structure descriptions.

A model of a physical situation consists of qualitative constraints, a linearly ordered set of landmark values for all variables (called a quantity space), and the bounds of the model. Qualitative values are described by a magnitude and a direction

of change. The magnitude of a qualitative value is a symbolic value bound by its quantity space. A quantity space gives the range of a parameter annotated with important magnitudes of the parameter (called landmarks). In QSIM, landmarks are totally ordered and specified before the simulation. However, additional landmarks may be "discovered" during simulation. Qualitative magnitudes during a simulation are then categorized as being at landmarks or between landmarks. A qualitative value can have a direction of change that is either increasing, decreasing, or steady.

The constraints of a model include familiar mathematical relationships such as addition, multiplication, and derivation. Other relationships might be purely qualitative such as monotonic functions. An example of a monotonically increasing function is (M+ A B). Intuitively, this means that when A increases, B also increases. Also, this means that when A decreases, B also decreases. This constraint is very weak when compared with quantitative constraints. Monotonic functions group a large set of quantitative functions together. Quantitative constraints $A = 10^{-6} B$ and $A = 10^{6} B$ are both modeled by (M+ A B). Monotonically decreasing equations are also available. (M- A B) means that as A increases, B decreases. Other qualitative constraints (common only to version 0.4 of QSIM) will be discussed in section 3.2.2.

A qualitative simulation begins with an initial state and then proceeds to generate all possible successor states. Successor states are generated by matching each qualitative value to a possible new qualitative value through the use of a transition table (Table 2-1). These states are filtered for consistency through the constraints of the model and through various QSIM system constraints. The simulation continues until all possible behaviors are generated.

P-transitions	QS(f, t_i)	=>QS(f, t_i ,t_{i+1})	I-transitions	QS(f, t_i ,t_{i+1})	=>QS(f, t_i)
P1	<l_j, std>	<l_j, std>	I1	<l_j, std>	<l_j, std>
P2	<l_j, std>	<(l_j, l_{j+1}), inc>	I2	<(l_j, l_{j+1}), inc>	<l_{j+1}, std>
P3	<l_j, std>	<(l_{j-1}, l_j), dec>	I3	<(l_j, l_{j+1}), inc>	<l_{j+1}, inc>
P4	<l_j, inc>	<(l_j, l_{j+1}), inc>	I4	<(l_j, l_{j+1}), inc>	<(l_j, l_{j+1}), inc>
P5	<(l_j, l_{j+1}), inc>	<(l_j, l_{j+1}), inc>	I5	<(l_j, l_{j+1}), dec>	<l_j, std>
P6	<l_j, dec>	<(l_{j-1}, l_j), dec>	I6	<(l_j, l_{j+1}), dec>	<l_j, dec>
P7	<(l_j, l_{j+1}), dec>	<(l_j, l_{j+1}), dec>	I7	<(l_j, l_{j+1}), dec>	<(l_j, l_{j+1}), dec>
			I8	<(l_j, l_{j+1}), inc>	<l*, std>
			I9	<(l_j, l_{j+1}), dec>	<l*, std>

In cases I8 and I9, f becomes std at l*, a new landmark value such that $l_j < l^* < l_{j+1}$. In these cases, a previously unknown landmark value is discovered because other constraints force f'(t) to become zero.

Table 2-1 QSIM Transition Table [Kuipers 1986]

The results of a qualitative simulation are a tree of behaviors. Examples of the results of QSIM are given in Chapter 4. A qualitative behavior is defined as a sequence of qualitative states over time. The results include new landmarks that are "discovered" through simulation. Comparing new landmark values to old landmark values can distinguish between increasing, decreasing, and steady oscillations. The distinction between oscillatory behaviors is critical for many situations but this additional complexity can also create a new set of problems. [Forbus 1988] Consider for example a decaying oscillation, such as a ball bouncing up and down, each time rising only some fraction of the height it reached before. The dynamic creation of additional landmarks needs to be avoided in these cases.

2.1.2 QPE Concepts

QPE (Qualitative Process Engine) [Forbus 1986] is an implementation of Qualitative Process Theory, a framework for common sense reasoning through the use of physical processes. The goals of Qualitative Process Theory encompass general ideas about developing an effective method of common sense reasoning. Common sense reasoning is a formalization of the common sense knowledge used by humans about the physical world. Since QPE is an implementation of Qualitative Process Theory, it naturally inherits these goals. Qualitative Process Theory addresses three major properties necessary in developing a complete qualitative reasoning system. The first issue concerns causality. Causal relationships and the propagation of these relationships must be explicitly specified. The second issue concerns an important assumption in qualitative reasoning. All behaviors must be predictable from the parts of the situation and the relationships between the parts. That last issue is the ability to predict the same behavior with more precise data and the ability to resolve ambiguous data into more precise information. These three properties are important in building intelligent software problem solvers with common sense.

The framework for qualitative numbers and relationships in QPE are similar to those found in QSIM. Qualitative numbers are called quantities. There are two parts to all quantities, an amount and a derivative. The derivative of a quantity provides the direction of change information and is also a quantity. Quantities are further separated into a magnitude and a sign. Magnitudes are in symbolic terms and signs can take on values of -1, 0, 1 corresponding to the magnitude of the quantity being less-than, equal, or greater than the distinguished landmark zero. Quantities can be compared

using a quantity space. The quantity space in QPE is over different parameters of the model and are not necessarily defined individually for each parameter. Thus, a quantity space is partially ordered over the entire model in QPE while a quantity space is totally ordered over each individual parameter in QSIM.

There are two types of functional relationships in QPE. Indirect influences are qualitative proportionalities between quantities. If Q1 is positively qualitative proportional to Q2, then Q2 causes Q1 to change in the same direction assuming all other functional parameters to be equal. Direct influences are qualitative proportionalities where the derivative of Q1 is qualitatively proportional to Q2 (i.e. Q2 affects the rate of change of Q1). Direct influences are used only in processes.

Processes are central to Qualitative Process Theory. A process is something that causes changes to objects over time. Examples of processes are fluid flow, stretching, and boiling. Since only processes can cause change in a system, the issue of causality is already partially addressed. The rest of this issue is resolved through a thorough modeling paradigm. The modeling of a physical situation is divided in QPE into a domain model and a scenario model. [Forbus 1988] The domain model provides for a description of a class of related phenomena of systems. The scenario model is the description of a specific physical situation. This separation between scenario model and domain model ensures that ad hoc models are built that are robust in their description of a physical situation.

QPE domain models consist of three parts: entity (object) descriptions, views (limit points), and processes. The first part is a description of the different entities and the attributes (called quantities) of the entities. Entity descriptions include qualitative relationships between quantities that do not occur through processes. When the

quantities of an object change and reach a specific value, a limit point is reached. Views describe limit points for objects and are described in four parts: individuals, preconditions, quantity conditions, and relations. Individuals are objects that exist in a view. Those conditions that are outside of qualitative modeling and reasoning are preconditions. An example of a precondition is that a valve in a fluid connection is closed. Quantity conditions are the required conditions of the attributes of the objects. An example is that the temperature of a stove must be higher than the temperature of the fluid for boiling to occur. The last part of a view are the relations, those relationships that are true when the view is active. Processes are central to qualitative process theory and are described using individuals, preconditions, quantity conditions, relations, and influences.

QPE carries out the simulation by the following steps [Forbus, 1990].

1. Expand the scenario model.
2. Install initial assumptions.
3. Resolve unambiguous influences.
4. Construct initial situations.
5. Resolve ambiguous influences.
6. Perform limit analysis.

The results of a qualitative simulation in QPE is a total envisionment. Examples of the results of QPE are given in Chapter 4. An envisionment is a collection of qualitative states and the transitions between the states. Total envisionments have multiple initial states. The envisionment identifies all possible states and their possible transitions to all other states. These results show only possibilities of behaviors and not the actual history generation of behaviors. There are two major considerations in

the envisionments produced by QPE. The advantages of a total envisionment are that all behaviors are generated and that some failure mode of the model is not being pruned by improper selection of initial conditions. The tradeoff is that finding all possible behaviors in complex models may not be tractable.

2.2 QPE and QSIM as software tools

The comparison of two different methodologies requires investigation of the software implementations. Implementations are important to verify correctness and completeness of advances in qualitative reasoning. An exploration of issues concerning the implementations as software tools and brief overviews of the two implementations are presented in the following sections.

2.2.1 QSIM implementation

QSIM is available from the University of Texas at Austin courtesy of Benjamin Kuipers. The current version is available through anonymous ftp and is provided as a research tool (all standard disclaimers are applicable). Version 0.4 of QSIM is a Common Lisp implementation. The current version is easily ported to Symbolics machines, TI explorers, and Sun machines. There are a set of manuals (available through the University of Texas) that extensively cover the use of QSIM and the maintenance of the software.

The four sections of files are user interface, QSIM core, QSIM fundamentals, and extensions. The machine dependent user interface includes simple menu functions for accessing features. The core and fundamentals sections contain the definition and satisfaction of the qualitative constraints. There are various extensions to QSIM that are

available and running in the current version. These extensions include:

Q2 reasoning with incomplete quantitative knowledge [Kuipers, Berleant 1988]

S+ and S- constraints. (Non-Analytic Functional Constraint)

Improved time-scale abstraction simulation and plotting. [Kuipers 1987]

A graphical output section is included. This graphical output section creates the graphical representation of the behaviors and the behavior tree. The best feature of this section is the availability of hard output through the use of the Postscript standard.

2.2.2 QPE implementation

QPE is available from the University of Illinois at Urbana-Champaign courtesy of Kenneth Forbus (Norwestern University) and the Xerox Palo Alto Research Center (PARC). Xerox PARC provides the assumption based truth maintenance system (ATMS) [de Kleer 1986] which QPE is based upon. The current version is available through anonymous ftp and is provided as a research tool (all standard disclaimers are applicable). QPE version 2.1 beta test is a Common Lisp implementation. The current version is easily ported to Symbolics machines, IBM RT's and Sun 4 machines. The QPE manual comes in TeX format with the files for QPE.

The organization of the files is in three major parts. The first section consists of the assumption-based truth maintenance system. Truth maintenance systems (TMS) serve three roles in intelligent systems [de Kleer 1986]. The TMS functions as a cache for all the inferences ever made, allows the problem solver to make nonmonotonic inferences (The important point about nonmonotonicity is that information may be retracted and still maintain integrity of data), and ensures that the database is contradiction-free. A set of assumptions (i.e. context) in an ATMS are used to define a

current environment state. The use of contexts in an ATMS allows multiple solutions (due to varying qualitative assumptions) to be easily derived. Conventional TMS are designed to find the best *single* solution. Qualitative reasoning requires comparison of multiple possible solutions which is best suited for an ATMS.

The second section is QPE's interface to the ATMS. Direct access to the ATMS is possible for asserting inferences that are not available as macro primitives in QPE. However, there are a large set of macro primitives (i.e. objects, views, processes) in QPE that enable easy modeling.

The third section contains QPE's code. The features of QPE include a generic command interface (machine independent), a benchmarking system, and a textual report generation system. Machine-specific graphical systems for viewing the envisionments are available for Symbolics Release 6.1 and 7.2 and for IBM RT running Lucid Common Lisp.

2.2.3 Overview of implementations

The evaluation of AI research includes many aspects besides the performance of the AI system. Since this comparison is useful to new researchers in the field of qualitative reasoning, the aspects of portability, support, and extendibility of the implementations are investigated. One important aspect of any kind of research is the ability to reproduce the results of other researchers. In most kinds of AI research, this involves the ability to port software from one research location to another.

The selection of the hardware and the software affects the portability of the implementations. Research that involves computer implementations have been traditionally done on computer workstations. This can be attributed to workstations

providing the latest technology especially in the areas of speed and graphics capabilities. Recent advances in the personal computer have narrowed the gap between workstations and personal computers. Since there has been such a proliferation of personal computers, the higher end personal computers become possible platforms for research. The Macintosh IIfx provides the necessary amount of computing power with ease of use of the Macintosh family of computers. For this reason, the architecture chosen for this comparison was a Macintosh IIfx.

The development of Common Lisp has greatly increased portability of software in the area of artificial intelligence. Macintosh Allegro Common Lisp (MACL) version 1.3.2 was the Lisp environment chosen for the comparison. Menus and graphical capabilities are easily accessed through the use of the object oriented paradigm. The standardization of Lisp has eased the porting of large Lisp programs from one implementation to another. Unfortunately, the Common Lisp Object System (CLOS) is not used in this version of MACL. The standardization of graphic capabilities in Lisp would greatly benefit the portability of these two implementations. The largest time and effort was spent on these graphic aspects which are important in visualization of the qualitative results.

The support available is highly commendable for both implementations. Forbus, Kuipers, and their graduate students were very helpful with porting QPE and QSIM to MACL. Most of the code is independent of the machine and the machine-dependent parts of the code are isolated into clearly marked sections. The set of manuals available with QSIM through the University of Texas are excellent. Specific problems that were encountered in porting these implementations to MACL are given in Appendix E and F.

Extending the implementations is fairly easily done because of the well-documented code and the modularity of the software. QSIM includes the most stable extensions (Sec. 2.2.1) with the code. The use of time scale abstraction and the integration of quantitative constraints are the most promising extensions of those included in the current version. There are a few examples which illustrate the basic use of these extensions but the extensions are not error-free and do not always behave as prescribed. A thorough understanding of the internals of QSIM is necessary to test and use these extensions. Other extensions that are not available with the current version of QSIM are a more natural syntax for equation constraints [QSIM User's Manual 1990]; MIMIC, a model-based monitoring of dynamic systems [Dvorak & Kuipers 1990]; and QPC, a qualitative process compiler [Crawford, Farquhar, Kuipers 1990]. QPC is a model building system similar to the model building of QPE which uses QSIM as the underlying simulator. QPC is discussed in detail in Chapter 3.

Some extensions that are scheduled to be released with the initial version of QPE are a method of reconstructing an envisionment without recomputation, a syntax checker, domain-specific sets, and a batch mode that allows the computation of envisionments on other machines from one server [QPE Manual 1990]. Other extensions include mathematical extensions to QPE [D'Ambrosio 1990]; the use of probabilities with QPE [D'Ambrosio (in press)]; and SIMGEN, a self-explanatory simulator that integrates qualitative and quantitative knowledge [Forbus & Falkenhainer 1990].

Chapter 3

Findings in Qualitative Modeling

3.1 Issues in Modeling

Modeling of a physical system is the first step in qualitative reasoning. A model is a cost-effective representation used to predict the behavior of a physical system [Rothenberg 1989]. Qualitative reasoning provides an important modeling granularity that has many advantages and disadvantages. The foundation of qualitative modeling is the formalization and understanding of the basic qualitative calculus (numbers, inequalities, functions). Modeling is explored through the U-tube example (available in the libraries accompanying both QPE and QSIM) and through new models created in the domain of fatigue and fracture of metals. Overall, modeling is judged by the ability to model different kinds of physical situations and the ease in which models can be built.

3.1.1 Origin of Qualitative Models

The origin of qualitative models is an interesting issue of concern. In QSIM, qualitative differential equations (QDE) are an abstraction of ordinary differential equations. However, QDE's are rarely abstracted directly from ordinary differential equations (ODE). In fact, qualitative reasoning is often used to represent systems not easily modeled through standard mathematical methods. [Kuipers 1988] presumes that an ODE model exists for all physical systems and that the solutions of ODE's are the best standard of comparison for qualitative models. Qualitative Process Theory models

systems through the use of processes. This modeling technique is motivated by people's description of physical changes in the world. Processes are the cause of all changes in a system. Examples of processes are heating, water flow, boiling, and stretching. [Forbus & Gentner 1986] present processes as central to human learning of physical domains through their implied causality. The bulk of qualitative models (so far) are developed through common sense observations and/or abstractions of mathematical models.

3.1.2 Qualitative Calculus

The basic qualitative calculus (numbers, inequalities, and functions) of QSIM and QPE are similar (discussed in 2.2.1 & 2.2.2). Numbers are defined by a magnitude and a sign (direction of change). The sign of a number is determined by comparison between the magnitude of the number and landmarks (distinguished magnitudes). In model building, landmarks are known *a priori* to the simulation. A discussion of the additional landmarks inserted during the simulation by QSIM is included in Chapter 4. Qualitative constraints and qualitative proportionalities are similar in meaning. The difference is that qualitative proportionalities are only necessarily true when all other parameters of the model remain equal. This difference does not show up in the modeling process but is used in influence resolution during the simulation. Addition and multiplication constraints are used in both implementations. These constraints maintain sign relationships (Fig. 3-1).

$$f + g = h \quad \text{or} \quad f * g = h \;\; f > 0, g > 0, h > 0$$

f \ g	inc	std	dec
inc	inc	inc	any
std	inc	std	dec
dec	any	dec	dec

Fig 3-1 Addition and Multiplication Constraints [Kuipers 1986]

3.1.3 Basic Modeling

There are a number of issues to be resolved before the actual model building phase. First, the purpose of the model is determined. This consists of narrowing the number of aspects of the physical system to be modeled. A common mistake in modeling is to make an extremely complicated representation. The goal of modeling is to abstract away unnecessary or unimportant aspects of the modeled system and to determine the possible behaviors. The granularity of the model is selected next. Qualitative models can represent things on the microscopic level of detail or they can represent things on a macroscopic level of detail. The selection of the granularity is dependent on the type of behaviors that are of interest (i.e. modeling electron flow in semiconductors is not needed if the behavior of interest is the overall circuit behavior). The parts of the model (parameters or objects) and the relationships between the parts can then be determined.

3.2 Modeling in QSIM

There are two main parts in a QSIM representation of a physical system. The first part is the basic QSIM model consisting of the parameters of the model, their quantity spaces, and the boundaries of the model. The second part is the set of initial conditions used to initiate simulation.

3.2.1 The U-tube example

A U-tube is a two tank system that is connected by a fluid path.

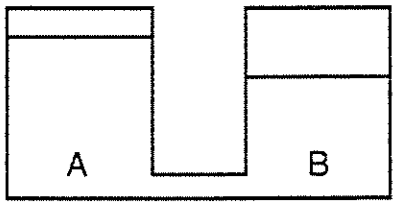

Fig. 3-2 Picture of a U-tube

The U-tube is a simple example illustrating basic qualitative modeling techniques (available in the libraries accompanying QPE and QSIM). Reasoning about the system requires modeling of several parameters. The amount of fluid in each tank is denoted as amount A and amount B respectively. Similarly, the pressure in each tank is denoted as pressure A and pressure B respectively. The system behaves in the following manner: If the pressure in one tank is greater than the pressure in the other tank, then there is fluid flow from the higher pressure tank to the lower pressure tank. The fluid flow causes the amount of fluid in the tanks to increase or decrease

respectively. This change in the amount of fluid affects the pressure in its corresponding tank. When the pressure in the tanks are equal, then the fluid flow stops.

In QSIM, these relationships are defined with the following monotonic constraints:

pressure A = M^+ (amount A)

pressure B = M^+ (amount B)

total amount = amount A + amount B

pressure difference of A to B = pressure A - pressure B

flow from A to B = d/dt amount A

flow from A to B = - d/dt amount B

flow from A to B = M^+ (pressure difference of A to B)

Next, the quantity space of each parameter is determined. The quantity space determines the range for each parameter. Initially, we will assume the range of each parameter extends from negative infinity ("minf") to positive infinity ("inf"). Next, the insertion of the zero landmark between these bounds is necessary. The internal code of QSIM recognizes "0" as a special landmark so including "0" in the quantity space means that the range is both negative and positive. Since the U-tube example is a model of an actual physical system, having negative fluid is impossible. Therefore, we eliminate "minf" from the quantity space of amount A, amount B, and the total amount. Similarly, negative pressure is meaningless in this system. We eliminate "minf" from the quantity space of pressure A and pressure B. In the U-tube example, the types of behaviors that are of interest are the flow of fluid between tanks. The only other meaningful landmark is that the tanks contain a limited amount of fluid. Therefore, we

include the landmarks of Amax and Bmax.

Corresponding values are also included in the constraints of the model. This means that if the amount A = 0, then the pressure A = 0. Also, if amount A = inf, then pressure A = inf. Corresponding values are possible for all constraints except for derivative constraints.

The last part of concern is the bounds of the model. In the QSIM version, the tanks are closed. If we reach a qualitative state where amount A (or amount B) increases to Amax (or Bmax), then the monotonic constraints are no longer true. Therefore, if further simulation of the model is needed, then a transition is made to another model. In the QSIM example, we assume that tank B will burst if the tank is full and there is still potential fluid flow from A to B. Then, when amount B = Bmax and it's direction of change is increasing, a transition is made to another model.

The basic model of the U-tube is defined (Fig. 3-3). The next step is defining the initial conditions of the model. If we define that tank A is full and tank B is empty, then this is enough information to specify an initial state. The direction of change of amount A and amount B can also be defined. The four choices are increasing, steady, decreasing, or nil. Nil means that the direction of change is unknown. It is also possible to define parameters as being between landmarks in the initial conditions. For instances, it is possible to say that amount A is between 0 and Amax initially. With the initial conditions specified, the simulator is called.

```
(define-QDE U-tube
 (text "U-tube")
 (quantity-spaces
  (amtA                    (0 Amax inf))
  (amtB                    (0 Bmax inf))
  (total                   (0 inf))
  (pressureA               (0 inf))
```

```
   (pressureB                          (0 inf))
   (pAB                                (minf 0 inf))
   (flowAB                             (minf 0 inf))
   (mflowAB                            (minf 0 inf)))
  (constraints
   ((M+ amtA pressureA)                (0 0) (inf inf))
   ((M+ amtB pressureB)                (0 0) (inf inf))
   ((ADD amtA amtB total))
   ((ADD pAB pressureB pressureA))
   ((M+ pAB flowAB)                    (0 0) (inf inf))
   ((d/dt amtB flowAB))
   ((minus flowAB mflowAB)             (inf minf) (minf inf))
   ((d/dt amtA mflowAB))
   ((constant total)))
  (transitions
   ((amtB (Bmax inc)) tank-B-burst))
   ((amtA (Amax inc)) t)))


(defun simple-U-tube-figure ()
 (declare (special u-tube))
 (let* ((init (make-initial-state U-tube
                                  `((amtA (Amax nil))
                                    (amtB (0 nil)))
                                  "Tank A full; B empty"))
        (nlayout '((amtA amtB total)
                   (pressureA pressureB)
                   (pAB flowAB)))
        )
   (format *QSIM-Report*
"~2%  This is the simple U-tube example:  a closed two-tank system, starting with
tank A full and tank B empty.  We get the usual three-way branch according to
whether the system reaches equilibrium before tank B overflows.

For dramatic effect, if tank B overflows, it bursts!
We get a region transition to a model where tank B has lost all its contents.
The contents of tank A now drain across the channel, unopposed by backpressure
from tank B, until the entire system is empty.  Meltdown! ~2%")
   (qsim init)
   (qsim-display init :layout nlayout)
   ))
```

Fig. 3-3 QSIM U-tube model

3.2.2 Modeling Extensions in QSIM

There are many types of extensions for modeling in QSIM. One of the problems with qualitative reasoning is that the level of abstraction may be too high for particular models and the loss of quantitative information is costly. In most modeling and simulation problems, some quantitative information is known. However, most of the current qualitative simulators can not use this information. [Kuipers & Berleant 1988] introduce a method (Q2) of using incomplete quantitative knowledge in qualitative reasoning. This approach augments qualitative reasoning techniques by "narrowing" the possible qualitative values. In effect, this approach can benefit qualitative reasoning by discovering which qualitative behaviors are inconsistent. For instance, in the U-tube example, if tank B is larger than tank A, then it is not possible for tank B to become full. Also, the use of quantitative ranges for the qualitative constraints is possible. The quantitative ranges restrict the monotonic constraints into a strict range. Q2 reasoning is running in version 0.4 of QSIM. It must be stressed that the quantitative integration only restricts the qualitative reasoning and quantitative equations can not be simulated.

Another addition to QSIM is the use of non-analytic functions. The S+ function [QSIM Manual 1990] is defined as:

$y(t) = c$ if $x \leq a$;

$y(t) = d$ if $x \geq b$;

$y(t) = f(x(t))$ otherwise

where a and b are landmarks in the quantity space for x, and c and d are landmarks in the quantity space for y (Fig. 3-4).

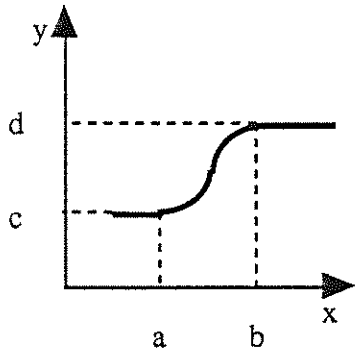

Fig. 3-4 The S+ function in QSIM

The use of the S+ constraint can be interpreted as the two parameters acting at different time rates but both parameters approaching corresponding limits. An example (from QSIM's library) of the use of this constraint is a rocket reaching escape velocity from the earth's gravity (Fig. 3-5).

```
; The constraint A = S+(Y) allows us to have a monotonic function
; constraint where A'(t)=0 while Y'(t)>0, so the rocket slows down
; asymptotically to a positive (non-zero) velocity.  This corrects a
; previous bug, where that behavior was excluded.

(define-QDE rocket
 (text "Rocket:  projectile with initial velocity and decreasing gravity.")
 (quantity-spaces
  (y   (minf 0 inf))
  (v   (minf 0 v* inf))
  (a   (minf g 0)))
 (constraints
  ((d/dt v a))
  ((d/dt y v))
  ((s+ y a (minf minf) (inf 0))  (0 g))
  )
 (transition    ((y (0 dec)) t))
 (layout (nil nil nil y)
         (nil nil nil v)
```

(nil nil nil a)
nil))

Fig. 3-5 Rocket example

Another extension of QSIM is the use of time-scale abstraction. Time-scale abstraction [Kuipers 1987] is a useful modeling technique that allows constraints to occur at different time scales. The use of this hierarchial method is a major attempt at reducing a complex model into a number of subsystems. This modeling technique is natural for some domains but this type of hierarchial system can not be naturally imposed on other domains (e.g. electronic circuits). The use of this method in a two-level time-scale hierarchy for modeling the water balance mechanism (fast) and the sodium balance mechanism (slow) of a kidney has been successful.

The Qualitative Process Compiler (QPC) [Crawford, Farquhar, Kuipers 1990] is another modeling system that uses QSIM as the underlying simulator. QPC models through the use of processes (from Qualitative Process Theory). QPC shows the need for modeling extensions for QSIM and that QSIM can be used with various types of modeling techniques.

3.3 Modeling in QPE

3.3.1 The U-tube example

The U-tube example is a model of fluid flow between two tanks through a fluid connection (Fig. 3-2). Modeling in QPE consists of a domain model (Figs. 3-6, 3-7, 3-8) and a scenario model (Fig. 3-9). The domain model will include all related

phenomena about liquids including boiling, freezing, heat flow, and fluid flow. A well designed and carefully thought out domain model can be reused for a variety of situations that might use some or all aspects of the domain model. In the U-tube example, we are only concerned with fluid flow.

Fluid flow is the process that causes change in the U-tube system (Fig. 3-6). A process is made up of individuals, preconditions, relations, and influences (Sec. 2-1-2). The definition of fluid flow is a substance in the liquid form moving from one container to another container. We will also include that there must be a fluid connection between the two containers. The individuals (objects) in the fluid-flow process are a liquid substance, two containers, and a fluid connection. In the preconditions section, the fluid connection must be open for fluid flow to occur. Note that the preconditions section contain conditions that are outside of the simulator's reasoning ability. The preconditions must be true before anything can possibly happen and there are not any processes in the domain model that will change the status of the preconditions. However, the simulator can and will note differences in the pressure difference between the tanks. This fact is defined in the quantity conditions as the pressure in the source must be greater than the pressure in the destination for fluid flow to occur. So far, we have identified the objects that must be present in the process and the conditions that must hold for the process to be active.

We now define the consequences of the process being active as relations and influences. The quantity flow-rate is defined as qualitatively equal to the pressure difference between the two tanks. Qualitatively equal is defined as equality between quantities in which the magnitudes and the derivatives of the two quantities are equal. A direct influence of the process is that the derivative of the amount of fluid in the

destination container is positively qualitatively proportional to the flow rate. The second influence is the derivative of the amount of fluid in the source container being negatively proportional to the flow rate. The complete process is given in (Fig. 3-6).

```
(defprocess (Liquid-flow ?sub ?src ?dst ?path)
        Individuals ((?sub :type Substance)
                     (?src :type Container)
                     (?dst :type Container)
                     (?src-cl :bind (C-S ?sub LIQUID ?src))
                     (?dst-cl :bind (C-S ?sub LIQUID ?dst))
                     (?path :type Fluid-Path
                            :conditions
                            (Fluid-Connection ?path ?src ?dst)))
        Preconditions ((aligned ?path))
        QuantityConditions
        ((greater-than (A (pressure ?src-cl)) (A (pressure ?dst-cl))))
        Relations ((quantity flow-rate)
                   (Q= flow-rate (- (pressure ?src-cl) (pressure ?dst-cl)))
                   (filled ?path)
                   (greater-than (A flow-rate) zero))
        Influences ((I+ (Amount-of-in ?sub LIQUID ?dst) (A flow-rate))
                    (I- (Amount-of-in ?sub LIQUID ?src) (A flow-rate))))
```

Fig. 3-6 The liquid flow process defined in QPE

There are other relationships that must be identified in the domain model. There are two ontologies of fluid called the contained stuff ontology and the pieces of stuff ontology. The contained stuff ontology allows for easier reasoning about containment and fluid flow. Thermodynamic properties of fluids are represented in the pieces of stuff ontology (See [Hayes 1985] or [Collins & Forbus 1987] for a complete discussion on the two ontologies of fluid). These two ontologies are related by the use of the view structure (Fig. 3-7). The view states that if a substance is contained, then the substance can also be viewed as pieces of stuff.

```
(defview (Contained-Stuff (C-S ?s ?st ?c))
        Individuals ((?c :type container)
                     (?s :type substance)
                     (?st :type state))
        Preconditions ((Can-Contain-Substance ?c ?s ?st))
        QuantityConditions ((greater-than (A (Amount-of-in ?s ?st ?c)) ZERO))
        Relations ((there-is-unique (C-S ?s ?st ?c))
                   (Q= (amount-of (C-S ?s ?st ?c)) (amount-of-in ?s ?st ?c))
                   (qprop (amount-of (C-S ?s ?st ?c)) (amount-of-in ?s ?st ?c))
                   (physob (C-S ?s ?st ?c))))
```

Fig. 3-7 View structure for contained stuff

In this example, we are mainly concerned with the contained stuff ontology. The contained liquid is defined as an object. The only condition that must hold in this case is that the temperature of the substance must be less than it's boiling temperature (i.e. the substance is in the liquid state). We define a parameter called *level* to indicate the amount of fluid in a container. The other parameters are defined in the pieces of stuff ontology. The qualitative proportionalities (Qprop) that hold in the contained stuff entity are:

1.The amount of fluid (Qprop+) the level of fluid in the container.

2.The level of fluid in the container (Qprop+) the pressure in the container.

Also, if the level of the fluid in the container equals the bottom height of the container, then the amount of fluid in the container is zero. The full description is given in (Fig. 3-8).

```
(defentity (Contained-Liquid (C-S ?sub liquid ?can))
        ;;; Liquids have a novel quantity, level.
        (quantity (level (C-S ?sub liquid ?can)))
        ;;; Bound temperature from above.
```

```
(not (Greater-Than (A (temperature (C-S ?sub liquid ?can)))
                   (A (Tboil ?sub ?can))))
;;; Now specify some functional relationships
(Function-Spec Level-Function
               (Qprop (level (C-S ?sub liquid ?can))
                      (Amount-of (C-S ?sub liquid ?can))))
(Correspondence ((A (level (C-S ?sub liquid ?can)))
                 (A (bottom-height ?can)))
                ((A (amount-of (C-S ?sub liquid ?can))) zero))
;;; This should depend on the container being open
(Function-Spec P-L-Function
               (Qprop (pressure (C-S ?sub liquid ?can))
                      (level (C-S ?sub liquid ?can)))))
```

Fig. 3-8 Contained Liquid Object

The last description that is necessary is the scenario model. The scenario model will describe the actual physical representation of the situation (Fig. 3-9).

```
;;; Two containers example

(assertq (state liquid))
(assertq (substance water))

;; Declare individuals and their types
(assertq (container F))
(assertq (container G))
(assertq (fluid-path P1))

;; Specify their connectivity
(assertq (fluid-connection P1 F G))
(assertq (fluid-connection P1 G F))

;pin down the geometry a bit.
(assertq (equal-to (A (bottom-height f)) (A (max-height p1))))
(assertq (equal-to (A (bottom-height g)) (A (max-height p1))))
```

Fig. 3-9 Scenario model for the U-tube

3.3.2 Modeling extensions using QPE

There are two major modeling extensions that are based around QPE. SIMGEN [Forbus & Falkenhainer 1990] is a system that generates self-explanatory simulations using both qualitative and quantitative knowledge. This extension takes a qualitative domain model, a corresponding math-model library, and a specific system to model. The qualitative model guides the quantitative models and also provides explanations. The limitations of this system are in building proper qualitative domain models. SIMGEN works best in domains that support causal reasoning rather than domains that use simplifying algebraic analyses. This approach is opposite to Q2 reasoning. Quantitative reasoning guides qualitative reasoning in Q2 whereas SIMGEN uses both qualitative reasoning and quantitative reasoning and specifically uses qualitative models to guide quantitative models.

[Falkenhainer & Forbus 1988] addresses the issue of setting up large-scale qualitative models. Their approach sets up a hierarchial system for modeling which makes assumptions. These assumptions are used in selecting a specific granularity of the model and in making assumptions about the operating conditions of the model. These assumptions can also be modeled through QPE macros and the scenario model. An example using fatigue and fracture is given in (Sec 3-4).

3.4 Modeling fatigue and fracture of metals

Fatigue and fracture of metals is a domain of major concern because of the possibly catastrophic results of failure. In modeling a simple prototype, we are concerned with fatigue and fracture failure and with the causes of the failure. Metal fatigue is a process which causes failure or damage of a component subjected to

repeated loading [Bannantine et. al. 1990]. Brittle fracture is a type of catastrophic failure in structural materials that usually occurs without plastic deformation and at extremely high speeds (as high as 7000 ft/sec) [Barsom & Rolfe 1987]. Fatigue failure is modeled by the condition deltaK = K_T where deltaK is the stress intensity range and K_T is the upper fatigue-rate transition value. Fracture failure is modeled by the condition K = K_c where K is the stress intensity factor and K_c is the fracture toughness. K_T is a landmark in the quantity space for deltaK and K_c is a landmark in the quantity space for K. We include the landmark K_{TH}, lower fatigue-rate transition value in the quantity space for deltaK and define the fatigue process as occurring when K_{TH} < deltaK < K_T. The other parameters of the simple prototype of fatigue and fracture in metals are vertical crack dimension (a), horizontal crack dimension (c), and temperature (T).

The relationships of the parameters are in (Fig. 3-10). The dotted line in the figure denotes that K_c is in the quantity space for K. Solid lines denote the qualitative constraints that hold in the model. The notation of M+ is used for both qualitative proportionalities (in QPE) and monotonic constraints (in QSIM).

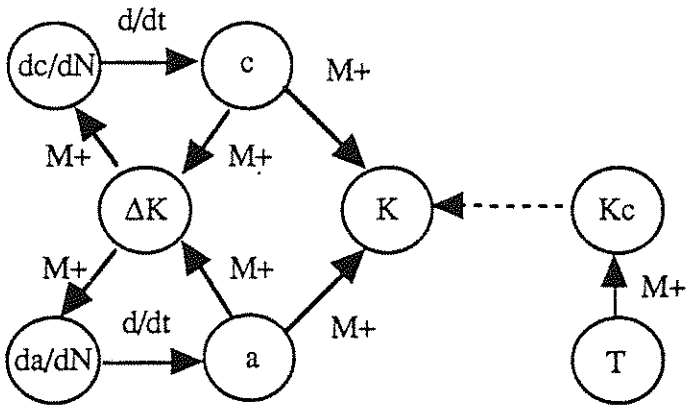

Fig. 3-10 Qualitative Constraints of Fatigue and Fracture Model

There are some assumptions that are made in this model. First, it is assumed that there is some initial crack but the model does not account for the origin of the crack. Also, it is assumed that there is only one crack in the structure and that crack will propagate through time because of fatigue. Another assumption is that the fatigue process occurs because of some cyclic loading that is not represented specifically in the model (The model ignores the effects of loading on the fracture process).

The driving loop of the model is the fatigue process. There is some initial crack in a metal structure. The cyclic loading on the structure causes fatigue to occur (deltaK > K_{TH}). This causes the crack to grow either horizontally, vertically, or in both directions. We limit the geometric considerations to two dimensions for simplicity. In turn, K and deltaK become larger. The rate of fatigue increases and the crack continues to grow. This cycle continues until either fracture failure occurs or fatigue failure occurs. There are two other limits that can be reached before failure occurs. It is possible for the vertical length of the crack to reach the vertical dimension of the structure. Correspondingly, it is possible for the horizontal length of the crack to reach the horizontal dimension of the structure. Another thing that can occur is that the ambient temperature can drop and then K_c drops. Fracture failure can occur because K_c drops to K.

The modeling process for fatigue and fracture for both QPE and QSIM is an incremental and iterative process. Simple models are built and then more complex models are incrementally constructed. Correct modeling requires many iterations of examining the model for inconsistencies and careful study of the results of simulation. In the first model for fatigue and fracture, crack growth was restricted to the vertical

dimension and the temperature factor was not included. The basic QSIM model is given in (Fig. 3-11) and the basic QPE model is given in (Fig. 3-12).

```
(define-QDE crack-model
 (text "Basic qualitative fatigue and fracture behavior.")
 (quantity-spaces
          (deltaK       (0 kth kt))
          (da/dN        (0 fast_da/dN inf))
          (a            (0 a_i thickness)))
 (constraints
          ((M+ a deltaK)                          )
          ((M+ deltaK da/dN)     (kth 0)(kt fast_da/dN))
          ((d/dt a da/dN)                         ))
 (print-names (deltaK   "stress intensity range")
              (da/dN    "crack growth rate")
              (a        "crack length"))
 (layout
       (nil deltaK)
       (nil da/dN)
       (nil a)))

; Model fatigue and fracture behavior.

(defun model-FFB ()
 (let ((initial-state
          (make-initial-state crack-model
                              '((a (a_i inc))))))
  (qsim initial-state)
  (qsim-display initial-state)
  ))
```

Fig. 3-11 Simple Fatigue and Fracture Model in QSIM

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Package: USER -*-

(in-package qpe::*user-package*)

(adb::rules-file "FF-DOMAIN")

;;;; Fatigue and Fracture Theory for QPE
;;
```

```
(defQuantity-Type vertical-crack Individual)
(defQuantity-Type deltaK Individual)
(defQuantity-Type Kt individual)
(defQuantity-Type fatigue-rate Individual)

;;; Direct access to ATMS
;;; These rules are later removed because
;;; these things can be modeled in the metal entity

(adb:Rule :INTERN (((metal ?m) . :TRUE))
          ;; There is some vertical-crack in all metals
          (adb:rassert! ((quantity (vertical-crack ?m)) . :TRUE))
          ;; vertical-crack is never negative.
          (adb:rassert! ((Greater-Than (A ((vertical-crack ?m)) zero) . :TRUE)))
          ;; There is a Kt for all metals
          (adb:rassert! ((quantity (Kt ?m)) . :TRUE))
          ;; Kt is never negative.
          (adb:rassert! ((Greater-Than (A ((Kt ?m)) zero) . :TRUE))))

(defentity metal
          ;;; Main characteristic is that it has a number of quantities
 (quantity (deltaK ?self))
 (quantity (vertical-crack ?self))
 (quantity (Kt ?self))
 ;;; There are a few state-independent relationships
 (Qprop (deltaK ?self)(vertical-crack ?self))
 (not (less-than (A (vertical-crack ?self)) zero)))

;;;; View vocabulary

(defview (fatigue-failure ?material)
 Individuals ((?material :type metal))
 QuantityConditions ((equal-to (A (deltaK ?material)) (A (Kt ?material)))))

;;;; Process vocabulary

(defprocess (fatigue ?material)
 Individuals ((?material :type metal))
 QuantityConditions ((less-than (A (deltaK ?material)) (A (Kt ?material)))
                     (greater-than (A (vertical-crack ?material)) zero))
 Relations ((quantity fatigue-rate)
            (Q= fatigue-rate (deltaK ?material))
            (greater-than (A fatigue-rate) zero))
 Influences ((I+ (vertical-crack ?material) (A fatigue-rate))))
```

Fig. 3-12 Simple Fatigue and Fracture Domain Model in QPE

Both models (Fig. 3-11 & Fig. 3-12) correctly describe the basic fatigue feedback loop. However, both models do not have any information about the bounds of the model or fracture failure, and is limited to one dimensional crack growth. In QSIM, a transition is required to another model to describe the behavior in which the vertical dimension of the structure and the vertical crack length are equal but the structure has not failed. These issues are addressed in the next QSIM model.

The QPE domain model "separates" the fatigue process (in the *process* macro) from the other qualitative proportionalities (in the *entity* macro). Also, the view structure distinctly defines when an interesting behavior occurs, in this case, that fatigue failure has occurred. The fatigue failure view is active when deltaK = K_T. The QSIM model does not explicitly define fatigue failure. The simulation stops when deltaK reaches a boundary condition of its quantity space (i.e. deltaK = K_T).

The next QSIM model (Fig. 3-13) and QPE model (Fig. 3-14) adds the horizontal dimension for crack growth and fracture failure.

```
;;; -*- Syntax: Common-lisp; Package: qsim; Default-character-style: (:FIX :ROMAN
:NORMAL) -*-

; This is a modified version of figure 5-4 page 115
; of K. Roddis PhD thesis at MIT 1988.
; Basic fatigue and fracture behavior

(define-QDE crack-model
 (text "Basic qualitative fatigue and fracture behavior.")
 (quantity-spaces
        (deltaK      (0 kth kt))
        (da/dN       (0 fast_da/dN))
        (dc/dN       (0 fast_dc/dN))
        (a           (0 thickness))
        (c           (0 width))
        (K           (0 inf))
```

```
        (Kr          (0 inf))
        (Kc          (kmin kmax))
        (C&A         (0 t+w))
        )

 (constraints
        ((ADD Kr K Kc)              )
        ((constant Kc)              )
        ((ADD c a C&A)     (width thickness t+w))
        ((M+ C&A K)        (0 0))
        ((M+ C&A deltaK)   (0 0))
        ((M+ deltaK da/dN) (kth 0)(kt fast_da/dN))
        ((M+ deltaK dc/dN) (kth 0)(kt fast_dc/dN))
        ((d/dt a da/dN)             )
        ((d/dt c dc/dN)             )
        )

 (transitions
        ((c (width inc)) no-more-horizontal-growth)
        ((a (thickness inc)) no-more-vertical-growth)
        )

 (layout
       (Kr Kc)
       (K deltaK)
       (c dc/dN)
       (a da/dN))
 )

(defun no-more-vertical-growth (growth-state)
  (create-transition-state :from-state  growth-state
                  :to-qde     no-vertical-growth
                  :assert     '((a (thickness std))
                                (da/dN (0 std)))
                  :inherit-qmag :rest
                  :inherit-qdir :rest))

(defun no-more-horizontal-growth (growth-state)
  (create-transition-state :from-state  growth-state
                  :to-qde     no-horizontal-growth
                  :assert     '((c (width std))
                                (dc/dN (0 std)))
                  :inherit-qmag :rest
                  :inherit-qdir :rest))

(define-QDE no-vertical-growth
  (text "Basic qualitative fatigue and fracture behavior.")
  (quantity-spaces
```

```
        (deltaK      (0 kth kt))
        (da/dN       (0 fast_da/dN))
        (dc/dN       (0 fast_dc/dN))
        (a           (0 thickness))
        (c           (0 width))
        (K           (0 inf))
        (Kr          (0 inf))
        (Kc          (kmin kmax))
        (C&A         (0 t+w))
        )

(constraints
        ((ADD Kr K Kc)              )
        ((constant Kc)              )
        ((ADD c a C&A)    (width thickness t+w))
        ((M+ C&A K)       (0 0))
        ((M+ C&A deltaK)  (0 0))
        ((M+ deltaK dc/dN (kth 0)(kt fast_dc/dN))
        ((d/dt a da/dN)             )
        ((d/dt c dc/dN)             )
        )

(transitions
        ((c (width inc)) no-more-horizontal-growth)
        ;((a (thickness inc)) no-more-vertical-growth)
        )

(layout
      (Kr Kc)
      (K deltaK)
      (c dc/dN)
      (a da/dN))
)

(define-QDE no-horizontal-growth
 (text "Basic qualitative fatigue and fracture behavior.")
 (quantity-spaces
        (deltaK      (0 kth kt))
        (da/d (0 fast_da/dN))
        (dc/dN       (0 fast_dc/dN))
        (a           (0 thickness))
        (c           (0 width))
        (K           (0 inf))
        (Kr          (0 inf))
        (Kc          (kmin kmax))
        (C&A         (0 t+w))
        )
```

```
(constraints
        ((ADD Kr K Kc)              )
        ((constant Kc)              )
        ((ADD c a C&A)      (width thickness t+w))
        ((M+ C&A K)         (0 0))
        ((M+ C&A deltaK)    (0 0))
        ((M+ deltaK da/dN)  (kth 0)(kt fast_da/dN))
        ((d/dt a da/dN)             )
        ((d/dt c dc/dN)             )
        )

(transitions
        ;((c (width inc)) no-more-horizontal-growth)
        ((a (thickness inc)) no-more-vertical-growth)
        )

(layout
      (Kr Kc)
      (K deltaK)
      (c dc/dN)
      (a da/dN))
)

; Model fatigue and fracture behavior.

(defun model-FFB ()
 (let ((initial-state
        (make-initial-state crack-model
                            '(
                    (Kc ((kmin kmax) std))
                    (c ((0 width) inc))
                    (a ((0 thickness) inc))
                    (deltaK ((kth kt) inc))
                    ))))
  (qsim initial-state)
  (qsim-display initial-state)
  ))
```

Fig. 3-13 Second Fatigue and Fracture Model in QSIM

```
;; -*- Mode: LISP; Syntax: Common-lisp; Package: USER -*-

(in-package qpe::*user-package*)

(adb::rules-file "FF-DOMAIN")
```

```
;;;; Fatigue and Fracture Theory for QPE
;;
(defQuantity-Type vertical-dimension Individual)
(defQuantity-Type horizontal-dimension Individual)
(defQuantity-Type vertical-max individual)
(defQuantity-Type horizontal-max individual)
(defQuantity-Type deltaK Individual)
(defQuantity-Type K Individual)
(defQuantity-Type Kt individual)
(defQuantity-Type Kc individual)

(defentity metal
        ;;; Main characteristic is that it has a number of quantities
  (quantity (deltaK ?self))
  (quantity (K ?self))
  (quantity (vertical-dimension ?self))
  (quantity (horizontal-dimension ?self))
  (quantity (Kt ?self))
  (quantity (Kc ?self))
  (quantity (vertical-max ?self))
  (quantity (horizontal-max ?self))
  ;;; There are a few state-independent relationships
  (Qprop (deltaK ?self)(horizontal-dimension ?self))
  (Qprop (deltaK ?self)(vertical-dimension ?self))
  (Qprop (K ?self)(vertical-dimension ?self))
  (Qprop (K ?self)(horizontal-dimension ?self))
  ;;dimensions are never negative
  (not (less-than (A (vertical-dimension ?self)) zero))
  (not (less-than (A (horizontal-dimension ?self)) zero))
  ;; deltaK is not negative
  (greater-than (A (deltaK ?self)) zero)
  ;;these are all positive values
  (greater-Than (A (vertical-max ?self)) zero)
  (greater-Than (A (horizontal-max ?self)) zero)
  (greater-Than (A (Kc ?self)) zero)
  (greater-Than (A (Kt ?self)) zero)
  ;; Physical impossibilities for this model
  (not (greater-than (A (deltaK ?self)) (A (Kt ?self))))
  (not (greater-than (A (K ?self)) (A (Kc ?self))))
  (not (greater-than (A (vertical-dimension ?self)) (A (vertical-max ?self))))
  (not (greater-than (A (horizontal-dimension ?self)) (A (horizontal-max ?self)))))

;;;; View vocabulary

(defview (fatigue-failure ?material)
  Individuals ((?material :type metal))
  QuantityConditions ((equal-to (A (deltaK ?material)) (A (Kt ?material)))))
```

```
(defview (fracture-failure ?material)
 Individuals ((?material :type metal))
 QuantityConditions ((equal-to (A (K ?material)) (A (Kc ?material)))))

(defview (penny/surface-crack ?material)
 Individuals ((?material :type metal))
 QuantityConditions ((less-than (A (horizontal-dimension ?material))
                                (A (horizontal-max ?material)))
                     (less-than (A (vertical-dimension ?material))
                                (A (vertical-max ?material)))))

(defview (edge-crack ?material)
 Individuals ((?material :type metal))
 QuantityConditions ((equal-to (A (horizontal-dimension ?material))
                               (A (horizontal-max ?material)))))

(defview (thru-thickness-crack ?material)
 Individuals ((?material :type metal))
 QuantityConditions ((equal-to (A (vertical-dimension ?material))
                               (A (vertical-max ?material)))))

;;;; Process vocabulary

(defprocess (vertical-fatigue ?material)
 Individuals ((?material :type metal))
 QuantityConditions ((less-than (A (deltaK ?material)) (A (Kt ?material)))
                     (less-than (A (vertical-dimension ?material))
                                (A (vertical-max ?material))))
;; Relations ()
 Influences ((I+ (vertical-dimension ?material)(A (deltaK ?material)))))

(defprocess (horizontal-fatigue ?material)
 Individuals ((?material :type metal))
 QuantityConditions ((less-than (A (deltaK ?material)) (A (Kt ?material)))
                     (less-than (A (horizontal-dimension ?material))
                                (A (horizontal-max ?material))))
;; Relations ()
 Influences ((I+ (horizontal-dimension ?material)(A (deltaK ?material)))))
```

Fig. 3-14 Second Fatigue and Fracture Model in QPE

These two models (Fig. 3-13 & Fig. 3-14) represent all of the parameters (Fig. 3-10) except for temperature. Both models allow the crack to grow to either the vertical

or horizontal dimension of the structure and not fail through fatigue or fracture. However, QSIM uses three models for this type of representation. The three models represent possible crack growth, respectively in the vertical direction, the horizontal direction, or in both directions. These models are necessary because QSIM "stops" simulation when the bound of a parameter is reached. But the behavior of interest in this model is fatigue or fracture failure. Region transitions allow for discontinuous behavior to be modeled and the shifting from one model to another in QSIM (Fig. 3-15).

The use of processes and views in QPE clearly defines the cause of change in a system and reaching important limits in a model. In QSIM, these issues are all defined through the monotonic constraints and through the parameter's quantity spaces. The separation of processes and views from the model's parameters and constraints provides a uniform method of modeling.

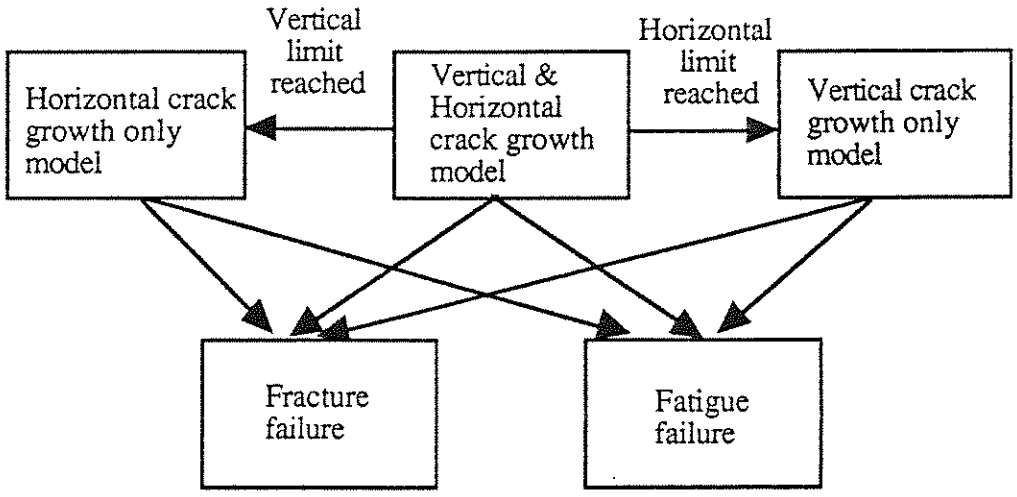

Fig. 3-15 Model transitions in QSIM

The additional parameter of temperature is added in QPE (Fig. 3-16). Temperature affects the landmark K_c in the quantity space of K. However, QSIM does not allow constraints between parameters and landmarks. Therefore, the parameter K_r (residual toughness where $K + K_r = K_c$) is introduced so that the monotonic constraint ($M+ T K_c$) can be included. However, K_c must be kept constant for actual behaviors to be simulated. The benefits of modeling K_c as a monotonic function are lost.

```
;; -*- Mode: LISP; Syntax: Common-lisp; Package: USER -*-

(in-package qpe::*user-package*)

(adb::rules-file "FF-DOMAIN")

;;;; Fatigue and Fracture Theory for QPE
;;

(defQuantity-Type vertical-dimension Individual)
(defQuantity-Type horizontal-dimension Individual)
(defQuantity-Type vertical-max individual)
(defQuantity-Type horizontal-max individual)
(defQuantity-Type deltaK Individual)
(defQuantity-Type K Individual)
(defQuantity-Type Kt individual)
(defQuantity-Type Kc individual)
(defQuantity-Type temperature individual)

(defentity metal
        ;;; Main characteristic is that it has a number of quantities
 (quantity (deltaK ?self))
 (quantity (K ?self))
 (quantity (vertical-dimension ?self))
 (quantity (horizontal-dimension ?self))
 (quantity (Kt ?self))
 (quantity (Kc ?self))
 (quantity (vertical-max ?self))
```

```
  (quantity (horizontal-max ?self))
  (quantity (temperature ?self))
  ;;; There are a few state-independent relationships
  (Qprop (deltaK ?self)(horizontal-dimension ?self))
  (Qprop (deltaK ?self)(vertical-dimension ?self))
  (Qprop (K ?self)(vertical-dimension ?self))
  (Qprop (K ?self)(horizontal-dimension ?self))
  ;;dimensions are never negative
  (not (less-than (A (vertical-dimension ?self)) zero))
  (not (less-than (A (horizontal-dimension ?self)) zero))
  ;; deltaK is not negative
  (greater-than (A (deltaK ?self)) zero)
  ;;these are all positive values
  (greater-Than (A (vertical-max ?self)) zero)
  (greater-Than (A (horizontal-max ?self)) zero)
  (greater-Than (A (Kc ?self)) zero)
  (greater-Than (A (Kt ?self)) zero)
  ;; Physical impossibilities for this model
  (not (greater-than (A (deltaK ?self)) (A (Kt ?self))))
  (not (greater-than (A (K ?self)) (A (Kc ?self))))
  (not (greater-than (A (vertical-dimension ?self)) (A (vertical-max ?self))))
  (not (greater-than (A (horizontal-dimension ?self)) (A (horizontal-max ?self)))))

;;;; View vocabulary

(defview (fatigue-failure ?material)
  Individuals ((?material :type metal))
  QuantityConditions ((equal-to (A (deltaK ?material)) (A (Kt ?material)))))

(defview (fracture-failure ?material)
  Individuals ((?material :type metal))
  QuantityConditions ((equal-to (A (K ?material)) (A (Kc ?material)))))

(defview (penny/surface-crack ?material)
  Individuals ((?material :type metal))
  QuantityConditions ((less-than (A (horizontal-dimension ?material))
                                 (A (horizontal-max ?material)))
                      (less-than (A (vertical-dimension ?material))
                                 (A (vertical-max ?material)))))

(defview (edge-crack ?material)
  Individuals ((?material :type metal))
  QuantityConditions ((equal-to (A (horizontal-dimension ?material))
                                (A (horizontal-max ?material)))))

(defview (thru-thickness-crack ?material)
  Individuals ((?material :type metal))
  QuantityConditions ((equal-to (A (vertical-dimension ?material))
```

```
                                    (A (vertical-max ?material)))))

;;;; Process vocabulary

(defprocess (vertical-fatigue ?material ?hcrack ?temp)
  Individuals ((?material :type metal
                          :conditions
                          (has-condition ?material Vcrack-varying ?hcrack ?temp)))
  QuantityConditions ((less-than (A (deltaK ?material)) (A (Kt ?material)))
                      (less-than (A (K ?material)) (A (Kc ?material)))
                      (less-than (A (vertical-dimension ?material))
                                 (A (vertical-max ?material))))
  Influences ((I+ (vertical-dimension ?material)(A (deltaK ?material)))))

(defprocess (horizontal-fatigue ?material ?vcrack ?temp)
  Individuals ((?material :type metal
                          :conditions
                          (has-condition ?material ?vcrack Hcrack-varying ?temp)))
  QuantityConditions ((less-than (A (deltaK ?material)) (A (Kt ?material)))
                      (less-than (A (K ?material)) (A (Kc ?material)))
                      (less-than (A (horizontal-dimension ?material))
                                 (A (horizontal-max ?material))))
  Influences ((I+ (horizontal-dimension ?material)(A (deltaK ?material)))))

(defprocess (temperature-effect ?material ?vcrack ?hcrack)
  Individuals ((?material :type metal
                          :conditions
                          (has-condition ?material ?vcrack ?hcrack temp-varying)))
  QuantityConditions ((less-than (A (deltaK ?material)) (A (Kt ?material)))
                      (less-than (A (K ?material)) (A (Kc ?material))))
  Influences ((I+ (Kc ?material)(A (temperature ?material)))))
```

Fig. 3-16 Third Fatigue and Fracture Domain Model in QPE

The use of the separation of the domain model and the scenario model in QPE is useful in modeling. One of the features that can be incorporated into the scenario model (Fig. 3-17) is the ability to "turn on or off" parts of the model. The fatigue and fracture scenario model can "turn on or off" the effects of vertical crack growth, horizontal crack growth, and temperature effects. This feature was only used in testing the effects of temperature without crack growth. Adding additional parameters and relationships

incrementally does not become more difficult because of the additional complexity of the results. In building more complex models, the ability to "turn on or off" certain parts of the fatigue and fracture model is useful in capturing a wide range of materials that might not have similar relationships. For instances, steel and aluminum have different behaviors [Barsom & Rolfe 1987]. The scenario model can guide the selection of the features that are used in the domain model.

```
;;; -*- Mode: LISP; Syntax: Common-Lisp; Package: USER -*-
(in-package qpe::*user-package*)
;;; Fatigue and fracture example
(assertq (metal Steel))
(assertq (has-condition Steel Vcrack-varying Hcrack-varying temp-varying))
```

Fig. 3-17 Scenario Model for Fatigue and Fracture in QPE

Chapter 4

Findings in Qualitative Simulation

4.1 Issues in Simulation

Simulation is the process of using a model to represent certain aspects of a physical system [Rothenberg 1989]. The results of simulation are generally judged by the correctness and completeness of the prediction of behaviors. Additional considerations must be made in the evaluation of qualitative simulation. Qualitative simulation can result in behaviors that are not possible in an actual physical system. The generation of erroneous behaviors needs to be limited to as few as possible and these behaviors need to be recognizable. Since the results of qualitative simulation are more general than other methods, an additional criteria is the usefulness of simulation results. The evaluation of the results of qualitative simulation will not include an extensive study of the mechanism of simulation of QPE or QSIM. The results of the U-tube example and the results of the new models created in the domain of fatigue and fracture of metals are explored. Correctness, completeness, and usefulness of the results are the criteria of evaluation.

4.1.1 Simulation results

Qualitative simulation results are organized into qualitative states and the transitions between the states. A qualitative state is defined by the values (magnitude and sign) of the parameters of the model. The transition to a set of possible successor states rather than only to a single successor state is unique to qualitative simulation.

The possible branches of behaviors are non-deterministic and are necessary to predict behaviors through knowledge about the structure of the model even though specific numeric information may be unavailable.

The results of QPE are an envisionment and the results of QSIM are a direct history generation. Envisionment is a type of qualitative simulation that is temporally generic. A possible history (one actual behavior) is the selection of a path of transitions through the qualitative states. However, [Kuipers 1986] shows that not every path is a physically realizable behavior. The path selection requires additional knowledge that is not available in a qualitative description of a model. Direct history generation chooses the transitions between states during simulation and therefore is temporally specific. However, the generation of new landmarks in QSIM can also introduce erroneous behaviors [Kuipers & Chiu 1987].

4.1.2 Using qualitative results

Qualitative results are more general than the results used by quantitative techniques. There are various ways of making use of these results. Traditional simulation methods are labor-intensive. The validation of the model involves selecting various conditions that will test all regions of interest in the model. This requires selecting the bounds of simulation and selecting correct initial conditions. Envisionments avoid this problem by generating the entire set of behaviors. Envisionments are also useful because they are a finite representation for a possibly infinite set of behaviors.

One of the uses of qualitative reasoning is to "guide" further analysis. Although incomplete quantitative knowledge is common in problems, some quantitative

information is often known. The combination of qualitative reasoning and quantitative reasoning provides a powerful technique. Qualitative reasoning provides a framework for organizing and using quantitative knowledge [Forbus & Falkenhainer 1990]. Another technique is the use of quantitative information to "refine" qualitative reasoning [Kuipers & Berleant 1988]. Both methods have a central goal of not losing available quantitative knowledge.

The most important use of qualitative simulation is to determine not only the possible behaviors of a model but the processes that cause the change. If the processes that cause change are identified, then the proper action can be taken to adjust the physical system so that the desired behavior can be achieved. Causal behavior is elucidated after simulation.

4.2 Simulation in QSIM

Qualitative results in QSIM are generated from an initial state defined at time point t_0. Time is defined as a continuous alternating sequence of time-points and time-intervals. The transitions that occur from a time-point to a time-interval are:

1) a parameter may change direction or

2) a parameter may move off a landmark.

The transitions that occur from a time-interval to a time-point are:

1) a parameter may change direction or

2) a parameter moving towards a landmark may reach it.

The change in direction of a parameter must be continuous. The direction can change from decreasing to steady, steady to decreasing, steady to increasing, and increasing to steady but can not change from decreasing to increasing or increasing to decreasing in

one time-point or one time-interval.

QSIM generates all of the possible transitions for the parameters of the model. The constraints of the model limit possible transitions of parameters to a manageable set of qualitative states. If there are a set of possible successor states, then the behavior prediction branches to all of the possible successor states. The behavior of the system is represented by graphs of the parameters of the model. Each parameter is plotted versus time.

4.2.1 The U-tube example

The results of a simulation of the U-tube model (Fig. 3-2) are given in (Fig. 4-1 through Fig. 4-3). The initial condition of the model is that tank A is full and tank B is empty. The first behavior of (Fig. 4-1) is the case where tank A is greater than tank B and tank B fills until it bursts. During the time interval of t_0 to t_1, amount B is moving towards the landmark Bmax. Amount B reaches the landmark Bmax at time-point t_1. A region transition occurs at Bmax to a model in which tank B bursts. The rest of the fluid from tank A flows out the side of tank B until tank A is also empty.

The second behavior (Fig. 4-2) is the case where the fluid from tank A flows to tank B until an equilibrium state is reached. Tank A and tank B reach equilibrium below the maximum capacity of the tanks. The last behavior (Fig. 4-3) is the case where the fluid from tank A flows to tank B and the equilibrium state is reached exactly at the maximum level of Bmax.

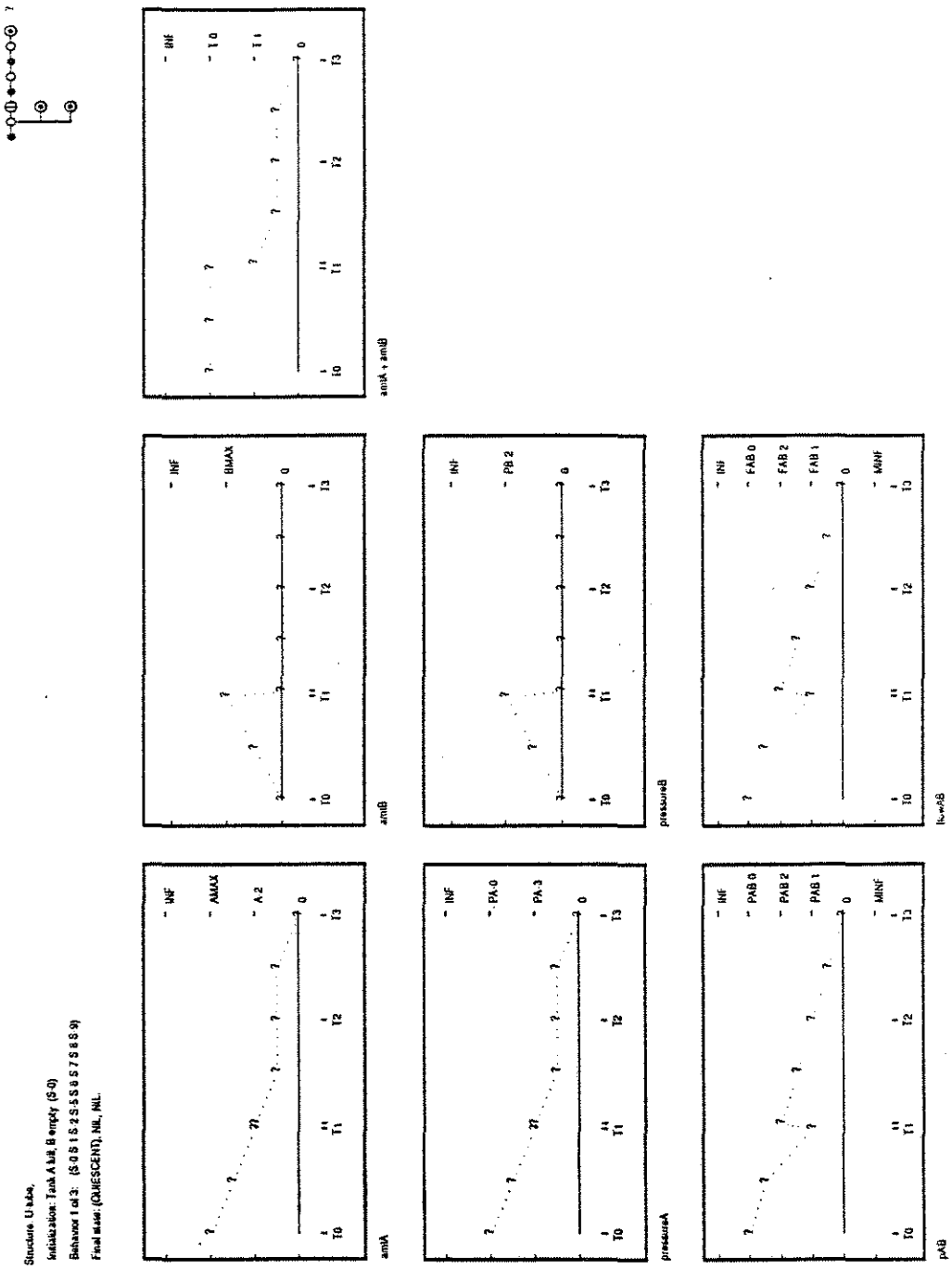

Fig. 4-1 Behavior 1 for U-tube example in QSIM

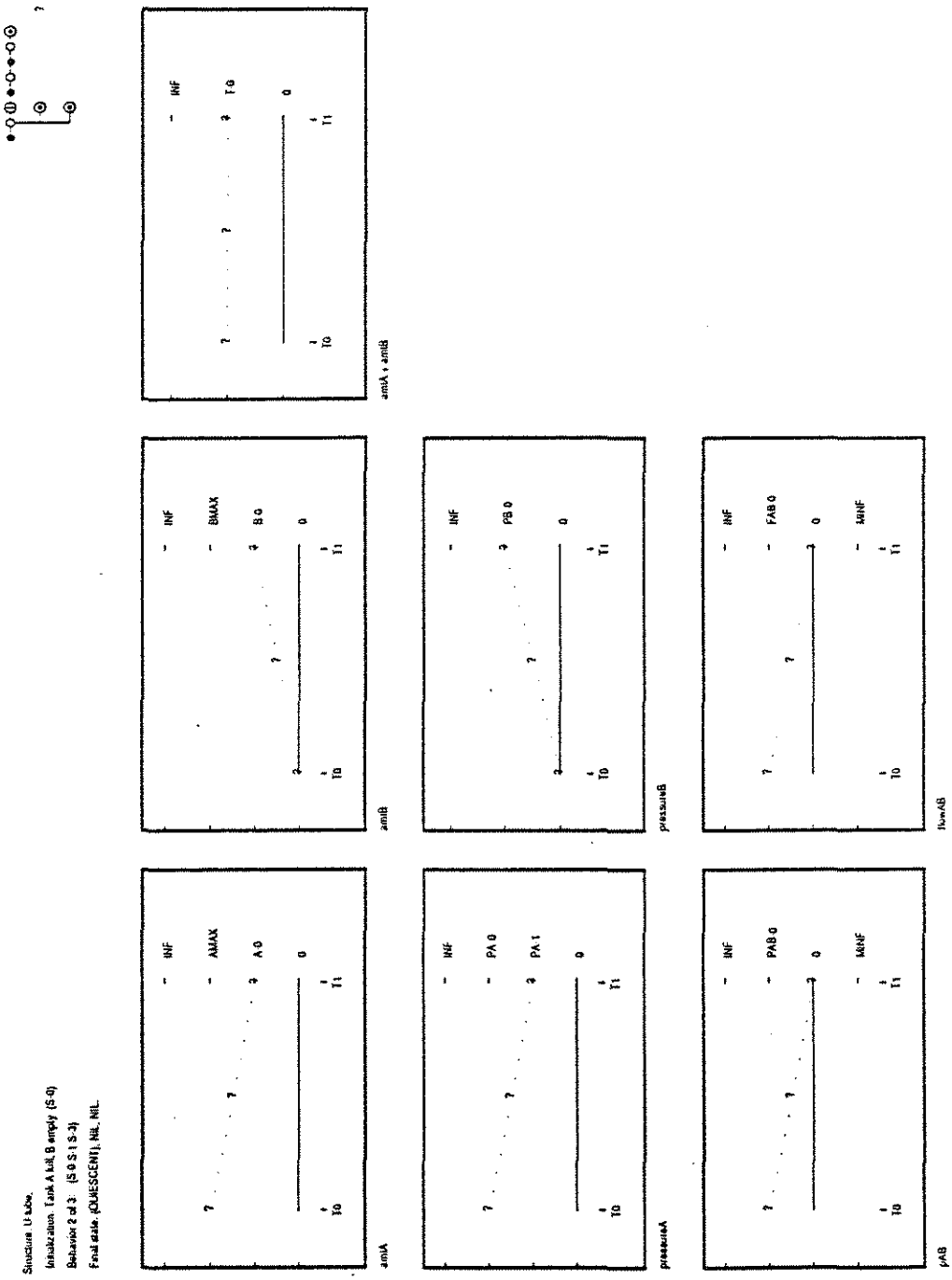

Fig. 4-2 Behavior 2 for U-tube example in QSIM

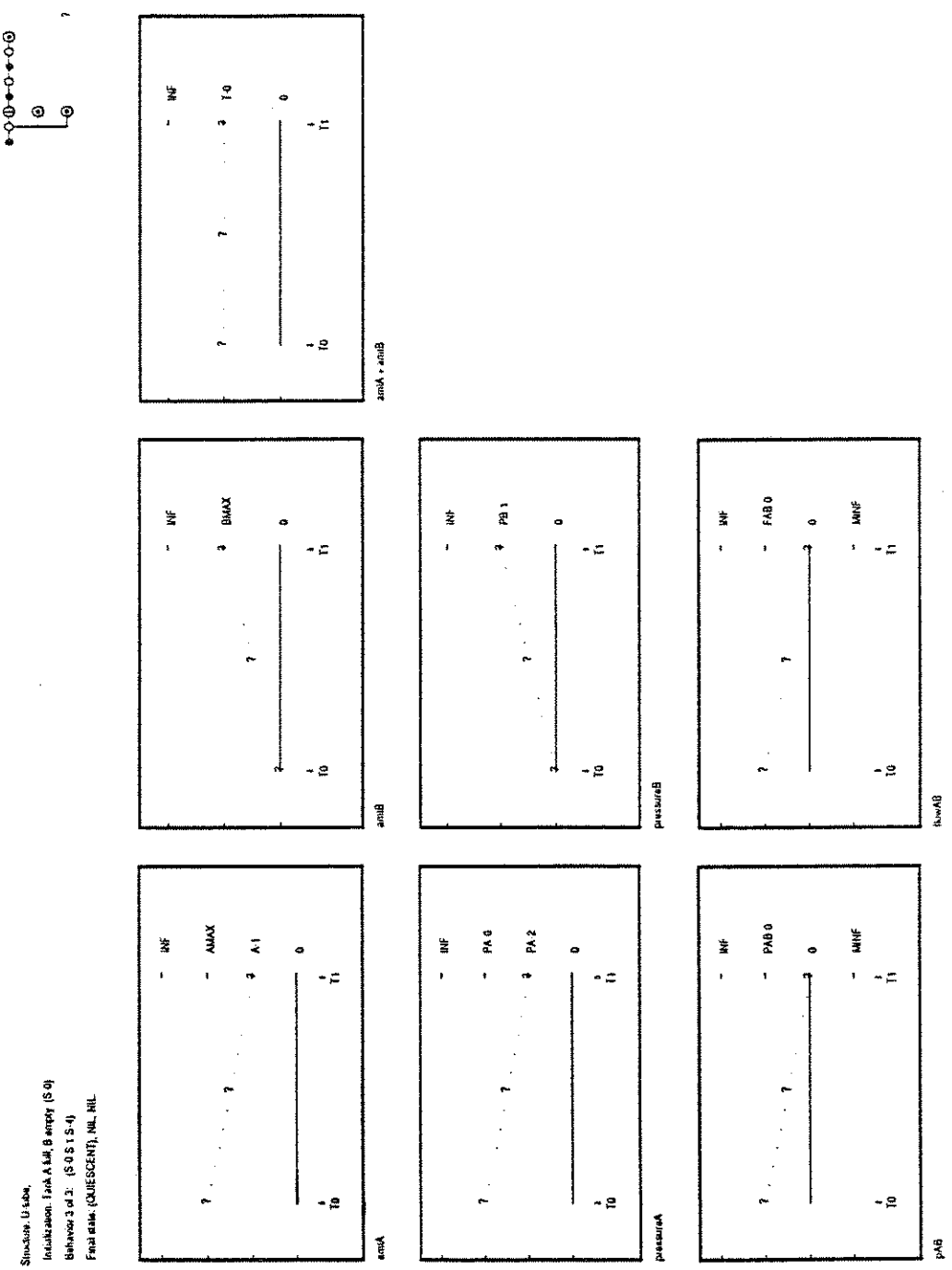

Fig. 4-3 Behavior 3 for U-tube example in QSIM

4.2.2 Landmarks

In the second behavior of the U-tube, additional landmarks are found for the equilibrium amount of fluid in the tanks. The amount of fluid in tank A and tank B reach a final value between zero and the top of the tank. These new landmarks identify new important values in the quantity space for these parameters. New landmarks are also useful in distinguishing between degenerative oscillations, stable oscillations, and increasing oscillations. Without the labeling of new landmarks, the corresponding value for stable oscillations must be specified during modeling.

The labeling of new landmarks adds a layer of complexity to the simulation results. The addition of some landmarks are unnecessary. For instances, the labeling of the new landmarks (PAB-1 & PAB-2) for the pressure difference (pAB) between tank A and tank B for (Fig. 4-1) are uninteresting. The labeling is not necessary in this example because the knowledge that the pressure decreases to zero is enough to determine the general behavior of the model. Also, the insertion of landmarks in decreasing oscillatory behavior and in "wandering" parameters is possibly infinite [Kuipers & Chiu 1987]. The problem of exponential landmark introduction leads to incorrect histories.

4.3 Simulation in QPE

The modeling ability of QPE adds additional complexity to the task of simulation (compared to QSIM). QPE takes a domain model and a scenario model as inputs and predicts the possible behaviors of the scenario through an envisionment. The basic architecture of the system is that the qualitative reasoning system (QPE) makes inferences and an assumption-based truth maintenance system (ATMS)

maintains the inferences. The ATMS reduces search space for inferences that are necessary in making the total envisionment.

The basic outline of the simulation task is determination of active processes and views, influence resolution, and limit hypothesis. Determining active processes and views requires expanding the scenario model. It is possible and likely that the domain model contains many kinds of processes and views which may or may not be valid in the specific scenario. After the scenario model is expanded, the views and processes are organized into consistent sets. For instance, in the U-tube example, either the fluid flow is from tank A to tank B or the fluid flow is from tank B to tank A but the fluid flow can't be in both directions. The next step is influence resolution. First, unambiguous influences are resolved. Unambiguous influences are cases in which all direct and indirect influences are known. Direct influences are additive but indirect influences are not. There are various methods of resolving ambiguous influences in QPE but the methods all depend on determining dependencies between quantities and making assumptions about relative magnitudes of quantities. The last step is limit hypothesis in which the transitions of the qualitative state are determined by quantity comparisons.

The results of the envisionment show all possible qualitative states and the transitions between the various states. Qualitative states are divided into s-classes which are equivalence classes of environments organized by derivative values. The transitions are described by the value of the changing quantities before and after the limit hypothesis.

4.3.1 The U-tube example

The results of the U-tube model (Fig. 3-9) are described by five s-classes and four limit hypotheses. The full textual envisionment is given in Appendix A but an interpretation of the envisionment is given next.

S-class (0) contains two environments in which both tanks are empty. In the first environment, the fluid connections is open and in the second, the fluid connection is closed.

S-class (1) contains two environments in which tank F contains some fluid but tank G is empty. Again, the fluid connections is open in one environment and closed in the other environment.

S-class (2) contains two environments in which tank G contains some fluid but tank F is empty.

Again, the fluid connection is open in one environment and closed in the other environment.

S-class (3) contains one environment (Env-226) in which tank G contains more fluid than tank F.

S-class (4) contains four environments. The first three environments have a closed fluid path. They are respectively, tank G contains more fluid than tank F, the amount of fluid is equal in the tanks, and tank F contains more fluid than tank G. The final environment (Env- 227) is where the fluid in the two tanks is equal and the fluid connections is open.

S-class (5) contains one environment (Env-228) in which tank F contains more fluid than tank G.

Limit Hypothesis (0): Initially, the pressure in tank F is less than tank G and then the pressure between the tanks becomes equal.

Limit Hypothesis (1): Initially, the pressure in tank G is less than tank F and then the pressure between the tanks becomes equal.

Limit Hypothesis (2): Initially, the amount of fluid in tank G is greater than zero and then the fluid in tank G equals zero.

Limit Hypothesis (3): Initially, the amount of fluid in tank F is greater than zero and then the fluid in tank F equals zero.

Possible transitions are: Env-226 to Env- 227 by limit hypothesis (0) or Env- 228 to Env- 227 by limit hypothesis (1).

A graphical description can greatly facilitate understanding of the envisionment. A graphical description of the U-tube (similar to other machine-independent versions) is given in (Fig. 4-4).

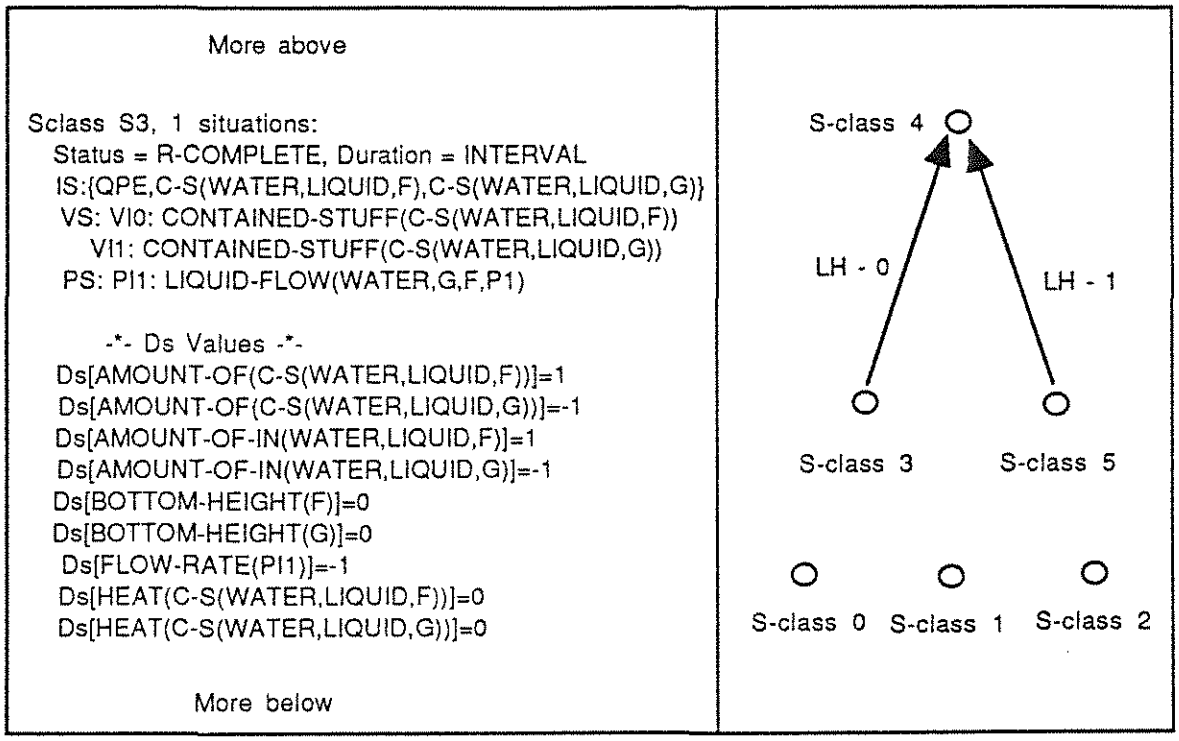

Fig. 4-4 Graphical version of U-tube Envisionment

4.3.2 Total Envisionments

QPE finds a total envisionment; an envisionment from all possible states. An initial state is needed for QSIM. The disadvantage of total envisionments are that in complex models, they can become intractable or unnecessary because additional information is known that can prune out branches in the behavior tree. But one of the distinct advantages of qualitative over quantitative simulation is the ability to simulate over the entire qualitative range. Also, when building the domain model, a total envisionment checks the model automatically for completeness. The tradeoff between completeness and efficiency is a major issue. However, the problem is not solely

dependent on total envisionment versus initial state simulation. The efficiency issue can be overcome with improvements in qualitative simulators and possibly by a multi-layered or incremental approach to simulation.

4.4 Simulation results of fatigue and fracture in metals

The results of simulation in QSIM and QPE show the possible sequence of events that occur which cause fatigue or fracture failure. There are three geometric crack growth situations for the model (four views in Fig. 4-5).

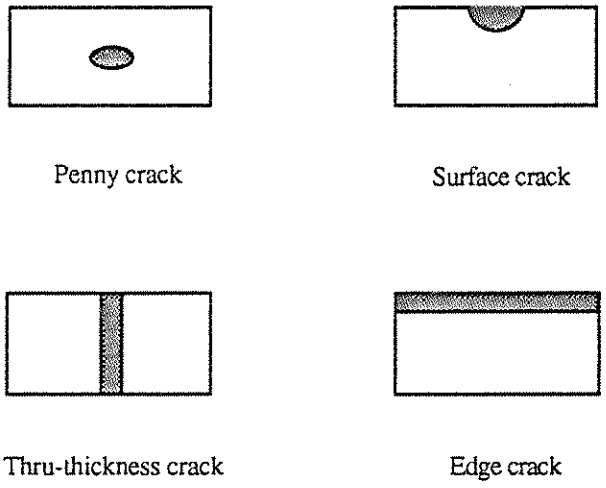

Fig. 4-5 Geometric crack growth situations

The first case is a penny or surface crack. QSIM and QPE recognizes this case when the vertical and horizontal growth of the crack is not equal to the dimensions of the structure. In QPE, this case is recognized by the penny/surface crack view being active. The possible transitions from this case are

1) Fatigue or fracture failure;

2) The crack becomes a thru-thickness crack;

3) The crack becomes an edge crack.

The second case is a thru-thickness crack which is recognized when the vertical length of the crack is equal to the vertical dimension of the structure. Again, the thru-thickness crack view is active in QPE. Possible transitions for a thru-thickness crack are

1) Fatigue or fracture failure;

2) The crack also becomes an edge crack.

The third case is an edge crack. Similarly, it is recognized when the horizontal length of the crack is equal to the horizontal dimension of the structure and the edge crack view is active in QPE. Possible transitions for an edge crack are

1) Fatigue or fracture failure;

2) The crack also becomes a thru-thickness crack.

Crack growth causes possible fatigue or fracture failure in any of the three cases.

QSIM generates 16 possible behaviors (Fig. 4-6 - Fig. 4-21) for the fatigue and fracture model (Fig. 3-13). The behaviors are

1) Fracture failure has occurred at the initial state (penny/surface crack);

2) The penny/surface crack becomes an edge crack and then fatigue failure occurs;

3) The penny/surface crack becomes an edge crack and then fracture failure occurs;

4) The penny/surface crack becomes an edge crack and then fatigue and fracture failure occurs simultaneously;

5) The penny/surface crack becomes a thru-thickness crack and then fatigue failure occurs;

6) The penny/surface crack becomes a thru-thickness crack and then fracture failure occurs;

7) The penny/surface crack becomes a thru-thickness crack and then fatigue and fracture failure occurs simultaneously;

8) Fatigue failure occurs after one time interval (penny/surface crack);

9) The penny/surface crack becomes an edge crack and fatigue failure occurs simultaneously;

10) The penny/surface crack becomes a thru-thickness crack and fatigue failure occurs simultaneously;

11) Fracture failure occurs after one time interval (penny/surface crack);

12) The penny/surface crack becomes an edge crack and fracture failure occurs simultaneously;

13) The penny/surface crack becomes a thru-thickness crack and fracture failure occurs simultaneously;

14) Fatigue and fracture failure occurs simultaneously after one time interval (penny/surface crack);

15) The penny/surface crack becomes an edge crack and fatigue and fracture failure occurs simultaneously;

16) The penny/surface crack becomes a thru-thickness crack and fatigue and fracture failure occurs simultaneously.

The simulation in QSIM determines the combination of events that occur. There are singular events such as fatigue failure occurring after one time interval and there are

combinations of the penny/surface crack becoming an edge crack and undergoing fatigue and fracture failure simultaneously. All of these events are possible although the probability of some of the behaviors actually occurring might be very low.

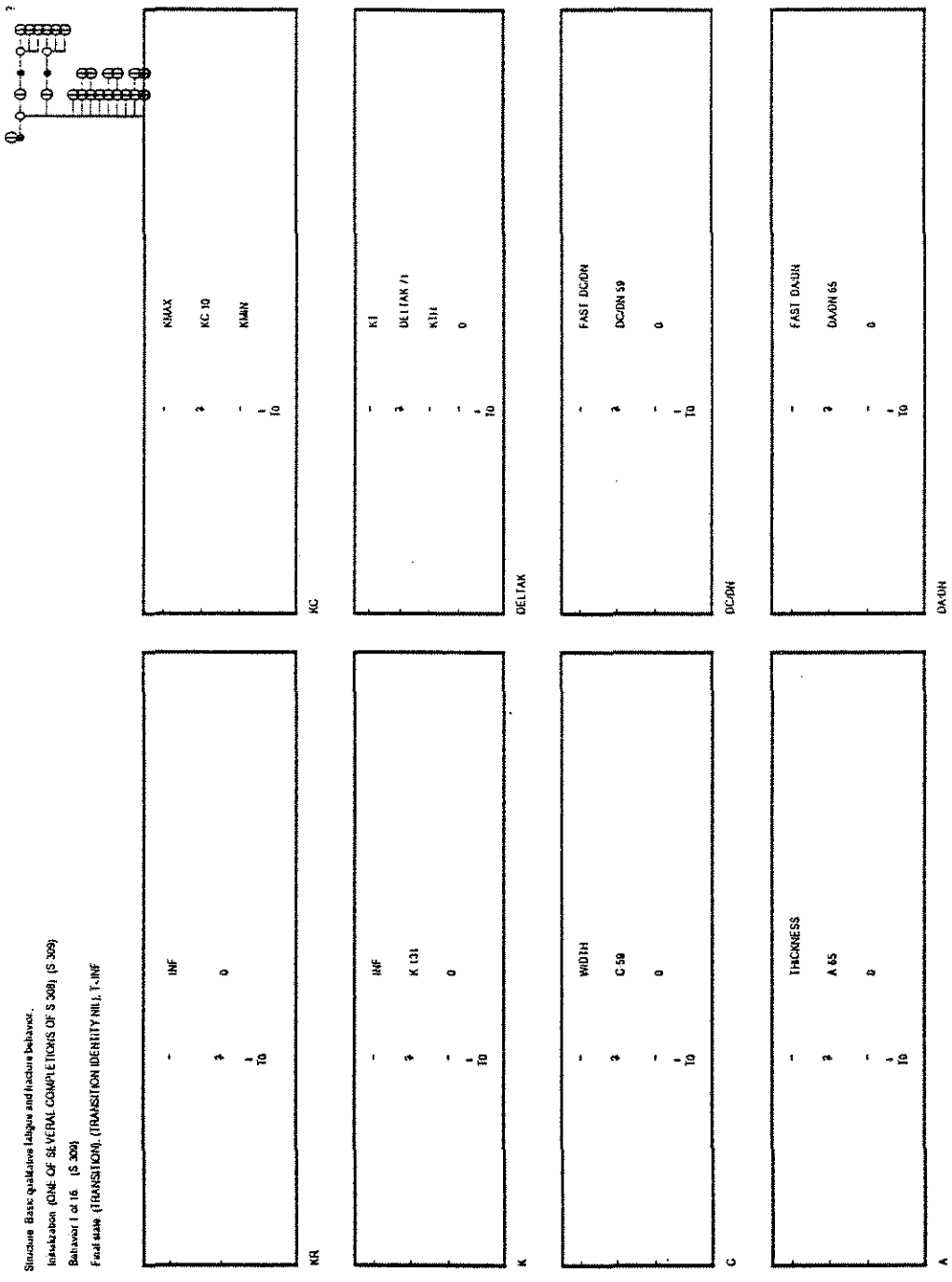

Fig. 4-6 Behavior 1 for the fatigue and fracture example in QSIM

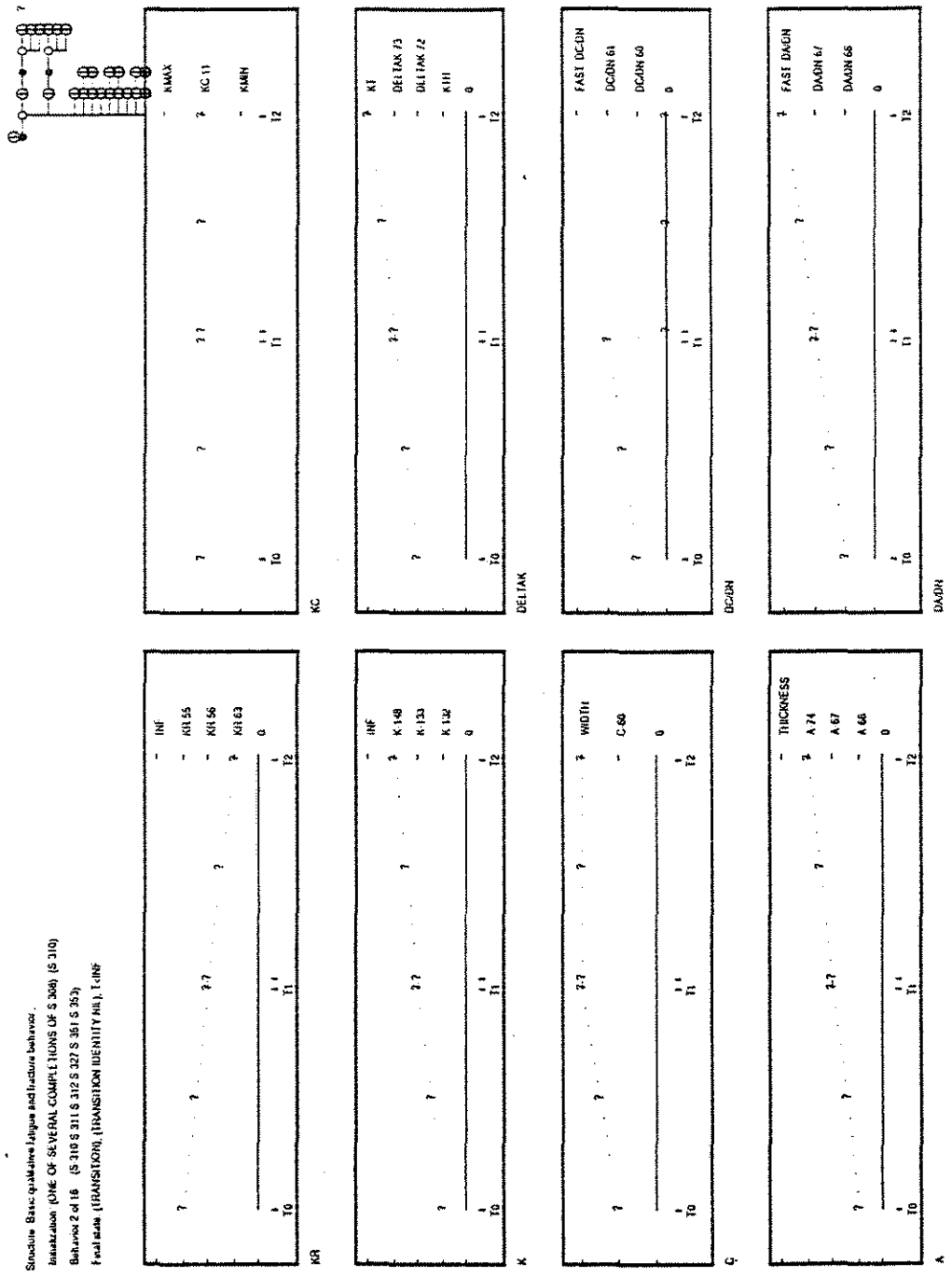

Fig. 4-7 Behavior 2 for the fatigue and fracture example in QSIM

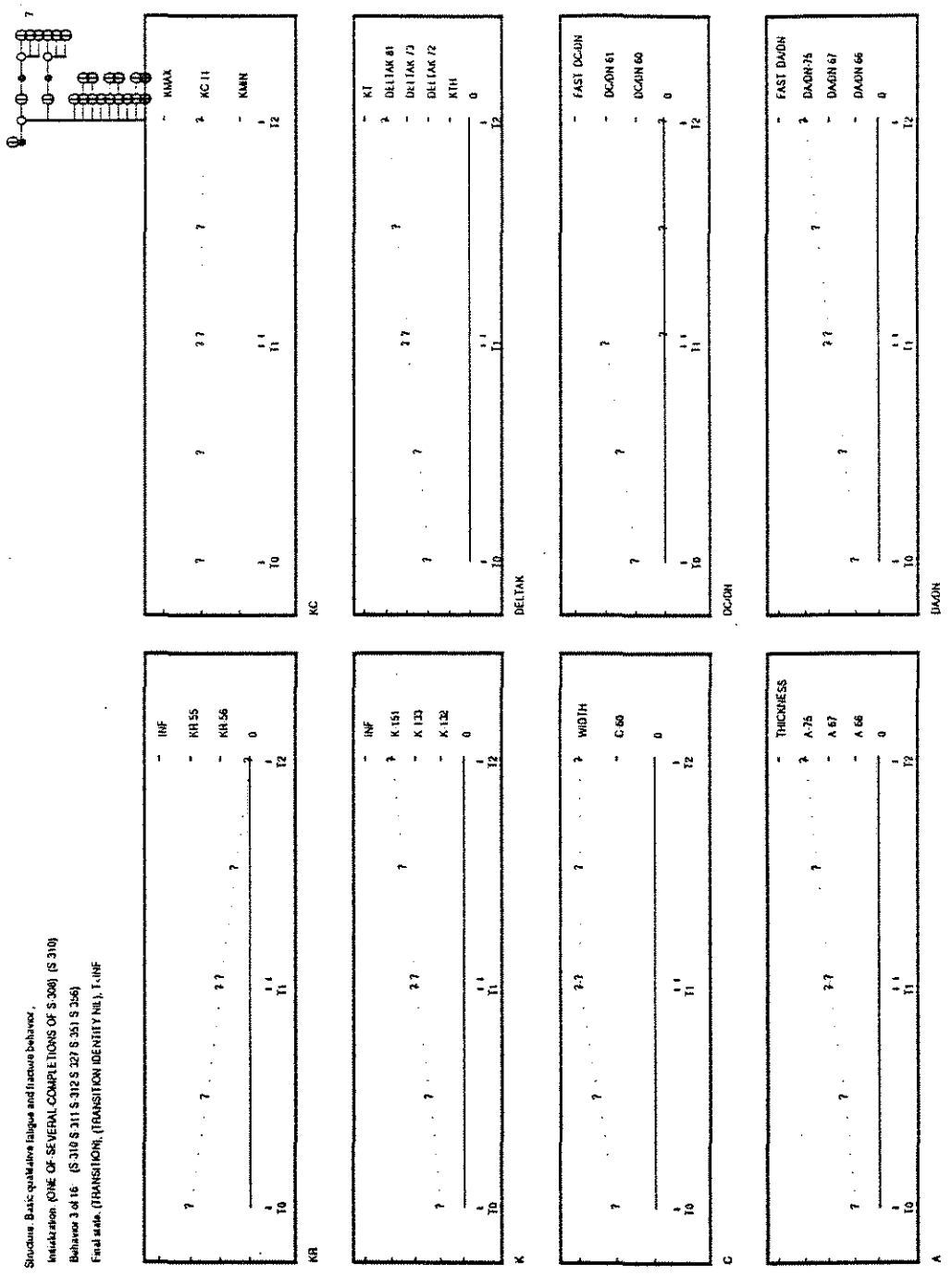

Fig. 4-8 Behavior 3 for the fatigue and fracture example in QSIM

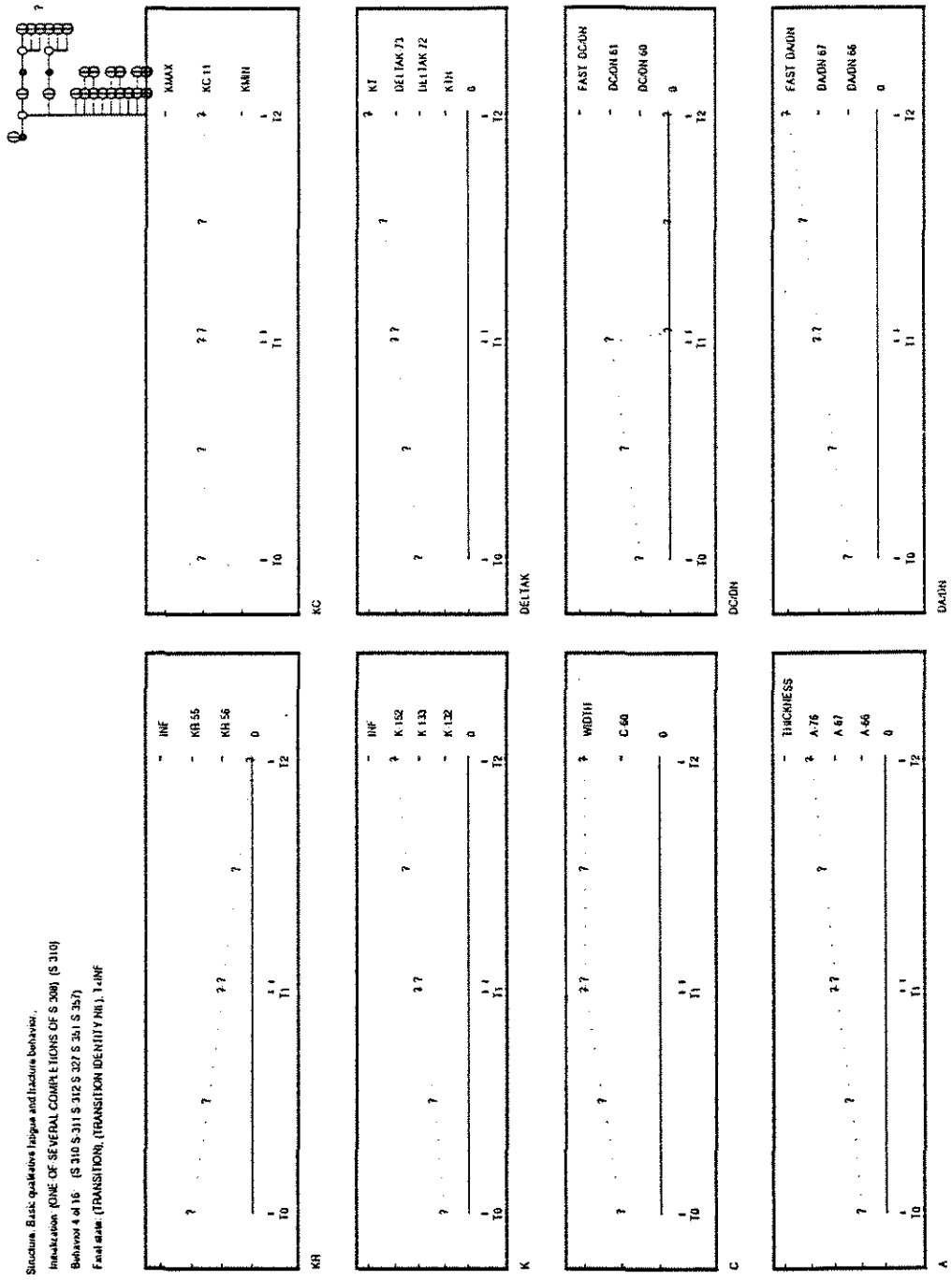

Fig. 4-9 Behavior 4 for the fatigue and fracture example in QSIM

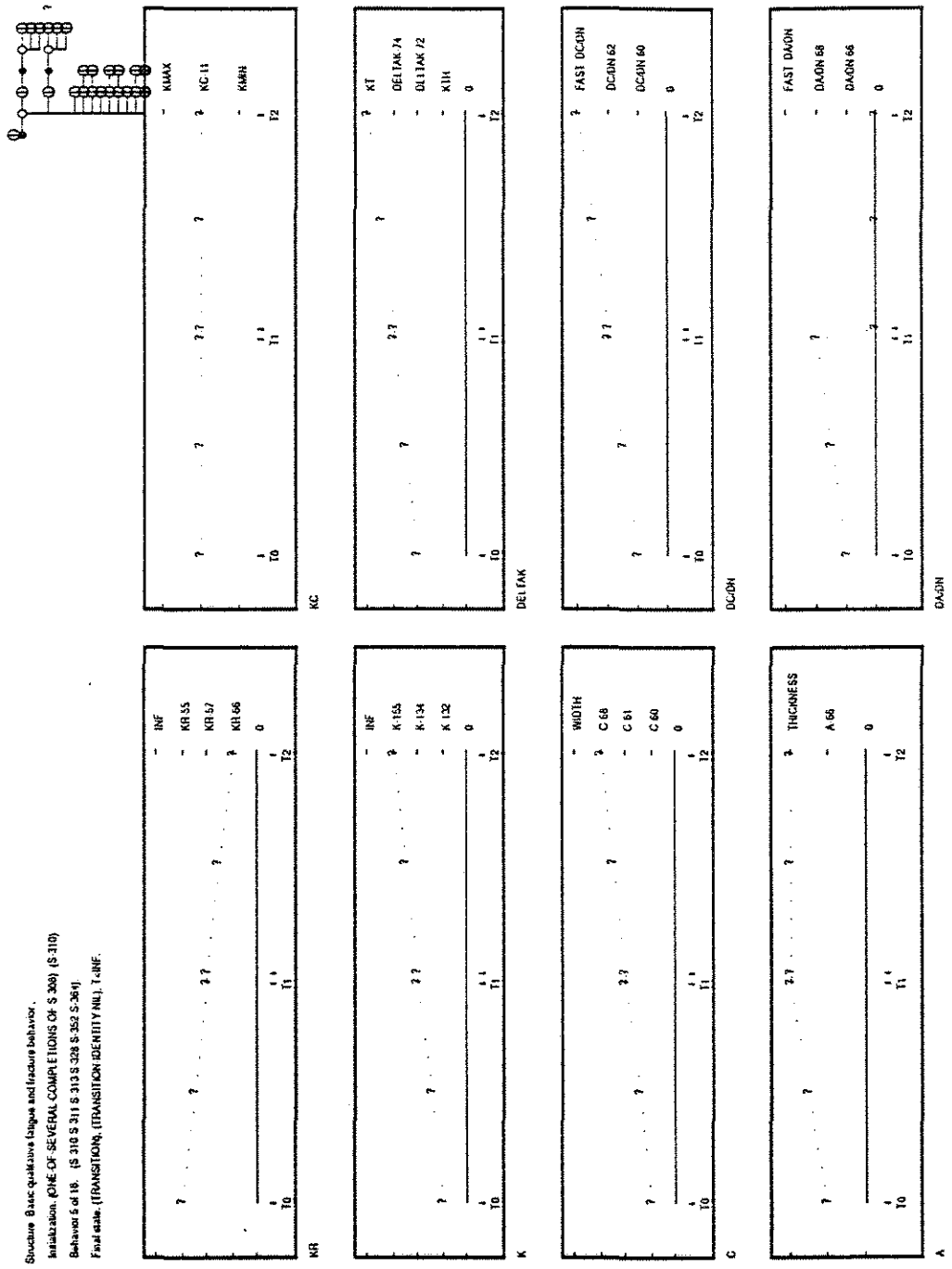

Fig. 4-10 Behavior 5 for the fatigue and fracture example in QSIM

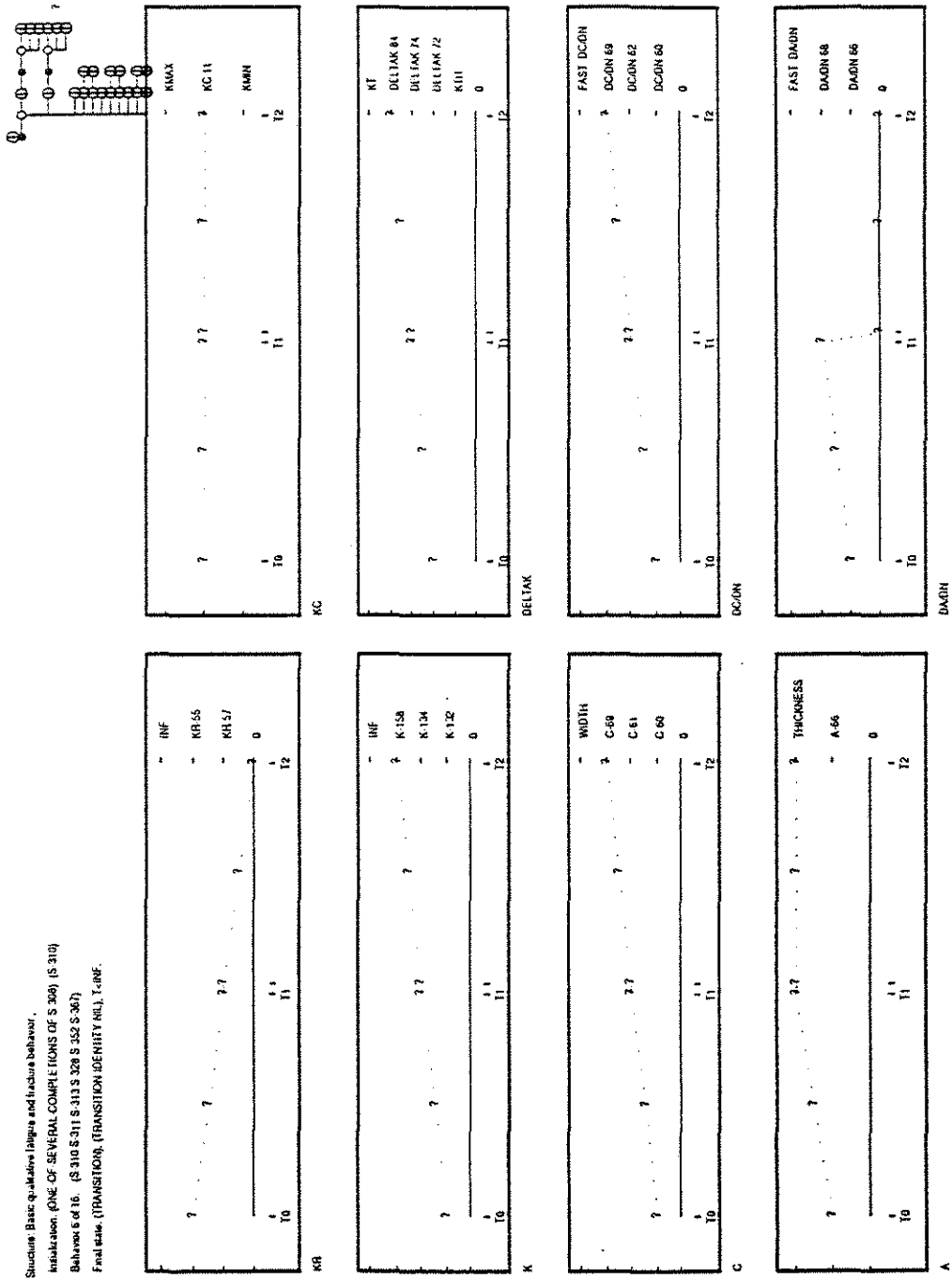

Fig. 4-11 Behavior 6 for the fatigue and fracture example in QSIM

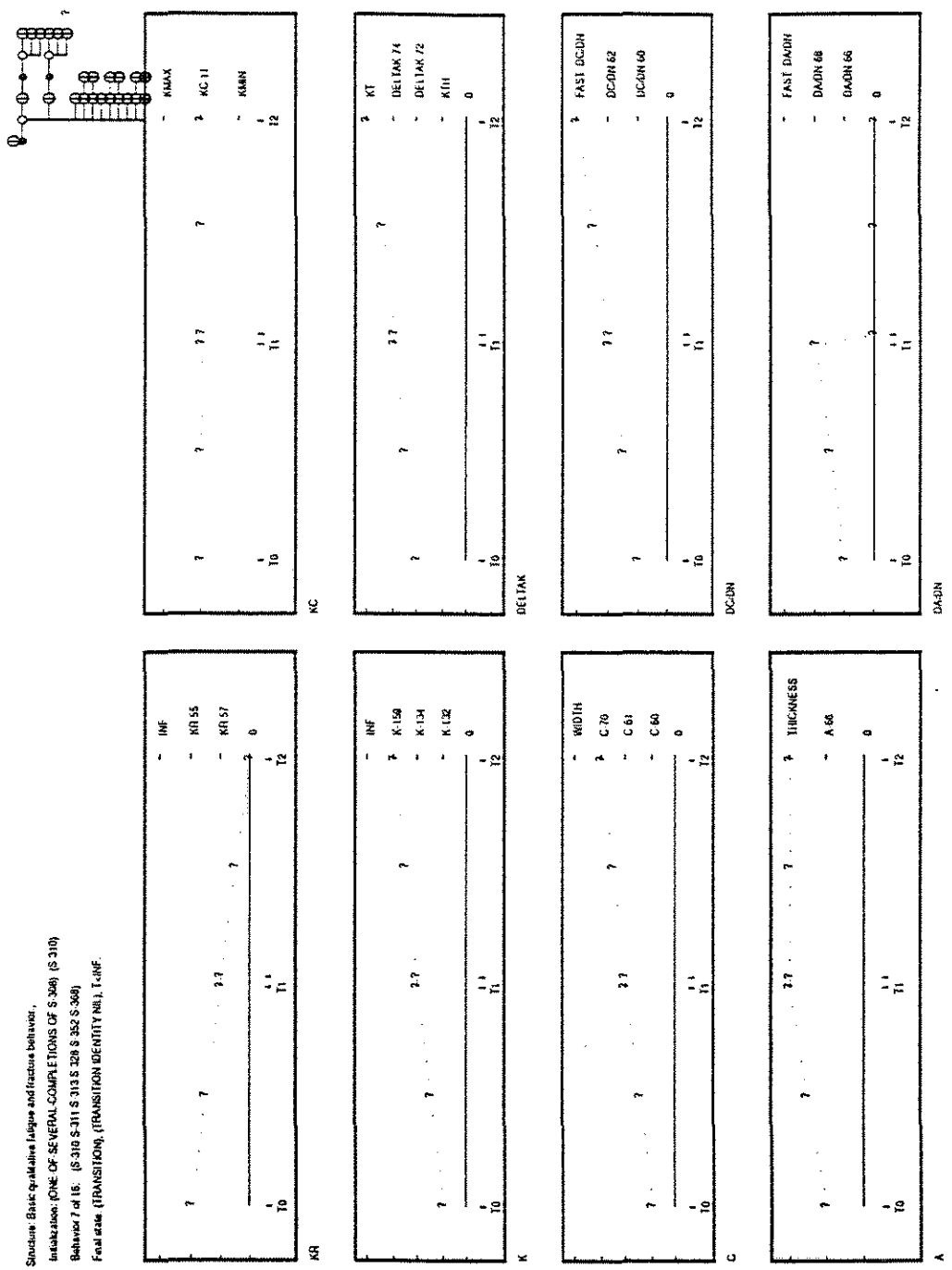

Fig. 4-12 Behavior 7 for the fatigue and fracture example in QSIM

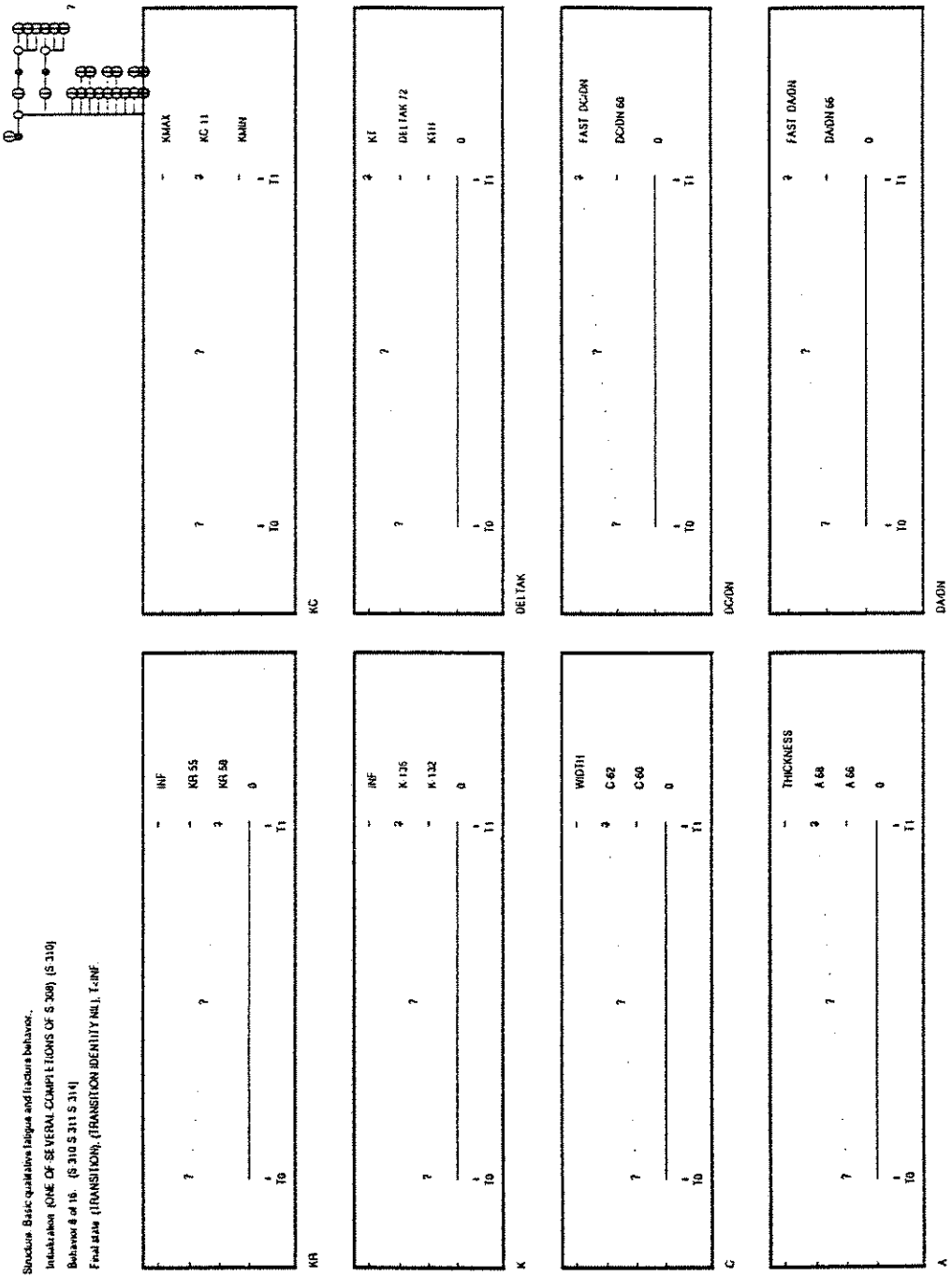

Fig. 4-13 Behavior 8 for the fatigue and fracture example in QSIM

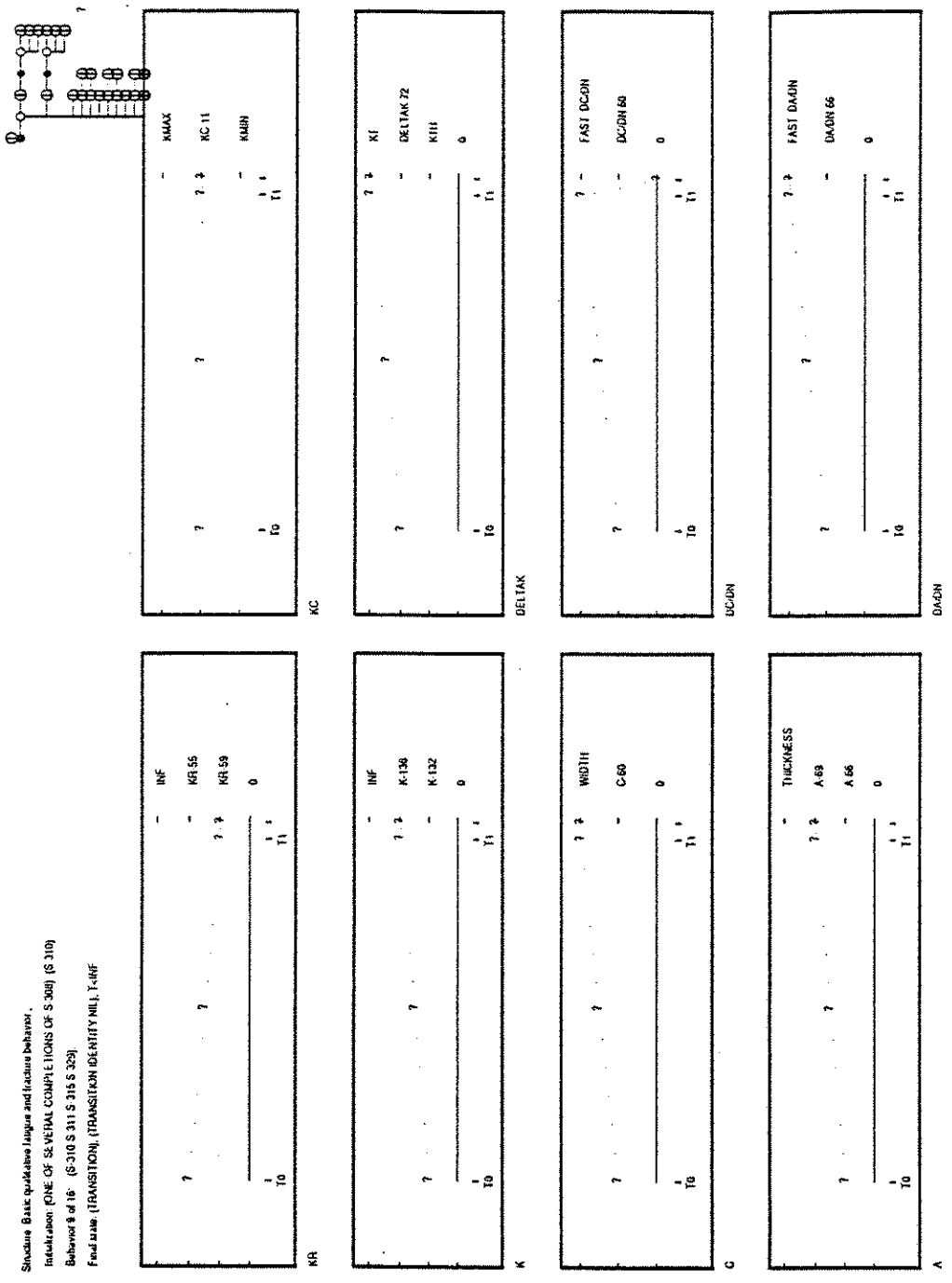

Fig. 4-14 Behavior 9 for the fatigue and fracture example in QSIM

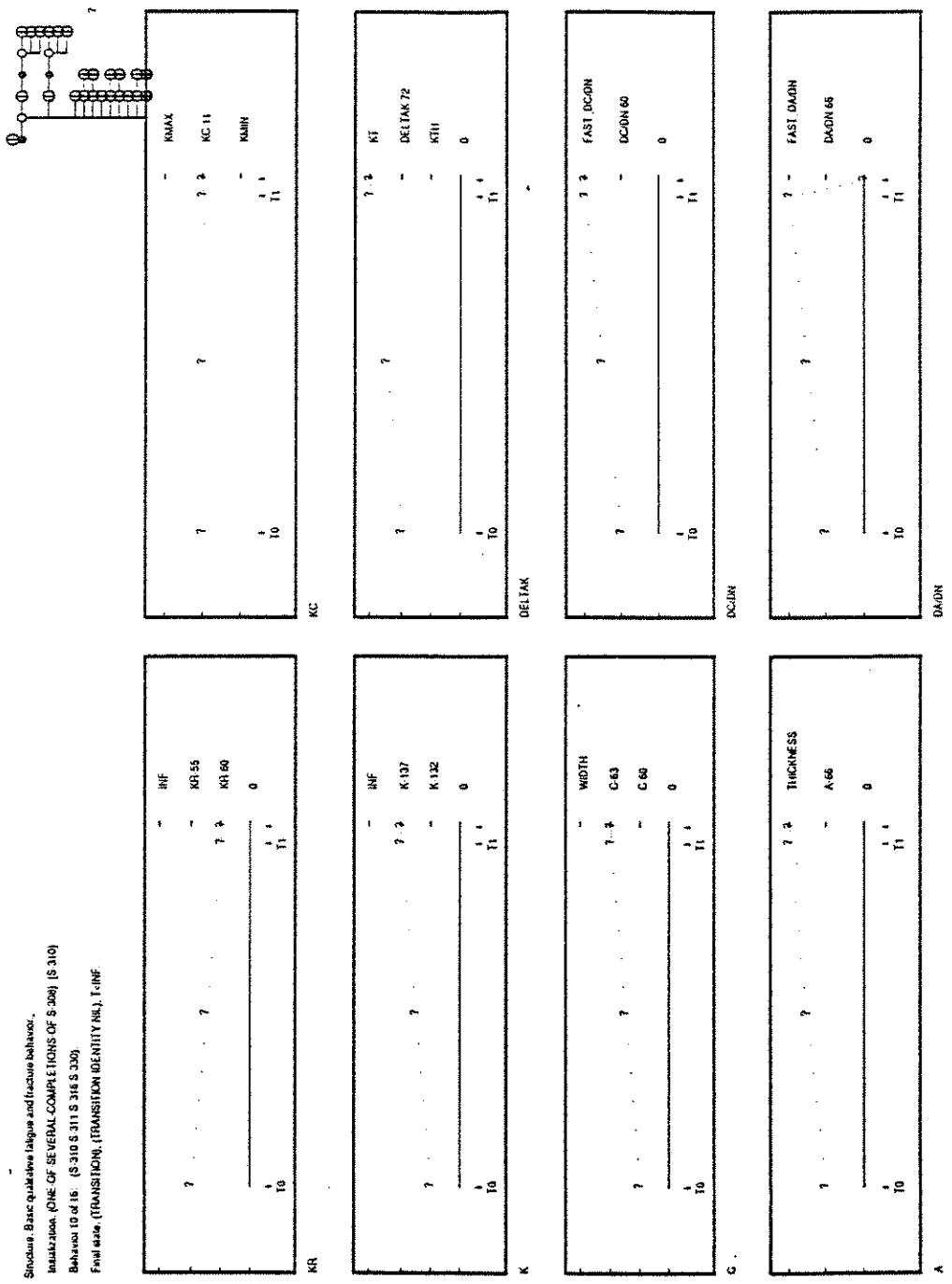

Fig. 4-15 Behavior 10 for the fatigue and fracture example in QSIM

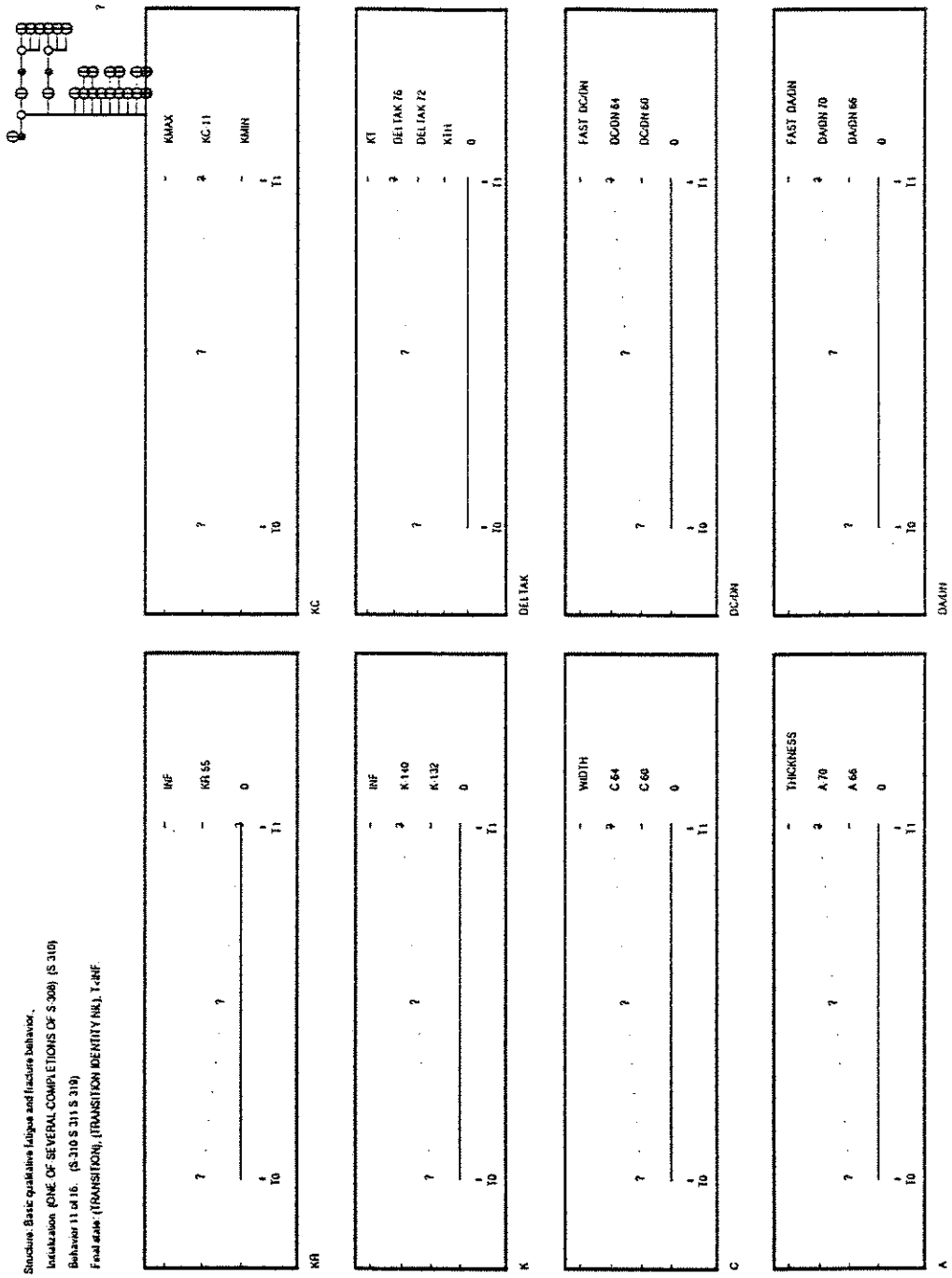

Fig. 4-16 Behavior 11 for the fatigue and fracture example in QSIM

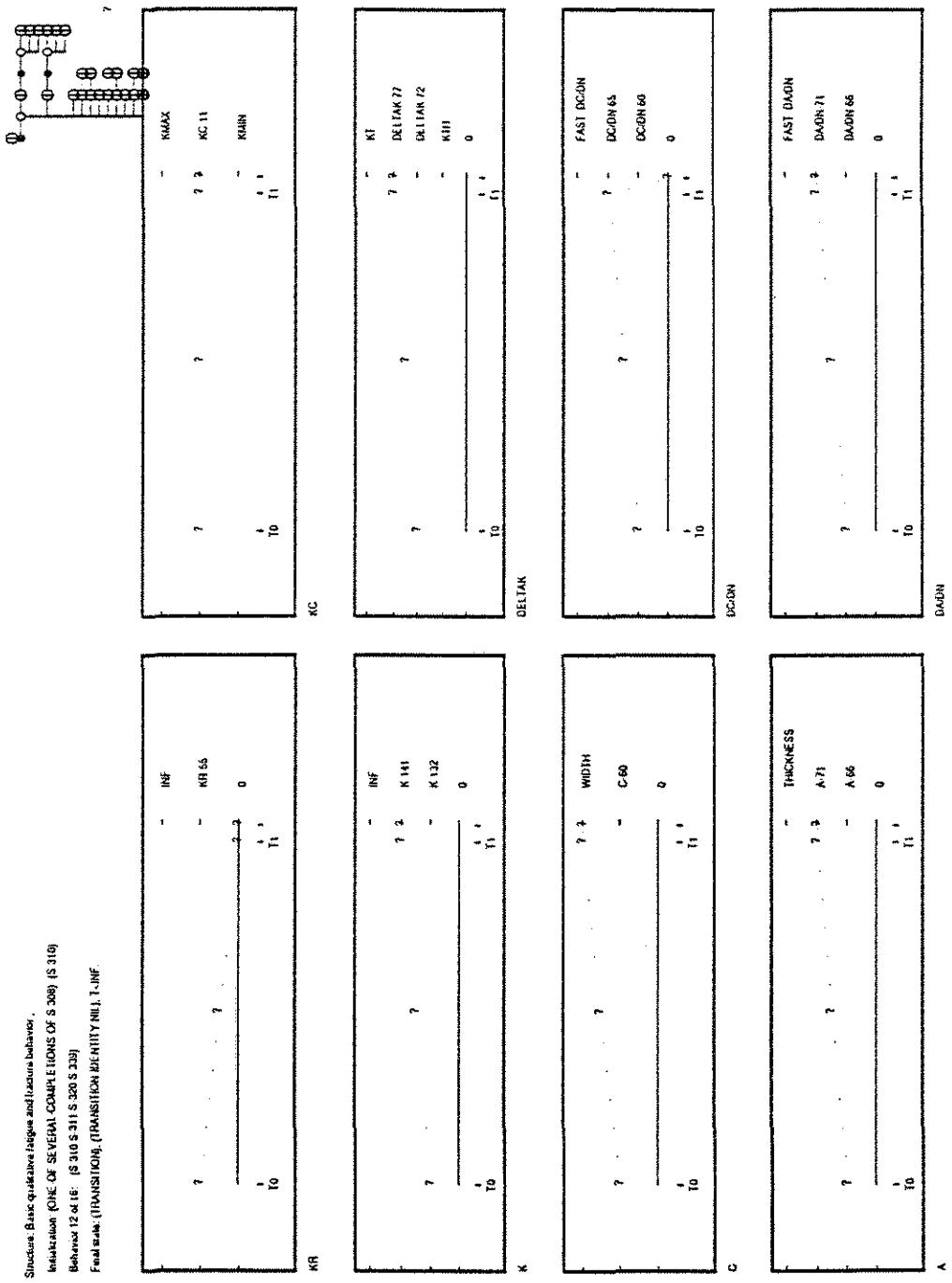

Fig. 4-17 Behavior 12 for the fatigue and fracture example in QSIM

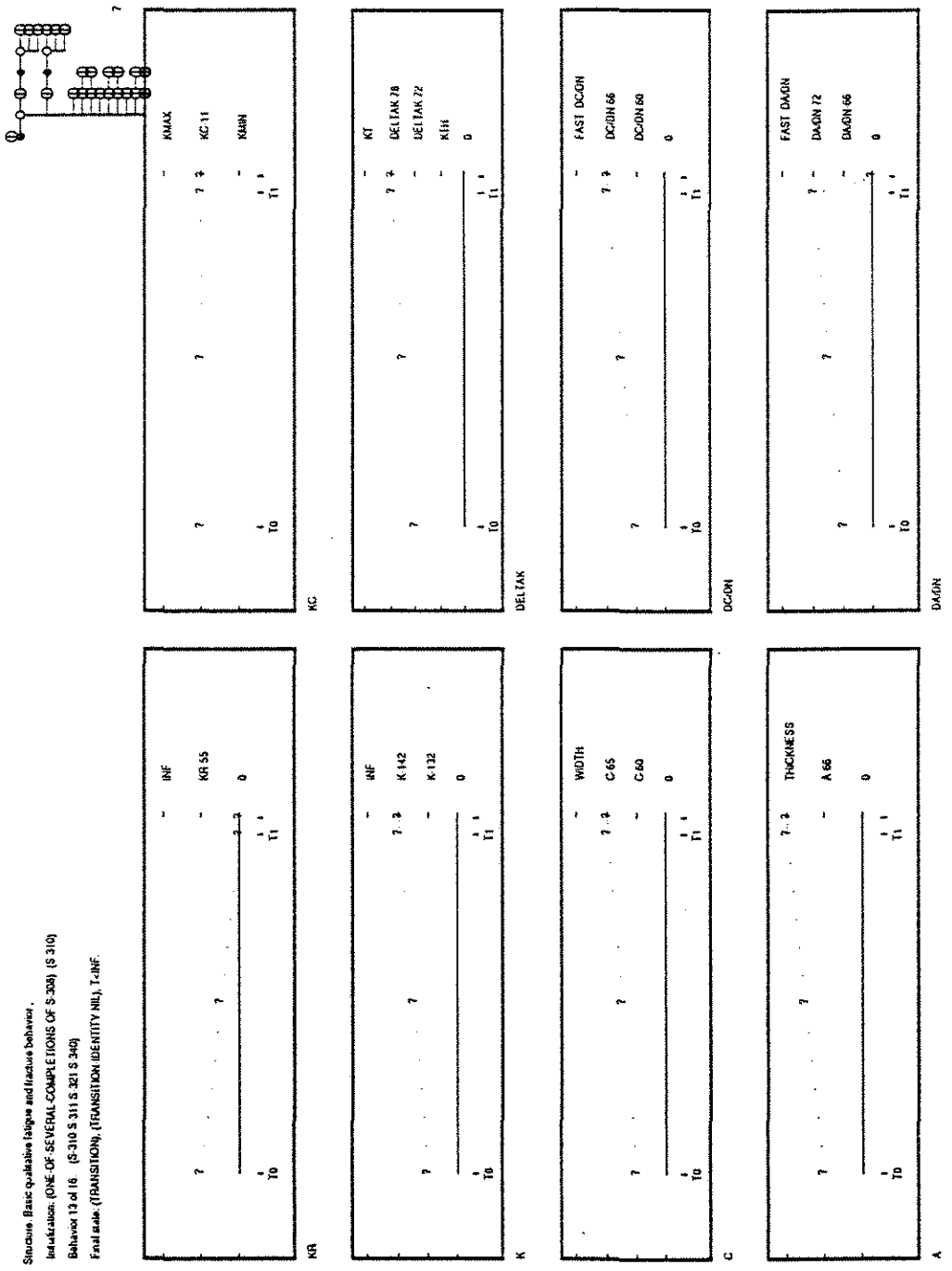

Fig. 4-18 Behavior 13 for the fatigue and fracture example in QSIM

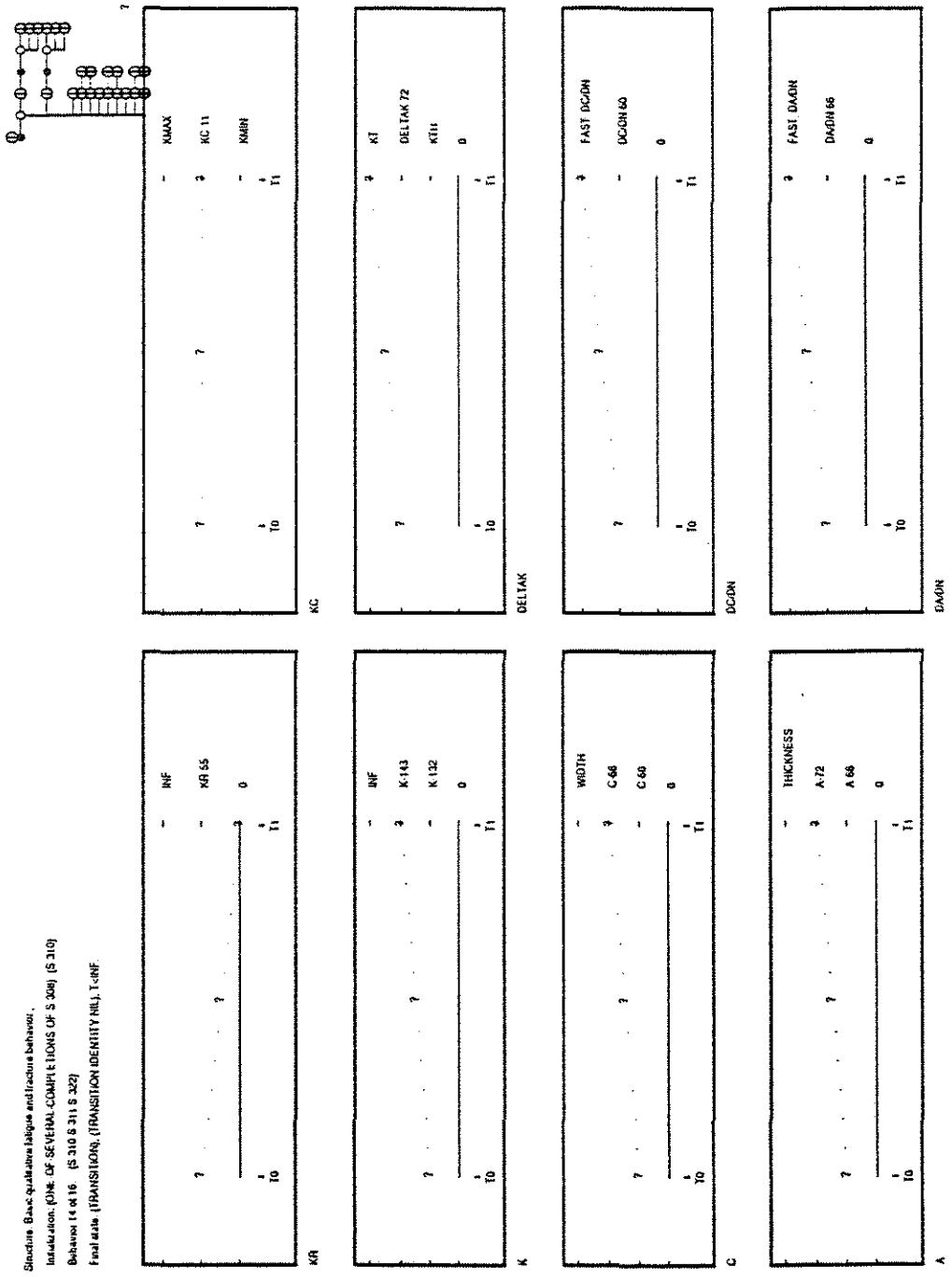

Fig. 4-19 Behavior 14 for the fatigue and fracture example in QSIM

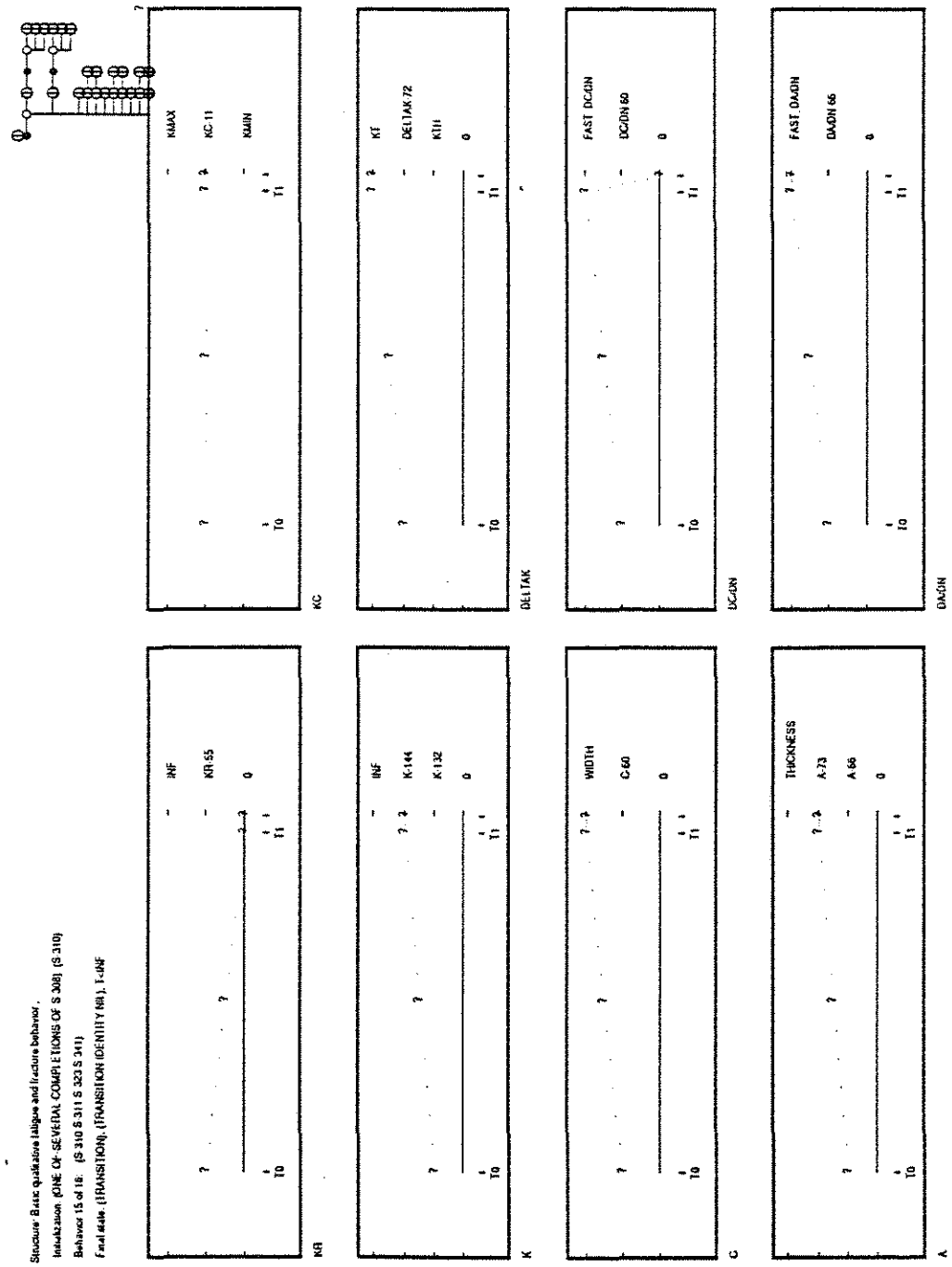

Fig. 4-20 Behavior 15 for the fatigue and fracture example in QSIM

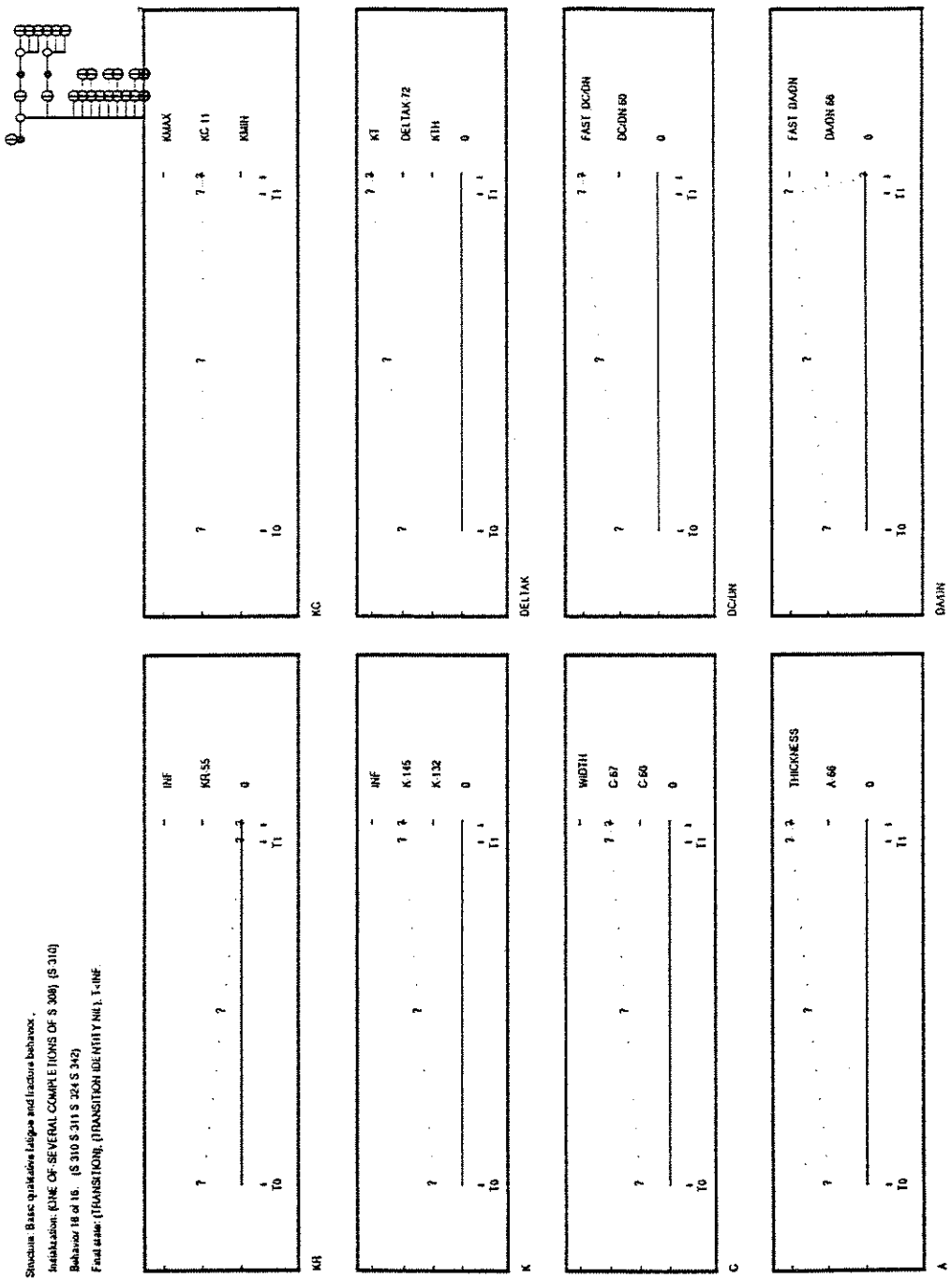

Fig. 4-21 Behavior 16 for the fatigue and fracture example in QSIM

The results of the envisionment produced by QPE for the fatigue and fracture model with temperature considerations (Fig. 3-16) are given in Appendix B. An interpreted version of the textual envisionment is given in (Fig. 4-22). The processes that cause the transitions between the qualitative states are the labels of the arrows. Although the temperature process should be labeled on the transitions (arrows), it is labeled in the qualitative state blocks to simplify the diagram. It is possible for a qualitative state to be in more than one block. For instances, the combinations of penny/surface fatigue failure block is a subset of the combinations of penny/surface fatigue and fracture failure block.

The fatigue failure blocks in the second part of (Fig. 4-22) show that the temperature causes K_c to increase so that $K = K_c$ is never true. However, temperature causes K_c to rise only over a limited range and not over an unlimited range (as modeled). Although the temperature effect is not a very good approximation for the actual behavior, it does raise some interesting issues. The temperature effect is modeled through an influence that states the derivative of K_c is qualitatively proportional to the ambient temperature. A better approximation is that temperature affects K_c over a limited range and this is modeled through an influence that is valid over a limited range. Thus, it is clear that the temperature effects must be modeled through the use of a process in which its influences are valid over a limited range (as defined by the quantity conditions of the process).

An actual history corresponds to the selection of the transitions between the qualitative states. In the fatigue and fracture model, all transitions lead to actual behaviors.

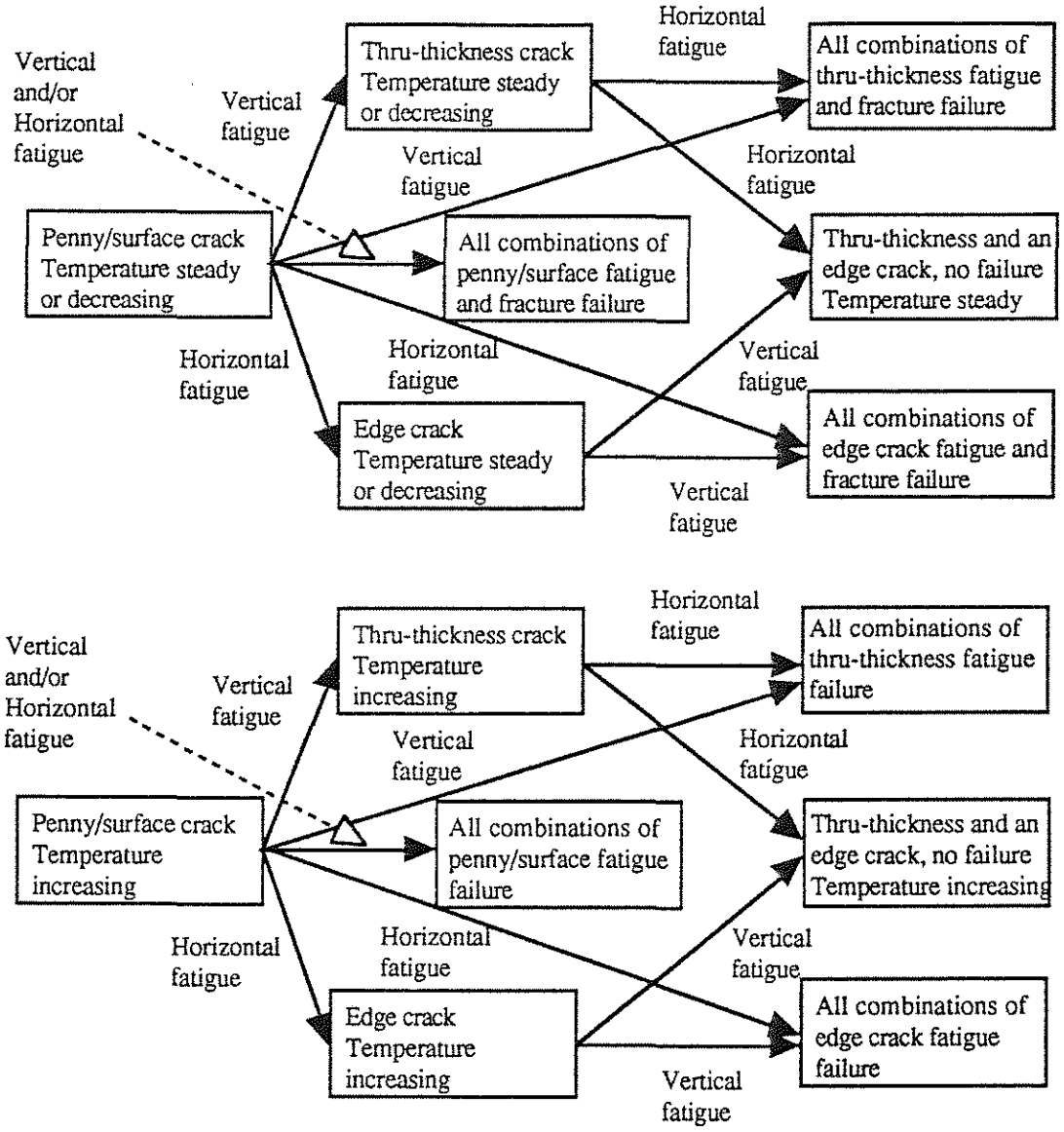

Fig. 4-22 Interpreted Envisionment of QPE for Fatigue and Fracture Model

Overall, the envisionment produced by QPE and the history generation produced by QSIM are similar. Both systems predict possible real behaviors for an

actual physical situation. Erroneous behaviors are caused by spurious introduction of landmarks in QSIM and by the qualitative state cycles in temporally generic envisionments in QPE. These problems did not arise in simulating the fatigue and fracture domain.

The simulation of the simple model of fatigue and fracture predicts all of the possible behaviors. QPE simulated all of the possible situations including the case in which the crack is both an edge crack and a thru-thickness crack but has not failed (Fig. 4-23). Simulation from the initial conditions of (Fig. 3-12) in QSIM results in sixteen distinct behaviors. However, if the only parameter that is specified in the initial conditions is that there is vertical crack growth, then QSIM predicts one hundred twenty-six behaviors. There are eighteen possible behaviors if fatigue is not specified in the initial conditions and there are seven combinations of initial states for these eighteen behaviors. The combination of parameter situations becomes very large.

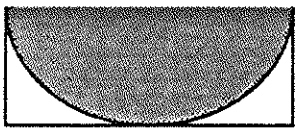

Fig. 4-23 Edge and Thru-thickness crack

The envisionment produced by QPE is also large and requires some interpretations. S-classes organize the behavior minimally in the fatigue and fracture domain (See envisionment in Appendix B). The different derivative values of environments are divided into different S-classes. The effects of temperature on K_c cause many similar environments to be classified into different S-classes.

Environments (or situations) 177, 256, 257, 258, and 175 are all penny/surface cracks that are under the influences of horizontal fatigue, vertical fatigue, and temperature effects. However, these environments are organized into different S-classes because of the relationship between the derivative of K_c and the landmark zero.

The large number of behaviors in QSIM and the number of S-classes in QPE require careful examination. One advantage of the interpretation of results in QPE is the causality expressed by the active processes. The processes that are active in a state and not active in the successor state are viewed as the *cause* of the change. For instances, a state in QPE is described by the horizontal fatigue process being active, the vertical fatigue process being active, and the penny/surface crack view being active. This state transitions to a successor state which is described by the horizontal fatigue being active and the thru-thickness view being active. Since the successor state is not described by the vertical fatigue process being active, the vertical fatigue process *caused* the crack to become a thru-thickness crack. In QSIM, the results of simulation do not explicitly show causality because the results show a set of parameters changing in various directions that may or may not be the cause of change.

Chapter 5

Conclusions

5.1 Comparison of Modeling Techniques

Modeling is the most important step in qualitative reasoning. Since qualitative models are rarely direct translations of mathematical equations, the qualitative modeling system should provide for a modular and intuitive approach to modeling. The division of the parts of a model provides an intuitive approach to incremental model building through reuse of already designed modular components. This division also ensures that models are robust and not designed solely for a particular situation.

Modeling in QSIM is through a qualitative differential equation which includes all objects, relationships between parameters, landmarks, and transitions to other models. Although the basic modeling process in QSIM is not conducive towards a modular approach, there is an evolution of research based around QSIM towards expanding the system for easier modeling [Crawford, Farquhar, Kuipers 1990], [Franke & Dvorak 1989], [Kuipers & Berleant 1988]. The other benefits of modeling in QSIM are useful qualitative extensions such as the use of time-scale abstraction and non-analytical functions. All of these modeling techniques are constrained by the advantages and disadvantages of the simulation process in QSIM.

QPE is a qualitative reasoning technique that addresses many of the important issues of qualitative modeling. The modeling system is divided into views, processes, and entities. Modeling systems through processes causing change and through views as important limit points of a behavior is useful in building intuitive common sense

models. Also, QPE is a useful method in organizing the "pre-analysis" part of problem solving. The process-centered theory enables the selection of the appropriate model for objects (depending on the situation) to occur during simulation. For instance, metal structures during fatigue and fracture analysis are modeled in different ways depending on their region of operation. The initial success of modeling in QPE to automate the selection of models and to organize diverse knowledge into reuseable libraries is encouraging.

5.2 Comparison of Simulation Techniques

QSIM is small, simple, and fast [Forbus 1990]. QSIM has the advantages of dynamic introduction of landmarks and actual history generation. Landmarks are useful in comparing qualitative values generated during simulation. However, the exponential introduction of landmarks is a serious problem in simulation of useful models.

The simulation process in QSIM is most useful in exploring specific regions of qualitative behaviors. It becomes difficult to interpret the results of simulation of hundreds of actual behaviors. The generation of specific results are useful in situations that have limited branches but the combinatorics of behaviors for under-specified initial conditions or for situations that have an under-developed qualitative structure is unwieldy.

The envisionment results of QPE are useful in creating a finite representation for possibly infinite behaviors. The envisionment "collapses" behaviors together by only specifying the possible transitions between qualitative states and not the actual individual behaviors. However, these results are also difficult to interpret in complex

models because S-classes do not group qualitative states effectively. The results of QPE also elucidate causality in a model through its representation of processes. A problem with QPE is that a created envisionment must be total which can become intractable for complex models. Also, the results of QPE are general and the generation of exact behaviors from envisionments can include erroneous behaviors.

5.3 Comparison for fatigue and fracture

5.3.1 Current models

The simple prototypes developed in QPE and QSIM illustrate the basic advantages and disadvantages of the use of qualitative reasoning in the domain of fatigue and fracture in metals. The QSIM model (Fig. 3-13) effectively represents the basic fatigue cycle due to horizontal and vertical crack growth. Crack growth is caused by some cyclic loading that is not explicitly represented in the model. The modeling of two dimensional crack growth requires three models (in QSIM). Although using three models to represent two dimensional crack growth is not a major issue, the number of models needed for more complex models (e.g. three dimensional crack growth) grows exponentially.

The QPE model (Fig. 3-16) also includes the effects of ambient temperature on the fracture toughness (K_c) of the material. Additional parameters are easily modeled through the use of processes and views. The ability to turn "on or off" parameters is accomplished through pattern matching of parameters in the scenario model with the "individuals" section of views and processes. This feature can be used to represent the different processes in various types of metals and is useful in testing correctness of

parameter modeling.

The effects of temperature in the fatigue and fracture models greatly adds to the complexity of the results. Manual interpretation of the results is a time-consuming process. The number of qualitative states for the QPE fatigue and fracture model is between forty and fifty (probably analogous to sixty or seventy behaviors in QSIM). Since the fatigue and fracture models are relatively simple, more complex models can result in hundreds of behaviors. The interpretation of the results of qualitative reasoning are a major concern in the continued evaluation of QPE and QSIM in an engineering tool.

5.3.2 Future work

The first step in continued use and evaluation of qualitative reasoning is a better understanding of the results of qualitative reasoning. The interpretation of the results must be concise and clear. Modeling and simulation is an iterative process. Proper interpretation of the results leads to better developed models which utilize the full potential of qualitative reasoning. One potential solution is the development of techniques that can "learn" and generalize the results of qualitative reasoning. Another possible solution in understanding the behaviors predicted through qualitative reasoning is the use of probability in behavior prediction.

The full solution is ultimately a better understanding of qualitative models. First, the origin of qualitative models is somewhat unclear. The abstraction of quantitative models into qualitative models is frustrating and clearly not the best technique. Preciseness is lost in the translation to qualitative models and lost again in the more general results of qualitative reasoning.

The qualitative models are created mainly from known equations in fatigue and fracture analysis. To best use QPE's modeling capabilities, a different kind of understanding is needed. Although there are many functions involving derivative values in fatigue and fracture analysis, it is unlikely that all of these functions need to be represented by processes. Discontinuous behavior that arise from processes and views also needs to be explored further in QPE. Qualitative models are most useful in modeling systems in which the combinations of parameters and equations makes quantitative analysis unwieldy.

There are many techniques that can be used to improve qualitative modeling. One technique that could very useful is the ability to "turn on or off" certain aspects of a model. This technique can be used to switch between different granularities of modeling in QPE. Granularity of modeling can not be dynamically selected in QSIM. Although QPE has many advantages over QSIM in modeling, the techniques of time-scale abstraction and non-analytic functions are useful qualitative extensions. For instance, time scale abstraction might be used to differentiate between the fatigue and the fracture process. The non-analytic function can be used to model the relationship between temperature and the fracture toughness of the material.

Further extensions to the fatigue and fracture prototype should include additional exploration of the effects of loading. A random loading of the structure might be interesting to explore. The use of dynamic creation of landmarks in QSIM to determine cyclic behavior could be useful in differentiating between random-stress loading and constant amplitude loading if the applied stress on the structure is explicitly represented. Also, the effects of temperature on K_c is a short-cut representation. The use of an intermediate variable so that temperature can directly affect K is a better

representation because additional parameters may affect the relationship between K and temperature.

The methods of qualitative reasoning alone are frequently too general for effective use. The use of QPE or QSIM in a successful engineering tool requires integration of other methods. Integration of quantitative knowledge can greatly reduce ambiguity of qualitative models. The natural flow of control is to use qualitative reasoning to "guide" quantitative techniques. However, the relationship between qualitative models and quantitative models is not well understood. This relationship can be explored by beginning with the quantitative models and then building qualitative models that represent important information which can select between different sets of quantitative models. Although flow of control should go from a higher level of abstraction to less abstraction, a bottom-up approach to design of an engineering reasoning system is warranted.

Qualitative reasoning is useful in solving many types of common sense problems that are not easily solved through other methods. The choice of using QPE or QSIM as part of an engineering tool depends on the level of problem to be solved. If qualitative reasoning is to be used in a specific problem, then QSIM is a quick and efficient tool. However, if the overall goals of the tool are a large integration of diverse knowledge bases and the use of qualitative reasoning to "guide" quantitative reasoning, then QPE is a better tool. The best use of QPE is as a high level tool which avoids the issue of behavior prediction being tractable.

5.4 Summary

This project involved porting the two implementations to another platform and

the discovery and correction of minor errors in the implementations. Simple prototypes in the domain of fatigue and fracture of metals were developed to illustrate the advantages and disadvantages in a comparison of QPE and QSIM. This comparison helps summarize the current level of qualitative reasoning and will encourage new research into qualitative reasoning.

Bibliography

Bannantine, Comer, Handrock 1990

Bannantine, J., Comer, J., Handrock, J. Fundamentals of Metal Fatigue Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1990.

Barsom & Rolfe 1987

Barsom, John M., and Stanley T. Rolfe, *Fracture & Fatigue Control in Structures*, Second Edition, Prentice-Hall, Englewood Cliffs, NJ, 1987.

Collins & Forbus 1987

Collins, John W., Forbus, Kenneth D. "Reasoning About Fluids Via Molecular Collections" *Proceedings* AAAI-87 *Sixth National Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, San Mateo, CA, 1988.

Crawford, Farquhar, Kuipers 1990

Crawford,J., Farquhar, A., Kuipers, B. "QPC: A Compiler from Physical Models into Qualitative Differential Equations", *Proceedings* AAAI-90 *Eight National Conference on Artificial Intelligence*,The MIT Press, Cambridge, MA, 1990.

Davis 1987

Davis, E., "Order of Magnitude Reasoning in Qualitative Differential Equations, Technical Report TR-312, Computer Science Department, New York University, New York. 1987

D'Ambrosio 1989a

D' Ambrosio, Bruce, *Qualitative Process Theory Using Linguistic Variables*, Springer-Verlag, New York, 1989.

D'Ambrosio 1989b

D'Ambrosio, Bruce "Extending the Mathematics in Qualitative Process Theory", Widman, Lawrence E., et al., editors, *Artificial Intelligence, Simulation, and Modeling*, John Wiley & Sons, Inc., 1989.

de Kleer 1977

de Kleer, Johan, "Multiple Representations of Knowledge in a Mechanics Problem Solver", Proceedings IJCAI-77, Cambridge, MA, 1977

de Kleer 1986

de Kleer, Johan, "An Assumption-based TMS", Artificial Intelligence, Volume 28, 1986.

de Kleer & Brown 1985

de Kleer, Johan, Brown, John S. "A Qualitative Physics Based On Confluences", Weld, D. & de Kleer, J., editors, *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann 1990.

Falkenhainer & Forbus 1990

Falkenhainer, B., Forbus, K. "Setting up Large-Scale Qualitative Models", Weld, D. & de Kleer, J., editors, *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann 1990.

Forbus 1984

Forbus, K. *Qualitative Process Theory*. Technical Report MIT-AI-TR-789, Massachusetts Institute of Technology, 1984.

Forbus 1988

Forbus, Kenneth D., "Intelligent Computer-Aided Engineering", AI Magazine, Vol.9 No.3, American Association for Artificial Intelligence, Fall, 1988.

Forbus 1990a

Forbus, K. "The Qualitative Process Engine", Weld, D. & de Kleer, J., editors, *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann 1990.

Forbus 1990b

Forbus, K. "Qualitative Physics: Past, Present, and Future", Weld, D. & de Kleer, J., editors, *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann 1990.

Forbus 1990c

Forbus, K. "Interpreting Observations of Physical Systems", Weld, D. & de Kleer, J., editors, *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann 1990.

Forbus & Falkenhainer 1990

Forbus, K., Falkenhainer, B. "Self-Explanatory Simulations: An integration of qualitative and quantitative knowledge", *Proceedings* AAAI-90 *Eight National Conference on Artificial Intelligence*,The MIT Press, Cambridge, MA, 1990.

Forbus & Genter 1986

Forbus, K.D., and Gentner, D., "Causal Reasoning about Quantities", *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA, 1986.

Franke & Dvorak 1990

Franke, David, Dvorak, Dan, "CC:Component Connection Models for Qualitative Simulation - A User's Guide" Artificial Intelligence Laboratory, The University of Texas at Austin, Technical Report A190-126, 1990.

Hayes 1979

Hayes, Patrick J., "Naive Physics I: Ontology For Liquids", Weld, D. & de Kleer, J., editors, *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann 1990.

Hayes 1985

Hayes, Patrick J., "The Second Naive Physics Manifesto", Weld, D. & de Kleer, J., editors, *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann 1990.

Kuipers 1985

Kuipers, B., "Commonsense Reasoning About Causality: Deriving Behavior from Structure", Bobrow, Daniel G., editor, *Qualitative Reasoning About Physical Systems*, The MIT Press, Cambridge, MA, 1985.

Kuipers 1986

Kuipers, B., "Qualitative Simulation", *Artificial Intelligence*, Vol. 29, No.2, March 1986.

Kuipers 1987

Kuipers, B., "Abstraction by Time-Scale in Qualitative Simulation", Weld, D. & de Kleer, J., editors, *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann 1990.

Kuipers 1989

Kuipers, B., "Qualitative reasoning:modeling and simulation with incomplete knowledge", *Automatica*, Vol. 25, 1989.

Kuipers & Berleant 1988

Kuipers, B. Berleant, D. "Using Incomplete Quantitative Knowledge in Qualitative Reasoning", *Proceedings* AAAI-88 *Seventh National Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, San Mateo, CA, 1988.

Kuipers & Chiu 1990

Kuipers, B., Chiu, C. "Taming Intractable Branching in Qualitative Simulation", Weld, D. & de Kleer, J., editors, *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann, 1990.

QPE manual 1990

Forbus, K., "The QPE Manual", University of Illinois Department of Computer Science Technical Report, in preparation.

QSIM Maintainers Guide 1990

Throop, David., Artificial Intelligence Laboratory, The University of Texas at Austin, 1990.

QSIM POS: The PostSCripting Facility 1990

Throop, David., Artificial Intelligence Laboratory, The University of Texas at Austin, 1990.

QSIM User's Manual 1990

Farquhar, Adam, Kuipers, Benjamin, Artificial Intelligence Laboratory, The University of Texas at Austin, 1990

Roddis 1988

Roddis, W. M. Kim, *Heuristic, Qualitative, and Quantitative Reasoning About Steel Bridge Fatigue and Fracture*, Doctoral Thesis, Civil Engineering Department, Massachusetts Institute of Technology, Cambridge, MA, September, 1988.

Rothenberg 1989

Rothenberg, Jeff, "Nature of Modeling", Widman, Lawrence E., et al., editors, *Artificial Intelligence, Simulation, and Modeling*, John Wiley & Sons, Inc., 1989

Weld 1988

Weld, D., "Comparative Analysis", Artificial Intelligence, Vol. 36, No.3, October 1988.

Appendix A: QPE Envisionment of U-tube

```
;U-tube
18:26:4  8-9-1990
QPE (version 2.6 Beta) Envisionment
  ATMoSphere version 3.2 Beta, ATMS version 21
      Macintosh Allegro Common Lisp, 1.3.2
     unspecified
```

```
   Envisoner results,

12 situations, in 6 classes
1 Domain files:
   HD:QPE files:DOMAINS:nst.lisp
1 Example files:
   HD:QPE files:EXAMPLES:ex1.lisp
No user augments declared.
```

```
QPE MODES
Careful mode OFF
Zero Ds values printed.
Display situations by differences
Preconditions assumed.
Add assumptions for resolving pairs of direct influences
Discontinuous direct influences allowed.
Derivative Augments: ON
Simultaneous changes to and from =: ON.
Equality Change Law, Case 2: ON
Unstable transitions not allowed.
Asymptotic approach states pruned.
```

```
LIMIT ANALYSIS STATISTICS
12 situations, 4 potential transitions (Max = 2, Min = 0, Ave =.33)
Before Continuity: 4 potential transitions (Max = 2, Min = 2, Ave =2.0)
Before ECL: 2 potential transitions (Max = 1, Min = 1, Ave =1.0)
Final result: 2 transitions (Max = 1, Min = 1, Ave =1.0)
```

```
ADB STATISTICS
# database items: 955
# database classes: 84
# rules run: 540
# C-rules run: 0
# TMS nodes: 1049
# TMS classes: 461
# Nogoods: 90
# assumptions: 18
# environments: 235
```

Justifications: 1773
Distribution of Nogoods:

Length	# nogoods
2	19
3	32
4	9
5	12

72 nogoods in all

---- View and Process Instances ----
View instances:
VI0 = CONTAINED-STUFF(C-S(WATER,LIQUID,F))
VI1 = CONTAINED-STUFF(C-S(WATER,LIQUID,G))
Process instances:
PI0 = LIQUID-FLOW(WATER,F,G,P1)
PI1 = LIQUID-FLOW(WATER,G,F,P1)

---- Union of Quantity Space information ----
From Quantity Conditions (3):
A[PRESSURE(C-S(WATER,LIQUID,F))] A[PRESSURE(C-S(WATER,LIQUID,G))]
A[AMOUNT-OF-IN(WATER,LIQUID,G)] ZERO
A[AMOUNT-OF-IN(WATER,LIQUID,F)] ZERO
From Influence Resolution (2):
A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE)]
ZERO
A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE)]
ZERO
From Limit Analysis (0):
Derived (15):
D[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE)]
ZERO
D[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE)]
ZERO
D[AMOUNT-OF-IN(WATER,LIQUID,F)] ZERO
D[AMOUNT-OF-IN(WATER,LIQUID,G)] ZERO
A[FLOW-RATE(PI1)] A[PRESSURE(C-S(WATER,LIQUID,G))]
D[FLOW-RATE(PI1)] D[PRESSURE(C-S(WATER,LIQUID,G))]
A[FLOW-RATE(PI0)] A[PRESSURE(C-S(WATER,LIQUID,F))]
D[PRESSURE(C-S(WATER,LIQUID,F))] D[PRESSURE(C-S(WATER,LIQUID,G))]
D[FLOW-RATE(PI0)] D[PRESSURE(C-S(WATER,LIQUID,F))]
A[AMOUNT-OF(C-S(WATER,LIQUID,F))]
A[AMOUNT-OF(C-S(WATER,LIQUID,G))]
A[LEVEL(C-S(WATER,LIQUID,F))] A[LEVEL(C-S(WATER,LIQUID,G))]
A[AMOUNT-OF(C-S(WATER,LIQUID,G))] ZERO
A[BOTTOM-HEIGHT(G)] A[LEVEL(C-S(WATER,LIQUID,G))]
A[AMOUNT-OF(C-S(WATER,LIQUID,F))] ZERO

A[BOTTOM-HEIGHT(F)] A[LEVEL(C-S(WATER,LIQUID,F))]
Total = 20.

--- Summary of Limit Hypotheses ---

LH0:[>=]A[PRESSURE(C-S(WATER,LIQUID,F))]
<A[PRESSURE(C-S(WATER,LIQUID,G))]-->=
LH1:[<=]A[PRESSURE(C-S(WATER,LIQUID,F))]
>A[PRESSURE(C-S(WATER,LIQUID,G))]-->=
LH2:[<=]A[AMOUNT-OF-IN(WATER,LIQUID,G)]
>ZERO-->=
LH3:[<=]A[AMOUNT-OF-IN(WATER,LIQUID,F)]
>ZERO-->=

----- STATE TRANSITION TABLE ------

2 states out of 12 have candidate transitions.

ENV-228 : {LH1:ENV-227}
ENV-226 : {LH0:ENV-227}

```
Sclass S0, 2 situations:
  Status = R-COMPLETE, Duration = INTERVAL
  IS:{QPE}
      No active individual views.
      No active processes.

     -*- Ds Values -*-
  Ds[AMOUNT-OF-IN(WATER,LIQUID,F)]=0
  Ds[AMOUNT-OF-IN(WATER,LIQUID,G)]=0
  Ds[BOTTOM-HEIGHT(F)]=0
  Ds[BOTTOM-HEIGHT(G)]=0
  Ds[MAX-HEIGHT(P1)]=0
  Ds[TBOIL(WATER,F)]=0
  Ds[TBOIL(WATER,G)]=0
  Ds[TOP-HEIGHT(F)]=0
  Ds[TOP-HEIGHT(G)]=0

     -*- Environments -*-
In common:
 Env ENV-237:
  A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE)]??ZERO
  A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE)]??ZERO
  A[AMOUNT-OF-IN(WATER,LIQUID,F)]=ZERO
  A[AMOUNT-OF-IN(WATER,LIQUID,G)]=ZERO
  A[PRESSURE(C-S(WATER,LIQUID,F))]??A[PRESSURE(C-S(WATER,LIQUID,G))]
  ENFORCE(QUANTITY-EXISTENCE)
Differences: (2)
 ENV-7 + ENV-237 = ENV-223
 Env ENV-7:
  ~ALIGNED(P1)
 ENV-6 + ENV-237 = ENV-221
 Env ENV-6:
  ALIGNED(P1)

Sclass S1, 2 situations:
  Status = R-COMPLETE, Duration = INTERVAL
  IS:{QPE,C-S(WATER,LIQUID,F)}
   VS: VI0: CONTAINED-STUFF(C-S(WATER,LIQUID,F))
      No active processes.

     -*- Ds Values -*-
  Ds[AMOUNT-OF(C-S(WATER,LIQUID,F))]=0
  Ds[AMOUNT-OF-IN(WATER,LIQUID,F)]=0
  Ds[AMOUNT-OF-IN(WATER,LIQUID,G)]=0
  Ds[BOTTOM-HEIGHT(F)]=0
  Ds[BOTTOM-HEIGHT(G)]=0
  Ds[HEAT(C-S(WATER,LIQUID,F))]=0
```

```
  Ds[LEVEL(C-S(WATER,LIQUID,F))]=0
  Ds[MAX-HEIGHT(P1)]=0
  Ds[PRESSURE(C-S(WATER,LIQUID,F))]=0
  Ds[TBOIL(WATER,F)]=0
  Ds[TBOIL(WATER,G)]=0
  Ds[TEMPERATURE(C-S(WATER,LIQUID,F))]=0
  Ds[TOP-HEIGHT(F)]=0
  Ds[TOP-HEIGHT(G)]=0
  Ds[VOLUME(C-S(WATER,LIQUID,F))]=0

     -*- Environments -*-
In common:
 Env ENV-238:
  A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE)]??ZERO
  A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE)]??ZERO
  A[AMOUNT-OF-IN(WATER,LIQUID,F)]>ZERO
  A[AMOUNT-OF-IN(WATER,LIQUID,G)]=ZERO
  A[PRESSURE(C-S(WATER,LIQUID,F))]??A[PRESSURE(C-S(WATER,LIQUID,G))]
  ENFORCE(QUANTITY-EXISTENCE)
Differences: (2)
 ENV-7 + ENV-238 = ENV-224
 Env ENV-7:
  ~ALIGNED(P1)
 ENV-6 + ENV-238 = ENV-222
 Env ENV-6:
  ALIGNED(P1)

Sclass S2, 2 situations:
   Status = R-COMPLETE, Duration = INTERVAL
   IS:{QPE,C-S(WATER,LIQUID,G)}
    VS: VI1: CONTAINED-STUFF(C-S(WATER,LIQUID,G))
       No active processes.

     -*- Ds Values -*-
  Ds[AMOUNT-OF(C-S(WATER,LIQUID,G))]=0
  Ds[AMOUNT-OF-IN(WATER,LIQUID,F)]=0
  Ds[AMOUNT-OF-IN(WATER,LIQUID,G)]=0
  Ds[BOTTOM-HEIGHT(F)]=0
  Ds[BOTTOM-HEIGHT(G)]=0
  Ds[HEAT(C-S(WATER,LIQUID,G))]=0
  Ds[LEVEL(C-S(WATER,LIQUID,G))]=0
  Ds[MAX-HEIGHT(P1)]=0
  Ds[PRESSURE(C-S(WATER,LIQUID,G))]=0
  Ds[TBOIL(WATER,F)]=0
  Ds[TBOIL(WATER,G)]=0
  Ds[TEMPERATURE(C-S(WATER,LIQUID,G))]=0
  Ds[TOP-HEIGHT(F)]=0
  Ds[TOP-HEIGHT(G)]=0
```

```
  Ds[VOLUME(C-S(WATER,LIQUID,G))]=0

     -*- Environments -*-
In common:
 Env ENV-239:
  A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE)]??ZERO
  A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE)]??ZERO
  A[AMOUNT-OF-IN(WATER,LIQUID,F)]=ZERO
  A[AMOUNT-OF-IN(WATER,LIQUID,G)]>ZERO
  A[PRESSURE(C-S(WATER,LIQUID,F))]??A[PRESSURE(C-S(WATER,LIQUID,G))]
  ENFORCE(QUANTITY-EXISTENCE)
Differences: (2)
 ENV-7 + ENV-239 = ENV-229
 Env ENV-7:
  ~ALIGNED(P1)
 ENV-6 + ENV-239 = ENV-225
 Env ENV-6:
  ALIGNED(P1)

Sclass S3, 1 situations:
   Status = R-COMPLETE, Duration = INTERVAL
   IS:[QPE,C-S(WATER,LIQUID,F),C-S(WATER,LIQUID,G)]
    VS: VI0: CONTAINED-STUFF(C-S(WATER,LIQUID,F))
       VI1: CONTAINED-STUFF(C-S(WATER,LIQUID,G))
    PS: PI1: LIQUID-FLOW(WATER,G,F,P1)

     -*- Ds Values -*-
   Ds[AMOUNT-OF(C-S(WATER,LIQUID,F))]=1
   Ds[AMOUNT-OF(C-S(WATER,LIQUID,G))]=-1
   Ds[AMOUNT-OF-IN(WATER,LIQUID,F)]=1
   Ds[AMOUNT-OF-IN(WATER,LIQUID,G)]=-1
   Ds[BOTTOM-HEIGHT(F)]=0
   Ds[BOTTOM-HEIGHT(G)]=0
   Ds[FLOW-RATE(PI1)]=-1
   Ds[HEAT(C-S(WATER,LIQUID,F))]=0
   Ds[HEAT(C-S(WATER,LIQUID,G))]=0
   Ds[LEVEL(C-S(WATER,LIQUID,F))]=1
   Ds[LEVEL(C-S(WATER,LIQUID,G))]=-1
   Ds[MAX-HEIGHT(P1)]=0
   Ds[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE)]=-1
   Ds[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE)]=1
   Ds[PRESSURE(C-S(WATER,LIQUID,F))]=1
   Ds[PRESSURE(C-S(WATER,LIQUID,G))]=-1
   Ds[TBOIL(WATER,F)]=0
   Ds[TBOIL(WATER,G)]=0
   Ds[TEMPERATURE(C-S(WATER,LIQUID,F))]=0
   Ds[TEMPERATURE(C-S(WATER,LIQUID,G))]=0
   Ds[TOP-HEIGHT(F)]=0
```

Ds[TOP-HEIGHT(G)]=0
Ds[VOLUME(C-S(WATER,LIQUID,F))]=0
Ds[VOLUME(C-S(WATER,LIQUID,G))]=0

-*- Environments -*-

Env ENV-226:
A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE)]>ZERO
A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE)]<ZERO
A[AMOUNT-OF-IN(WATER,LIQUID,F)]>ZERO
A[AMOUNT-OF-IN(WATER,LIQUID,G)]>ZERO
A[PRESSURE(C-S(WATER,LIQUID,F))]<A[PRESSURE(C-S(WATER,LIQUID,G))]
ALIGNED(P1)
ENFORCE(QUANTITY-EXISTENCE)

Sclass S4, 4 situations:
Status = R-COMPLETE, Duration = INTERVAL
IS:{QPE,C-S(WATER,LIQUID,F),C-S(WATER,LIQUID,G)}
VS: VI0: CONTAINED-STUFF(C-S(WATER,LIQUID,F))
VI1: CONTAINED-STUFF(C-S(WATER,LIQUID,G))
No active processes.

-*- Ds Values -*-
Ds[AMOUNT-OF(C-S(WATER,LIQUID,F))]=0
Ds[AMOUNT-OF(C-S(WATER,LIQUID,G))]=0
Ds[AMOUNT-OF-IN(WATER,LIQUID,F)]=0
Ds[AMOUNT-OF-IN(WATER,LIQUID,G)]=0
Ds[BOTTOM-HEIGHT(F)]=0
Ds[BOTTOM-HEIGHT(G)]=0
Ds[HEAT(C-S(WATER,LIQUID,F))]=0
Ds[HEAT(C-S(WATER,LIQUID,G))]=0
Ds[LEVEL(C-S(WATER,LIQUID,F))]=0
Ds[LEVEL(C-S(WATER,LIQUID,G))]=0
Ds[MAX-HEIGHT(P1)]=0
Ds[PRESSURE(C-S(WATER,LIQUID,F))]=0
Ds[PRESSURE(C-S(WATER,LIQUID,G))]=0
Ds[TBOIL(WATER,F)]=0
Ds[TBOIL(WATER,G)]=0
Ds[TEMPERATURE(C-S(WATER,LIQUID,F))]=0
Ds[TEMPERATURE(C-S(WATER,LIQUID,G))]=0
Ds[TOP-HEIGHT(F)]=0
Ds[TOP-HEIGHT(G)]=0
Ds[VOLUME(C-S(WATER,LIQUID,F))]=0
Ds[VOLUME(C-S(WATER,LIQUID,G))]=0

-*- Environments -*-
In common:
Env ENV-240:

```
 A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE)]??ZERO
 A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE)]??ZERO
 A[AMOUNT-OF-IN(WATER,LIQUID,F)]>ZERO
 A[AMOUNT-OF-IN(WATER,LIQUID,G)]>ZERO
 ENFORCE(QUANTITY-EXISTENCE)
Differences: (4)
 ENV-90 + ENV-240 = ENV-232
 Env ENV-90:
  A[PRESSURE(C-S(WATER,LIQUID,F))]>A[PRESSURE(C-S(WATER,LIQUID,G))]
  ~ALIGNED(P1)
 ENV-88 + ENV-240 = ENV-231
 Env ENV-88:
  A[PRESSURE(C-S(WATER,LIQUID,F))]=A[PRESSURE(C-S(WATER,LIQUID,G))]
  ~ALIGNED(P1)
 ENV-17 + ENV-240 = ENV-230
 Env ENV-17:
  A[PRESSURE(C-S(WATER,LIQUID,F))]<A[PRESSURE(C-S(WATER,LIQUID,G))]
  ~ALIGNED(P1)
 ENV-241 + ENV-240 = ENV-227
 Env ENV-241:
  A[PRESSURE(C-S(WATER,LIQUID,F))]=A[PRESSURE(C-S(WATER,LIQUID,G))]
  ALIGNED(P1)

Sclass S5, 1 situations:
   Status = R-COMPLETE, Duration = INTERVAL
   IS:{QPE,C-S(WATER,LIQUID,F),C-S(WATER,LIQUID,G)}
    VS: VI0: CONTAINED-STUFF(C-S(WATER,LIQUID,F))
        VI1: CONTAINED-STUFF(C-S(WATER,LIQUID,G))
    PS: PI0: LIQUID-FLOW(WATER,F,G,P1)

       -*- Ds Values -*-
   Ds[AMOUNT-OF(C-S(WATER,LIQUID,F))]=-1
   Ds[AMOUNT-OF(C-S(WATER,LIQUID,G))]=1
   Ds[AMOUNT-OF-IN(WATER,LIQUID,F)]=-1
   Ds[AMOUNT-OF-IN(WATER,LIQUID,G)]=1
   Ds[BOTTOM-HEIGHT(F)]=0
   Ds[BOTTOM-HEIGHT(G)]=0
   Ds[FLOW-RATE(PI0)]=-1
   Ds[HEAT(C-S(WATER,LIQUID,F))]=0
   Ds[HEAT(C-S(WATER,LIQUID,G))]=0
   Ds[LEVEL(C-S(WATER,LIQUID,F))]=-1
   Ds[LEVEL(C-S(WATER,LIQUID,G))]=1
   Ds[MAX-HEIGHT(P1)]=0
   Ds[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE)]=1
   Ds[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE)]=-1
   Ds[PRESSURE(C-S(WATER,LIQUID,F))]=-1
   Ds[PRESSURE(C-S(WATER,LIQUID,G))]=1
   Ds[TBOIL(WATER,F)]=0
```

Ds[TBOIL(WATER,G)]=0
Ds[TEMPERATURE(C-S(WATER,LIQUID,F))]=0
Ds[TEMPERATURE(C-S(WATER,LIQUID,G))]=0
Ds[TOP-HEIGHT(F)]=0
Ds[TOP-HEIGHT(G)]=0
Ds[VOLUME(C-S(WATER,LIQUID,F))]=0
Ds[VOLUME(C-S(WATER,LIQUID,G))]=0

-*- Environments -*-

Env ENV-228:
A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE)]<ZERO
A[NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE)]>ZERO
A[AMOUNT-OF-IN(WATER,LIQUID,F)]>ZERO
A[AMOUNT-OF-IN(WATER,LIQUID,G)]>ZERO
A[PRESSURE(C-S(WATER,LIQUID,F))]>A[PRESSURE(C-S(WATER,LIQUID,G))]
ALIGNED(P1)
ENFORCE(QUANTITY-EXISTENCE)

There are 4 limit hypotheses, 4 singletons and 0 conjunctive.

LH0:[>=]A[PRESSURE(C-S(WATER,LIQUID,F))]
<A[PRESSURE(C-S(WATER,LIQUID,G))]-->=
Start: (ENV-193)
End: (ENV-68)
Env ENV-193:
A[AMOUNT-OF-IN(WATER,LIQUID,F)]>ZERO
A[AMOUNT-OF-IN(WATER,LIQUID,G)]>ZERO
A[PRESSURE(C-S(WATER,LIQUID,F))]<A[PRESSURE(C-S(WATER,LIQUID,G))]
ALIGNED(P1)
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-68:
A[AMOUNT-OF-IN(WATER,LIQUID,F)]>ZERO
A[AMOUNT-OF-IN(WATER,LIQUID,G)]>ZERO
A[PRESSURE(C-S(WATER,LIQUID,F))]=A[PRESSURE(C-S(WATER,LIQUID,G))]
ENFORCE(QUANTITY-EXISTENCE)

LH1:[<=]A[PRESSURE(C-S(WATER,LIQUID,F))]
>A[PRESSURE(C-S(WATER,LIQUID,G))]-->=
Start: (ENV-194)
End: (ENV-68)
Env ENV-194:
A[AMOUNT-OF-IN(WATER,LIQUID,F)]>ZERO
A[AMOUNT-OF-IN(WATER,LIQUID,G)]>ZERO
A[PRESSURE(C-S(WATER,LIQUID,F))]>A[PRESSURE(C-S(WATER,LIQUID,G))]
ALIGNED(P1)
ENFORCE(QUANTITY-EXISTENCE)

```
 Env ENV-68:
  A[AMOUNT-OF-IN(WATER,LIQUID,F)]>ZERO
  A[AMOUNT-OF-IN(WATER,LIQUID,G)]>ZERO
  A[PRESSURE(C-S(WATER,LIQUID,F))]=A[PRESSURE(C-S(WATER,LIQUID,G))]
  ENFORCE(QUANTITY-EXISTENCE)

LH2:[<=]A[AMOUNT-OF-IN(WATER,LIQUID,G)]
        >ZERO-->=
   Start: (ENV-35)
     End: (ENV-23)
 Env ENV-35:
  A[AMOUNT-OF-IN(WATER,LIQUID,G)]>ZERO
  A[PRESSURE(C-S(WATER,LIQUID,F))]<A[PRESSURE(C-S(WATER,LIQUID,G))]
  ALIGNED(P1)
  ENFORCE(QUANTITY-EXISTENCE)
 Env ENV-23:
  A[AMOUNT-OF-IN(WATER,LIQUID,G)]=ZERO
  ENFORCE(QUANTITY-EXISTENCE)

LH3:[<=]A[AMOUNT-OF-IN(WATER,LIQUID,F)]
        >ZERO-->=
   Start: (ENV-64)
     End: (ENV-39)
 Env ENV-64:
  A[AMOUNT-OF-IN(WATER,LIQUID,F)]>ZERO
  A[PRESSURE(C-S(WATER,LIQUID,F))]>A[PRESSURE(C-S(WATER,LIQUID,G))]
  ALIGNED(P1)
  ENFORCE(QUANTITY-EXISTENCE)
 Env ENV-39:
  A[AMOUNT-OF-IN(WATER,LIQUID,F)]=ZERO
  ENFORCE(QUANTITY-EXISTENCE)
```

Table of possible direct influencers.

```
 AMOUNT-OF-IN(WATER,LIQUID,G) :
    ACTIVE(PI1) contributes:
    I-(AMOUNT-OF-IN(WATER,LIQUID,G),A[FLOW-RATE(PI1)])
    ACTIVE(PI0) contributes:
    I+(AMOUNT-OF-IN(WATER,LIQUID,G),A[FLOW-RATE(PI0)])

 AMOUNT-OF-IN(WATER,LIQUID,F) :
    ACTIVE(PI1) contributes:
    I+(AMOUNT-OF-IN(WATER,LIQUID,F),A[FLOW-RATE(PI1)])
    ACTIVE(PI0) contributes:
    I-(AMOUNT-OF-IN(WATER,LIQUID,F),A[FLOW-RATE(PI0)])
```

Table of possible indirect influencers.

NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE) :
ACTIVE(PI0) contributes:
Qprop-(NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE),FLOW-RATE(PI0))
ACTIVE(PI1) contributes:
Qprop(NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,F),QPE),FLOW-RATE(PI1))

NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE) :
ACTIVE(PI0) contributes:
Qprop(NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE),FLOW-RATE(PI0))
ACTIVE(PI1) contributes:
Qprop-(NET-INFLUENCE(AMOUNT-OF-IN(WATER,LIQUID,G),QPE),FLOW-RATE(PI1))

FLOW-RATE(PI1) :
ACTIVE(PI1) contributes:
Qprop(FLOW-RATE(PI1),PRESSURE(C-S(WATER,LIQUID,G)))
Qprop-(FLOW-RATE(PI1),PRESSURE(C-S(WATER,LIQUID,F)))

FLOW-RATE(PI0) :
ACTIVE(PI0) contributes:
Qprop(FLOW-RATE(PI0),PRESSURE(C-S(WATER,LIQUID,F)))
Qprop-(FLOW-RATE(PI0),PRESSURE(C-S(WATER,LIQUID,G)))

LEVEL(C-S(WATER,LIQUID,G)) :
CONTAINED-LIQUID(C-S(WATER,LIQUID,G)) contributes:
Qprop(LEVEL(C-S(WATER,LIQUID,G)),AMOUNT-OF(C-S(WATER,LIQUID,G)))

PRESSURE(C-S(WATER,LIQUID,G)) :
CONTAINED-LIQUID(C-S(WATER,LIQUID,G)) contributes:
Qprop(PRESSURE(C-S(WATER,LIQUID,G)),LEVEL(C-S(WATER,LIQUID,G)))

TEMPERATURE(C-S(WATER,LIQUID,G)) :
PHYSOB(C-S(WATER,LIQUID,G)) contributes:
Qprop(TEMPERATURE(C-S(WATER,LIQUID,G)),HEAT(C-S(WATER,LIQUID,G)))

AMOUNT-OF(C-S(WATER,LIQUID,G)) :
ACTIVE(VI1) contributes:
Qprop(AMOUNT-OF(C-S(WATER,LIQUID,G)),AMOUNT-OF-IN(WATER,LIQUID,G))

LEVEL(C-S(WATER,LIQUID,F)) :
CONTAINED-LIQUID(C-S(WATER,LIQUID,F)) contributes:
Qprop(LEVEL(C-S(WATER,LIQUID,F)),AMOUNT-OF(C-S(WATER,LIQUID,F)))

PRESSURE(C-S(WATER,LIQUID,F)) :
CONTAINED-LIQUID(C-S(WATER,LIQUID,F)) contributes:
Qprop(PRESSURE(C-S(WATER,LIQUID,F)),LEVEL(C-S(WATER,LIQUID,F)))

TEMPERATURE(C-S(WATER,LIQUID,F)) :
PHYSOB(C-S(WATER,LIQUID,F)) contributes:

Qprop(TEMPERATURE(C-S(WATER,LIQUID,F)),HEAT(C-S(WATER,LIQUID,F)))

AMOUNT-OF(C-S(WATER,LIQUID,F)) :
ACTIVE(VI0) contributes:
Qprop(AMOUNT-OF(C-S(WATER,LIQUID,F)),AMOUNT-OF-IN(WATER,LIQUID,F))

Appendix B: QPE Envisionment of Fatigue and Fracture

QPE envisionment for fatigue and fracture domain
12:2:3 11-23-1990
QPE (version 2.6 Beta) Envisionment
ATMoSphere version 3.2 Beta, ATMS version 21
Macintosh Allegro Common Lisp, 1.3.2
unspecified

Envisoner results, QPE envisionment

30 situations, in 24 classes
1 Domain files:
HD:QPE files:DOMAINS:FF.lisp
1 Example files:
HD:QPE files:EXAMPLES:FFex1.lisp
No user augments declared.

QPE MODES
Careful mode OFF
Zero Ds values printed.
Display situations by differences
Preconditions assumed.
Add assumptions for resolving pairs of direct influences
Discontinuous direct influences allowed.
Derivative Augments: ON
Simultaneous changes to and from =: ON.
Equality Change Law, Case 2: ON
Unstable transitions not allowed.
Asymptotic approach states pruned.

LIMIT ANALYSIS STATISTICS
30 situations, 114 potential transitions (Max = 15, Min = 0, Ave =3.8)
Before Continuity: 184 potential transitions (Max = 25, Min = 1, Ave =12.)
Before ECL: 119 potential transitions (Max = 16, Min = 1, Ave =7.4)
Final result: 111 transitions (Max = 15, Min = 1, Ave =6.9)

ADB STATISTICS
database items: 501
database classes: 2470
rules run: 262
C-rules run: 0
TMS nodes: 543
TMS classes: 253
Nogoods: 81
assumptions: 26

environments: 300
Justifications: 872
Distribution of Nogoods:

Length	# nogoods
2	32
3	1
4	18
6	30

81 nogoods in all

---- View and Process Instances ----
View instances:
VI0 = THRU-THICKNESS-CRACK(STEEL)
VI1 = EDGE-CRACK(STEEL)
VI2 = PENNY/SURFACE-CRACK(STEEL)
VI3 = FRACTURE-FAILURE(STEEL)
VI4 = FATIGUE-FAILURE(STEEL)
Process instances:
PI0 = TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)
PI1 = HORIZONTAL-FATIGUE(STEEL,VCRACK-VARYING,TEMP-VARYING)
PI2 = VERTICAL-FATIGUE(STEEL,HCRACK-VARYING,TEMP-VARYING)

---- Union of Quantity Space information ----
From Quantity Conditions (4):
A[VERTICAL-DIMENSION(STEEL)] A[VERTICAL-MAX(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)] A[HORIZONTAL-MAX(STEEL)]
A[K(STEEL)] A[KC(STEEL)]
A[DELTAK(STEEL)] A[KT(STEEL)]
From Influence Resolution (4):
D[KC(STEEL)] ZERO
A[NET-INFLUENCE(KC(STEEL),QPE)] ZERO
A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]
ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)] ZERO
From Limit Analysis (1):
D[K(STEEL)] D[KC(STEEL)]
Derived (7):
D[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)] ZERO
D[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]
ZERO
D[NET-INFLUENCE(KC(STEEL),QPE)] ZERO
D[DELTAK(STEEL)] D[KT(STEEL)]
D[K(STEEL)] D[KC(STEEL)]
D[HORIZONTAL-DIMENSION(STEEL)] D[HORIZONTAL-MAX(STEEL)]
D[VERTICAL-DIMENSION(STEEL)] D[VERTICAL-MAX(STEEL)]
Total = 16.

--- Summary of Limit Hypotheses ---

LH0:[>=]A[VERTICAL-DIMENSION(STEEL)]
<A[VERTICAL-MAX(STEEL)]-->=
LH1:[>=]A[HORIZONTAL-DIMENSION(STEEL)]
<A[HORIZONTAL-MAX(STEEL)]-->=
LH2:[>=]A[K(STEEL)]
<A[KC(STEEL)]-->=
LH3:[>=]A[DELTAK(STEEL)]
<A[KT(STEEL)]-->=
CLH4: Conjunction of {LH1,LH2}
CLH5: Conjunction of {LH1,LH3}
CLH6: Conjunction of {LH2,LH3}
CLH7: Conjunction of {LH1,LH2,LH3}
CLH8: Conjunction of {LH0,LH2}
CLH9: Conjunction of {LH0,LH3}
CLH10: Conjunction of {LH0,LH2,LH3}
CLH11: Conjunction of {LH0,LH1}
CLH12: Conjunction of {LH0,LH1,LH2}
CLH13: Conjunction of {LH0,LH1,LH3}
CLH14: Conjunction of {LH0,LH1,LH2,LH3}

----- STATE TRANSITION TABLE ------

16 states out of 30 have candidate transitions.

ENV-166 : {LH2:ENV-133}
ENV-169 :
{CLH7:ENV-137;CLH4:ENV-133;CLH5:ENV-135;CLH6:ENV-136;LH1:ENV-166;LH2:ENV-132;LH3:ENV-134}
ENV-252 :
{CLH7:ENV-137;CLH4:ENV-133;CLH5:ENV-135;CLH6:ENV-136;LH2:ENV-132;LH3:ENV-134}
ENV-251 : {CLH5:ENV-135;LH1:ENV-167;LH3:ENV-134}
ENV-250 : {CLH5:ENV-135;LH1:ENV-167;LH3:ENV-134}
ENV-171 :
{CLH7:ENV-137;CLH4:ENV-133;CLH5:ENV-135;CLH6:ENV-136;LH1:ENV-168;LH2:ENV-132;LH3:ENV-134}
ENV-172 :
{CLH10:ENV-137;CLH8:ENV-133;CLH9:ENV-135;CLH6:ENV-125;LH0:ENV-166;LH2:ENV-121;LH3:ENV-123}
ENV-255 :
{CLH10:ENV-137;CLH8:ENV-133;CLH9:ENV-135;CLH6:ENV-125;LH2:ENV-121;LH3:ENV-123}
ENV-254 : {CLH9:ENV-135;LH0:ENV-167;LH3:ENV-123}

ENV-253 : {CLH9:ENV-135;LH0:ENV-167;LH3:ENV-123}
ENV-174 :
{CLH10:ENV-137;CLH8:ENV-133;CLH9:ENV-135;CLH6:ENV-125;LH0:ENV-168;LH2:ENV-121;
LH3:ENV-123}
ENV-175 :
{CLH14:ENV-137;CLH12:ENV-133;CLH13:ENV-135;CLH10:ENV-136;CLH7:ENV-125;CLH11:E
NV-166;CLH8:ENV-132;CLH4:ENV-121;CLH9:ENV-134;CLH5:ENV-123;CLH6:ENV-124;LH0:E
NV-169;LH1:ENV-172;LH2:ENV-120;LH3:ENV-122}
ENV-258 :
{CLH14:ENV-137;CLH12:ENV-133;CLH13:ENV-135;CLH10:ENV-136;CLH7:ENV-125;CLH8:EN
V-132;CLH4:ENV-121;CLH9:ENV-134;CLH5:ENV-123;CLH6:ENV-124;LH0:ENV-252;LH1:ENV-
255;LH2:ENV-120;LH3:ENV-122}
ENV-257 :
{CLH13:ENV-135;CLH11:ENV-167;CLH9:ENV-134;CLH5:ENV-123;LH0:ENV-251;LH1:ENV-254
;LH3:ENV-122}
ENV-256 :
{CLH13:ENV-135;CLH11:ENV-167;CLH9:ENV-134;CLH5:ENV-123;LH0:ENV-250;LH1:ENV-253
;LH3:ENV-122}
ENV-177 :
{CLH14:ENV-137;CLH12:ENV-133;CLH13:ENV-135;CLH10:ENV-136;CLH7:ENV-125;CLH11:E
NV-168;CLH8:ENV-132;CLH4:ENV-121;CLH9:ENV-134;CLH5:ENV-123;CLH6:ENV-124;LH0:E
NV-171;LH1:ENV-174;LH2:ENV-120;LH3:ENV-122}

Sclass S27, 1 situations:
Status = R-COMPLETE, Duration = INTERVAL
IS:{QPE}
VS: VI2: PENNY/SURFACE-CRACK(STEEL)
PS: PI0: TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)
PI1: HORIZONTAL-FATIGUE(STEEL,VCRACK-VARYING,TEMP-VARYING)
PI2: VERTICAL-FATIGUE(STEEL,HCRACK-VARYING,TEMP-VARYING)

-*- Ds Values -*-
Ds[DELTAK(STEEL)]=1
Ds[HORIZONTAL-DIMENSION(STEEL)]=1
Ds[HORIZONTAL-MAX(STEEL)]=0
Ds[K(STEEL)]=1
Ds[KC(STEEL)]=0
Ds[KT(STEEL)]=0
Ds[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]=1
Ds[NET-INFLUENCE(KC(STEEL),QPE)]=0
Ds[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]=1
Ds[TEMPERATURE(STEEL)]=0
Ds[VERTICAL-DIMENSION(STEEL)]=1
Ds[VERTICAL-MAX(STEEL)]=0

-*- Environments -*-

Env ENV-177:
D[KC(STEEL)]=ZERO
A[NET-INFLUENCE(KC(STEEL),QPE)]=ZERO
A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]>ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]>ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

Sclass S26, 3 situations:
Status = R-COMPLETE, Duration = INTERVAL
IS:{QPE}
VS: VI2: PENNY/SURFACE-CRACK(STEEL)
PS: PI0: TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)
PI1: HORIZONTAL-FATIGUE(STEEL,VCRACK-VARYING,TEMP-VARYING)
PI2: VERTICAL-FATIGUE(STEEL,HCRACK-VARYING,TEMP-VARYING)

-*- Ds Values -*-
Ds[DELTAK(STEEL)]=1
Ds[HORIZONTAL-DIMENSION(STEEL)]=1
Ds[HORIZONTAL-MAX(STEEL)]=0
Ds[K(STEEL)]=1
Ds[KC(STEEL)]=1
Ds[KT(STEEL)]=0
Ds[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]=1
Ds[NET-INFLUENCE(KC(STEEL),QPE)]=0
Ds[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]=1
Ds[TEMPERATURE(STEEL)]=0
Ds[VERTICAL-DIMENSION(STEEL)]=1
Ds[VERTICAL-MAX(STEEL)]=0

-*- Environments -*-
In common:
Env ENV-176:
D[KC(STEEL)]>ZERO
A[NET-INFLUENCE(KC(STEEL),QPE)]>ZERO
A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]>ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]>ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Differences: (3)
ENV-222 + ENV-176 = ENV-256
Env ENV-222:

```
 D[K(STEEL)]<D[KC(STEEL)]
ENV-178 + ENV-176 = ENV-257
Env ENV-178:
 D[K(STEEL)]=D[KC(STEEL)]
ENV-209 + ENV-176 = ENV-258
Env ENV-209:
 D[K(STEEL)]>D[KC(STEEL)]

Sclass S25, 1 situations:
  Status = R-COMPLETE, Duration = INTERVAL
  IS:{QPE}
   VS: VI2: PENNY/SURFACE-CRACK(STEEL)
   PS: PI0: TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)
     PI1: HORIZONTAL-FATIGUE(STEEL,VCRACK-VARYING,TEMP-VARYING)
     PI2: VERTICAL-FATIGUE(STEEL,HCRACK-VARYING,TEMP-VARYING)

    -*- Ds Values -*-
  Ds[DELTAK(STEEL)]=1
  Ds[HORIZONTAL-DIMENSION(STEEL)]=1
  Ds[HORIZONTAL-MAX(STEEL)]=0
  Ds[K(STEEL)]=1
  Ds[KC(STEEL)]=-1
  Ds[KT(STEEL)]=0
  Ds[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]=1
  Ds[NET-INFLUENCE(KC(STEEL),QPE)]=0
  Ds[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]=1
  Ds[TEMPERATURE(STEEL)]=0
  Ds[VERTICAL-DIMENSION(STEEL)]=1
  Ds[VERTICAL-MAX(STEEL)]=0

    -*- Environments -*-

 Env ENV-175:
  D[KC(STEEL)]<ZERO
  A[NET-INFLUENCE(KC(STEEL),QPE)]<ZERO
  A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]>ZERO
  A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]>ZERO
  A[DELTAK(STEEL)]<A[KT(STEEL)]
  A[K(STEEL)]<A[KC(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

Sclass S24, 1 situations:
  Status = R-COMPLETE, Duration = INTERVAL
  IS:{QPE}
   VS: VI1: EDGE-CRACK(STEEL)
   PS: PI0: TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)
```

```
    PI2: VERTICAL-FATIGUE(STEEL,HCRACK-VARYING,TEMP-VARYING)

    -*- Ds Values -*-
  Ds[DELTAK(STEEL)]=1
  Ds[HORIZONTAL-DIMENSION(STEEL)]=0
  Ds[HORIZONTAL-MAX(STEEL)]=0
  Ds[K(STEEL)]=1
  Ds[KC(STEEL)]=0
  Ds[KT(STEEL)]=0
  Ds[NET-INFLUENCE(KC(STEEL),QPE)]=0
  Ds[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]=1
  Ds[TEMPERATURE(STEEL)]=0
  Ds[VERTICAL-DIMENSION(STEEL)]=1
  Ds[VERTICAL-MAX(STEEL)]=0

    -*- Environments -*-

 Env ENV-174:
  D[KC(STEEL)]=ZERO
  A[NET-INFLUENCE(KC(STEEL),QPE)]=ZERO
  A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
  A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]>ZERO
  A[DELTAK(STEEL)]<A[KT(STEEL)]
  A[K(STEEL)]<A[KC(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

 Sclass S23, 3 situations:
   Status = R-COMPLETE, Duration = INTERVAL
   IS:{QPE}
    VS: VI1: EDGE-CRACK(STEEL)
    PS: PI0: TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)
      PI2: VERTICAL-FATIGUE(STEEL,HCRACK-VARYING,TEMP-VARYING)

    -*- Ds Values -*-
  Ds[DELTAK(STEEL)]=1
  Ds[HORIZONTAL-DIMENSION(STEEL)]=0
  Ds[HORIZONTAL-MAX(STEEL)]=0
  Ds[K(STEEL)]=1
  Ds[KC(STEEL)]=1
  Ds[KT(STEEL)]=0
  Ds[NET-INFLUENCE(KC(STEEL),QPE)]=0
  Ds[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]=1
  Ds[TEMPERATURE(STEEL)]=0
  Ds[VERTICAL-DIMENSION(STEEL)]=1
  Ds[VERTICAL-MAX(STEEL)]=0
```

```
    -*- Environments -*-
In common:
 Env ENV-173:
  D[KC(STEEL)]>ZERO
  A[NET-INFLUENCE(KC(STEEL),QPE)]>ZERO
  A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
  A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]>ZERO
  A[DELTAK(STEEL)]<A[KT(STEEL)]
  A[K(STEEL)]<A[KC(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)
Differences: (3)
 ENV-222 + ENV-173 = ENV-253
 Env ENV-222:
  D[K(STEEL)]<D[KC(STEEL)]
 ENV-178 + ENV-173 = ENV-254
 Env ENV-178:
  D[K(STEEL)]=D[KC(STEEL)]
 ENV-209 + ENV-173 = ENV-255
 Env ENV-209:
  D[K(STEEL)]>D[KC(STEEL)]

Sclass S22, 1 situations:
   Status = R-COMPLETE, Duration = INTERVAL
   IS:{QPE}
    VS: VI1: EDGE-CRACK(STEEL)
    PS: PI0: TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)
       PI2: VERTICAL-FATIGUE(STEEL,HCRACK-VARYING,TEMP-VARYING)

      -*- Ds Values -*-
   Ds[DELTAK(STEEL)]=1
   Ds[HORIZONTAL-DIMENSION(STEEL)]=0
   Ds[HORIZONTAL-MAX(STEEL)]=0
   Ds[K(STEEL)]=1
   Ds[KC(STEEL)]=-1
   Ds[KT(STEEL)]=0
   Ds[NET-INFLUENCE(KC(STEEL),QPE)]=0
   Ds[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]=1
   Ds[TEMPERATURE(STEEL)]=0
   Ds[VERTICAL-DIMENSION(STEEL)]=1
   Ds[VERTICAL-MAX(STEEL)]=0

      -*- Environments -*-

Env ENV-172:
 D[KC(STEEL)]<ZERO
 A[NET-INFLUENCE(KC(STEEL),QPE)]<ZERO
```

A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]>ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

Sclass S2, 1 situations:
Status = R-COMPLETE, Duration = INTERVAL
IS:{QPE}
VS: VI2: PENNY/SURFACE-CRACK(STEEL)
VI3: FRACTURE-FAILURE(STEEL)
No active processes.

-*- Ds Values -*-
Ds[DELTAK(STEEL)]=0
Ds[HORIZONTAL-DIMENSION(STEEL)]=0
Ds[HORIZONTAL-MAX(STEEL)]=0
Ds[K(STEEL)]=0
Ds[KC(STEEL)]=0
Ds[KT(STEEL)]=0
Ds[TEMPERATURE(STEEL)]=0
Ds[VERTICAL-DIMENSION(STEEL)]=0
Ds[VERTICAL-MAX(STEEL)]=0

-*- Environments -*-

Env ENV-120:
A[NET-INFLUENCE(KC(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]=A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

Sclass S3, 1 situations:
Status = R-COMPLETE, Duration = INTERVAL
IS:{QPE}
VS: VI1: EDGE-CRACK(STEEL)
VI3: FRACTURE-FAILURE(STEEL)
No active processes.

-*- Ds Values -*-
Ds[DELTAK(STEEL)]=0
Ds[HORIZONTAL-DIMENSION(STEEL)]=0

Ds[HORIZONTAL-MAX(STEEL)]=0
Ds[K(STEEL)]=0
Ds[KC(STEEL)]=0
Ds[KT(STEEL)]=0
Ds[TEMPERATURE(STEEL)]=0
Ds[VERTICAL-DIMENSION(STEEL)]=0
Ds[VERTICAL-MAX(STEEL)]=0

-*- Environments -*-

Env ENV-121:
A[NET-INFLUENCE(KC(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]=A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

Sclass S4, 1 situations:
Status = R-COMPLETE, Duration = INTERVAL
IS:{QPE}
VS: VI2: PENNY/SURFACE-CRACK(STEEL)
VI4: FATIGUE-FAILURE(STEEL)
No active processes.

-*- Ds Values -*-
Ds[DELTAK(STEEL)]=0
Ds[HORIZONTAL-DIMENSION(STEEL)]=0
Ds[HORIZONTAL-MAX(STEEL)]=0
Ds[K(STEEL)]=0
Ds[KC(STEEL)]=0
Ds[KT(STEEL)]=0
Ds[TEMPERATURE(STEEL)]=0
Ds[VERTICAL-DIMENSION(STEEL)]=0
Ds[VERTICAL-MAX(STEEL)]=0

-*- Environments -*-

Env ENV-122:
A[NET-INFLUENCE(KC(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
A[DELTAK(STEEL)]=A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]

```
 ENFORCE(QUANTITY-EXISTENCE)

Sclass S5, 1 situations:
  Status = R-COMPLETE, Duration = INTERVAL
  IS:{QPE}
   VS: VI1: EDGE-CRACK(STEEL)
      VI4: FATIGUE-FAILURE(STEEL)
      No active processes.

     -*- Ds Values -*-
  Ds[DELTAK(STEEL)]=0
  Ds[HORIZONTAL-DIMENSION(STEEL)]=0
  Ds[HORIZONTAL-MAX(STEEL)]=0
  Ds[K(STEEL)]=0
  Ds[KC(STEEL)]=0
  Ds[KT(STEEL)]=0
  Ds[TEMPERATURE(STEEL)]=0
  Ds[VERTICAL-DIMENSION(STEEL)]=0
  Ds[VERTICAL-MAX(STEEL)]=0

     -*- Environments -*-

 Env ENV-123:
  A[NET-INFLUENCE(KC(STEEL),QPE)]??ZERO
  A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
  A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
  A[DELTAK(STEEL)]=A[KT(STEEL)]
  A[K(STEEL)]<A[KC(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

Sclass S6, 1 situations:
  Status = R-COMPLETE, Duration = INTERVAL
  IS:{QPE}
   VS: VI2: PENNY/SURFACE-CRACK(STEEL)
      VI3: FRACTURE-FAILURE(STEEL)
      VI4: FATIGUE-FAILURE(STEEL)
      No active processes.

     -*- Ds Values -*-
  Ds[DELTAK(STEEL)]=0
  Ds[HORIZONTAL-DIMENSION(STEEL)]=0
  Ds[HORIZONTAL-MAX(STEEL)]=0
  Ds[K(STEEL)]=0
  Ds[KC(STEEL)]=0
  Ds[KT(STEEL)]=0
  Ds[TEMPERATURE(STEEL)]=0
```

```
 Ds[VERTICAL-DIMENSION(STEEL)]=0
 Ds[VERTICAL-MAX(STEEL)]=0

    -*- Environments -*-

Env ENV-124:
 A[NET-INFLUENCE(KC(STEEL),QPE)]??ZERO
 A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
 A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
 A[DELTAK(STEEL)]=A[KT(STEEL)]
 A[K(STEEL)]=A[KC(STEEL)]
 A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
 A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
 ENFORCE(QUANTITY-EXISTENCE)

Sclass S7, 1 situations:
  Status = R-COMPLETE, Duration = INTERVAL
  IS:{QPE}
   VS: VI1: EDGE-CRACK(STEEL)
      VI3: FRACTURE-FAILURE(STEEL)
      VI4: FATIGUE-FAILURE(STEEL)
      No active processes.

    -*- Ds Values -*-
 Ds[DELTAK(STEEL)]=0
 Ds[HORIZONTAL-DIMENSION(STEEL)]=0
 Ds[HORIZONTAL-MAX(STEEL)]=0
 Ds[K(STEEL)]=0
 Ds[KC(STEEL)]=0
 Ds[KT(STEEL)]=0
 Ds[TEMPERATURE(STEEL)]=0
 Ds[VERTICAL-DIMENSION(STEEL)]=0
 Ds[VERTICAL-MAX(STEEL)]=0

    -*- Environments -*-

Env ENV-125:
 A[NET-INFLUENCE(KC(STEEL),QPE)]??ZERO
 A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
 A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
 A[DELTAK(STEEL)]=A[KT(STEEL)]
 A[K(STEEL)]=A[KC(STEEL)]
 A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
 A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
 ENFORCE(QUANTITY-EXISTENCE)

Sclass S21, 1 situations:
  Status = R-COMPLETE, Duration = INTERVAL
```

IS:{QPE}
VS: VI0: THRU-THICKNESS-CRACK(STEEL)
PS: PI0: TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)
PI1: HORIZONTAL-FATIGUE(STEEL,VCRACK-VARYING,TEMP-VARYING)

-*- Ds Values -*-
Ds[DELTAK(STEEL)]=1
Ds[HORIZONTAL-DIMENSION(STEEL)]=1
Ds[HORIZONTAL-MAX(STEEL)]=0
Ds[K(STEEL)]=1
Ds[KC(STEEL)]=0
Ds[KT(STEEL)]=0
Ds[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]=1
Ds[NET-INFLUENCE(KC(STEEL),QPE)]=0
Ds[TEMPERATURE(STEEL)]=0
Ds[VERTICAL-DIMENSION(STEEL)]=0
Ds[VERTICAL-MAX(STEEL)]=0

-*- Environments -*-

Env ENV-171:
D[KC(STEEL)]=ZERO
A[NET-INFLUENCE(KC(STEEL),QPE)]=ZERO
A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]>ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

Sclass S20, 3 situations:
Status = R-COMPLETE, Duration = INTERVAL
IS:{QPE}
VS: VI0: THRU-THICKNESS-CRACK(STEEL)
PS: PI0: TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)
PI1: HORIZONTAL-FATIGUE(STEEL,VCRACK-VARYING,TEMP-VARYING)

-*- Ds Values -*-
Ds[DELTAK(STEEL)]=1
Ds[HORIZONTAL-DIMENSION(STEEL)]=1
Ds[HORIZONTAL-MAX(STEEL)]=0
Ds[K(STEEL)]=1
Ds[KC(STEEL)]=1
Ds[KT(STEEL)]=0
Ds[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]=1
Ds[NET-INFLUENCE(KC(STEEL),QPE)]=0
Ds[TEMPERATURE(STEEL)]=0

```
  Ds[VERTICAL-DIMENSION(STEEL)]=0
  Ds[VERTICAL-MAX(STEEL)]=0

     -*- Environments -*-
In common:
 Env ENV-170:
  D[KC(STEEL)]>ZERO
  A[NET-INFLUENCE(KC(STEEL),QPE)]>ZERO
  A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]>ZERO
  A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
  A[DELTAK(STEEL)]<A[KT(STEEL)]
  A[K(STEEL)]<A[KC(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)
Differences: (3)
 ENV-222 + ENV-170 = ENV-250
 Env ENV-222:
  D[K(STEEL)]<D[KC(STEEL)]
 ENV-178 + ENV-170 = ENV-251
 Env ENV-178:
  D[K(STEEL)]=D[KC(STEEL)]
 ENV-209 + ENV-170 = ENV-252
 Env ENV-209:
  D[K(STEEL)]>D[KC(STEEL)]

Sclass S19, 1 situations:
   Status = R-COMPLETE, Duration = INTERVAL
   IS:{QPE}
    VS: VI0: THRU-THICKNESS-CRACK(STEEL)
    PS: PI0: TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)
       PI1: HORIZONTAL-FATIGUE(STEEL,VCRACK-VARYING,TEMP-VARYING)

     -*- Ds Values -*-
  Ds[DELTAK(STEEL)]=1
  Ds[HORIZONTAL-DIMENSION(STEEL)]=1
  Ds[HORIZONTAL-MAX(STEEL)]=0
  Ds[K(STEEL)]=1
  Ds[KC(STEEL)]=-1
  Ds[KT(STEEL)]=0
  Ds[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]=1
  Ds[NET-INFLUENCE(KC(STEEL),QPE)]=0
  Ds[TEMPERATURE(STEEL)]=0
  Ds[VERTICAL-DIMENSION(STEEL)]=0
  Ds[VERTICAL-MAX(STEEL)]=0

     -*- Environments -*-
```

```
Env ENV-169:
 D[KC(STEEL)]<ZERO
 A[NET-INFLUENCE(KC(STEEL),QPE)]<ZERO
 A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]>ZERO
 A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
 A[DELTAK(STEEL)]<A[KT(STEEL)]
 A[K(STEEL)]<A[KC(STEEL)]
 A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
 A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
 ENFORCE(QUANTITY-EXISTENCE)

Sclass S18, 1 situations:
  Status = R-COMPLETE, Duration = INTERVAL
  IS:{QPE}
   VS: VI0: THRU-THICKNESS-CRACK(STEEL)
       VI1: EDGE-CRACK(STEEL)
   PS: PI0: TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)

     -*- Ds Values -*-
  Ds[DELTAK(STEEL)]=0
  Ds[HORIZONTAL-DIMENSION(STEEL)]=0
  Ds[HORIZONTAL-MAX(STEEL)]=0
  Ds[K(STEEL)]=0
  Ds[KC(STEEL)]=0
  Ds[KT(STEEL)]=0
  Ds[NET-INFLUENCE(KC(STEEL),QPE)]=0
  Ds[TEMPERATURE(STEEL)]=0
  Ds[VERTICAL-DIMENSION(STEEL)]=0
  Ds[VERTICAL-MAX(STEEL)]=0

     -*- Environments -*-

Env ENV-168:
 D[KC(STEEL)]=ZERO
 A[NET-INFLUENCE(KC(STEEL),QPE)]=ZERO
 A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
 A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
 A[DELTAK(STEEL)]<A[KT(STEEL)]
 A[K(STEEL)]<A[KC(STEEL)]
 A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
 A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
 ENFORCE(QUANTITY-EXISTENCE)

Sclass S17, 1 situations:
  Status = R-COMPLETE, Duration = INTERVAL
  IS:{QPE}
   VS: VI0: THRU-THICKNESS-CRACK(STEEL)
       VI1: EDGE-CRACK(STEEL)
```

```
   PS: PI0: TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)

      -*- Ds Values -*-
  Ds[DELTAK(STEEL)]=0
  Ds[HORIZONTAL-DIMENSION(STEEL)]=0
  Ds[HORIZONTAL-MAX(STEEL)]=0
  Ds[K(STEEL)]=0
  Ds[KC(STEEL)]=1
  Ds[KT(STEEL)]=0
  Ds[NET-INFLUENCE(KC(STEEL),QPE)]=0
  Ds[TEMPERATURE(STEEL)]=0
  Ds[VERTICAL-DIMENSION(STEEL)]=0
  Ds[VERTICAL-MAX(STEEL)]=0

      -*- Environments -*-

 Env ENV-167:
  D[KC(STEEL)]>ZERO
  A[NET-INFLUENCE(KC(STEEL),QPE)]>ZERO
  A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
  A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
  A[DELTAK(STEEL)]<A[KT(STEEL)]
  A[K(STEEL)]<A[KC(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

Sclass S16, 1 situations:
   Status = R-COMPLETE, Duration = INTERVAL
   IS:{QPE}
    VS: VI0: THRU-THICKNESS-CRACK(STEEL)
        VI1: EDGE-CRACK(STEEL)
    PS: PI0: TEMPERATURE-EFFECT(STEEL,VCRACK-VARYING,HCRACK-VARYING)

      -*- Ds Values -*-
  Ds[DELTAK(STEEL)]=0
  Ds[HORIZONTAL-DIMENSION(STEEL)]=0
  Ds[HORIZONTAL-MAX(STEEL)]=0
  Ds[K(STEEL)]=0
  Ds[KC(STEEL)]=-1
  Ds[KT(STEEL)]=0
  Ds[NET-INFLUENCE(KC(STEEL),QPE)]=0
  Ds[TEMPERATURE(STEEL)]=0
  Ds[VERTICAL-DIMENSION(STEEL)]=0
  Ds[VERTICAL-MAX(STEEL)]=0

      -*- Environments -*-
```

Env ENV-166:
D[KC(STEEL)]<ZERO
A[NET-INFLUENCE(KC(STEEL),QPE)]<ZERO
A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

Sclass S10, 1 situations:
Status = R-COMPLETE, Duration = INTERVAL
IS:{QPE}
VS: VI0: THRU-THICKNESS-CRACK(STEEL)
VI3: FRACTURE-FAILURE(STEEL)
No active processes.

-*- Ds Values -*-
Ds[DELTAK(STEEL)]=0
Ds[HORIZONTAL-DIMENSION(STEEL)]=0
Ds[HORIZONTAL-MAX(STEEL)]=0
Ds[K(STEEL)]=0
Ds[KC(STEEL)]=0
Ds[KT(STEEL)]=0
Ds[TEMPERATURE(STEEL)]=0
Ds[VERTICAL-DIMENSION(STEEL)]=0
Ds[VERTICAL-MAX(STEEL)]=0

-*- Environments -*-

Env ENV-132:
A[NET-INFLUENCE(KC(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]=A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

Sclass S11, 1 situations:
Status = R-COMPLETE, Duration = INTERVAL
IS:{QPE}
VS: VI0: THRU-THICKNESS-CRACK(STEEL)
VI1: EDGE-CRACK(STEEL)
VI3: FRACTURE-FAILURE(STEEL)
No active processes.

```
    -*- Ds Values -*-
 Ds[DELTAK(STEEL)]=0
 Ds[HORIZONTAL-DIMENSION(STEEL)]=0
 Ds[HORIZONTAL-MAX(STEEL)]=0
 Ds[K(STEEL)]=0
 Ds[KC(STEEL)]=0
 Ds[KT(STEEL)]=0
 Ds[TEMPERATURE(STEEL)]=0
 Ds[VERTICAL-DIMENSION(STEEL)]=0
 Ds[VERTICAL-MAX(STEEL)]=0

    -*- Environments -*-

Env ENV-133:
 A[NET-INFLUENCE(KC(STEEL),QPE)]??ZERO
 A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
 A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
 A[DELTAK(STEEL)]<A[KT(STEEL)]
 A[K(STEEL)]=A[KC(STEEL)]
 A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
 A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
 ENFORCE(QUANTITY-EXISTENCE)

Sclass S12, 1 situations:
  Status = R-COMPLETE, Duration = INTERVAL
  IS:{QPE}
   VS: VI0: THRU-THICKNESS-CRACK(STEEL)
      VI4: FATIGUE-FAILURE(STEEL)
      No active processes.

    -*- Ds Values -*-
 Ds[DELTAK(STEEL)]=0
 Ds[HORIZONTAL-DIMENSION(STEEL)]=0
 Ds[HORIZONTAL-MAX(STEEL)]=0
 Ds[K(STEEL)]=0
 Ds[KC(STEEL)]=0
 Ds[KT(STEEL)]=0
 Ds[TEMPERATURE(STEEL)]=0
 Ds[VERTICAL-DIMENSION(STEEL)]=0
 Ds[VERTICAL-MAX(STEEL)]=0

    -*- Environments -*-

Env ENV-134:
 A[NET-INFLUENCE(KC(STEEL),QPE)]??ZERO
 A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
 A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
```

A[DELTAK(STEEL)]=A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

Sclass S13, 1 situations:
Status = R-COMPLETE, Duration = INTERVAL
IS:{QPE}
VS: VI0: THRU-THICKNESS-CRACK(STEEL)
VI1: EDGE-CRACK(STEEL)
VI4: FATIGUE-FAILURE(STEEL)
No active processes.

-*- Ds Values -*-
Ds[DELTAK(STEEL)]=0
Ds[HORIZONTAL-DIMENSION(STEEL)]=0
Ds[HORIZONTAL-MAX(STEEL)]=0
Ds[K(STEEL)]=0
Ds[KC(STEEL)]=0
Ds[KT(STEEL)]=0
Ds[TEMPERATURE(STEEL)]=0
Ds[VERTICAL-DIMENSION(STEEL)]=0
Ds[VERTICAL-MAX(STEEL)]=0

-*- Environments -*-

Env ENV-135:
A[NET-INFLUENCE(KC(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
A[DELTAK(STEEL)]=A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

Sclass S14, 1 situations:
Status = R-COMPLETE, Duration = INTERVAL
IS:{QPE}
VS: VI0: THRU-THICKNESS-CRACK(STEEL)
VI3: FRACTURE-FAILURE(STEEL)
VI4: FATIGUE-FAILURE(STEEL)
No active processes.

-*- Ds Values -*-
Ds[DELTAK(STEEL)]=0
Ds[HORIZONTAL-DIMENSION(STEEL)]=0

Ds[HORIZONTAL-MAX(STEEL)]=0
Ds[K(STEEL)]=0
Ds[KC(STEEL)]=0
Ds[KT(STEEL)]=0
Ds[TEMPERATURE(STEEL)]=0
Ds[VERTICAL-DIMENSION(STEEL)]=0
Ds[VERTICAL-MAX(STEEL)]=0

-*- Environments -*-

Env ENV-136:
A[NET-INFLUENCE(KC(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
A[DELTAK(STEEL)]=A[KT(STEEL)]
A[K(STEEL)]=A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

Sclass S15, 1 situations:
Status = R-COMPLETE, Duration = INTERVAL
IS:{QPE}
VS: VI0: THRU-THICKNESS-CRACK(STEEL)
VI1: EDGE-CRACK(STEEL)
VI3: FRACTURE-FAILURE(STEEL)
VI4: FATIGUE-FAILURE(STEEL)
No active processes.

-*- Ds Values -*-
Ds[DELTAK(STEEL)]=0
Ds[HORIZONTAL-DIMENSION(STEEL)]=0
Ds[HORIZONTAL-MAX(STEEL)]=0
Ds[K(STEEL)]=0
Ds[KC(STEEL)]=0
Ds[KT(STEEL)]=0
Ds[TEMPERATURE(STEEL)]=0
Ds[VERTICAL-DIMENSION(STEEL)]=0
Ds[VERTICAL-MAX(STEEL)]=0

-*- Environments -*-

Env ENV-137:
A[NET-INFLUENCE(KC(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE)]??ZERO
A[NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE)]??ZERO
A[DELTAK(STEEL)]=A[KT(STEEL)]
A[K(STEEL)]=A[KC(STEEL)]

A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

There are 15 limit hypotheses, 4 singletons and 11 conjunctive.

LH0:[>=]A[VERTICAL-DIMENSION(STEEL)]
<A[VERTICAL-MAX(STEEL)]-->=
Start: (ENV-259)
End: (ENV-107)
Env ENV-259:
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-107:
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

LH1:[>=]A[HORIZONTAL-DIMENSION(STEEL)]
<A[HORIZONTAL-MAX(STEEL)]-->=
Start: (ENV-262)
End: (ENV-109)
Env ENV-262:
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-109:
A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

LH2:[>=]A[K(STEEL)]
<A[KC(STEEL)]-->=
Start: (ENV-279 ENV-278 ENV-277 ENV-276 ENV-275 ENV-274 ENV-273 ENV-272 ENV-271 ENV-270 ENV-269 ENV-268 ENV-267 ENV-266 ENV-265)
End: (ENV-111)
Env ENV-279:
D[K(STEEL)]>D[KC(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-278:
D[KC(STEEL)]<ZERO
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

Env ENV-277:
D[KC(STEEL)]<ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-276:
D[KC(STEEL)]<ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-275:
D[KC(STEEL)]<ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-274:
D[KC(STEEL)]=ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-273:
D[KC(STEEL)]=ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-272:
D[KC(STEEL)]=ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-271:
A[NET-INFLUENCE(KC(STEEL),QPE)]<ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]

ENFORCE(QUANTITY-EXISTENCE)
Env ENV-270:
A[NET-INFLUENCE(KC(STEEL),QPE)]<ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-269:
A[NET-INFLUENCE(KC(STEEL),QPE)]=ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-268:
A[NET-INFLUENCE(KC(STEEL),QPE)]<ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-267:
A[NET-INFLUENCE(KC(STEEL),QPE)]=ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-266:
A[NET-INFLUENCE(KC(STEEL),QPE)]<ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-265:
A[NET-INFLUENCE(KC(STEEL),QPE)]=ZERO
A[DELTAK(STEEL)]<A[KT(STEEL)]
A[K(STEEL)]<A[KC(STEEL)]
A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)
Env ENV-111:
A[K(STEEL)]=A[KC(STEEL)]
ENFORCE(QUANTITY-EXISTENCE)

LH3:[>=]A[DELTAK(STEEL)]

```
        <A[KT(STEEL)]-->=
   Start: (ENV-282 ENV-281 ENV-280)
     End: (ENV-113)
 Env ENV-282:
  A[DELTAK(STEEL)]<A[KT(STEEL)]
  A[K(STEEL)]<A[KC(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)
 Env ENV-281:
  A[DELTAK(STEEL)]<A[KT(STEEL)]
  A[K(STEEL)]<A[KC(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)
 Env ENV-280:
  A[DELTAK(STEEL)]<A[KT(STEEL)]
  A[K(STEEL)]<A[KC(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]<A[HORIZONTAL-MAX(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]<A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)
 Env ENV-113:
  A[DELTAK(STEEL)]=A[KT(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

CLH4: Conjunction of {LH1,LH2}
 LH1:[>=]A[HORIZONTAL-DIMENSION(STEEL)]
        <A[HORIZONTAL-MAX(STEEL)]-->=
 LH2:[>=]A[K(STEEL)]
        <A[KC(STEEL)]-->=
    End: (ENV-264)
 Env ENV-264:
  A[K(STEEL)]=A[KC(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

CLH5: Conjunction of {LH1,LH3}
 LH1:[>=]A[HORIZONTAL-DIMENSION(STEEL)]
        <A[HORIZONTAL-MAX(STEEL)]-->=
 LH3:[>=]A[DELTAK(STEEL)]
        <A[KT(STEEL)]-->=
    End: (ENV-263)
 Env ENV-263:
  A[DELTAK(STEEL)]=A[KT(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

CLH6: Conjunction of {LH2,LH3}
```

```
 LH2:[>=]A[K(STEEL)]
        <A[KC(STEEL)]-->=
 LH3:[>=]A[DELTAK(STEEL)]
        <A[KT(STEEL)]-->=
    End: (ENV-295)
 Env ENV-295:
  A[DELTAK(STEEL)]=A[KT(STEEL)]
  A[K(STEEL)]=A[KC(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

CLH7: Conjunction of {LH1,LH2,LH3}
 LH1:[>=]A[HORIZONTAL-DIMENSION(STEEL)]
        <A[HORIZONTAL-MAX(STEEL)]-->=
 LH2:[>=]A[K(STEEL)]
        <A[KC(STEEL)]-->=
 LH3:[>=]A[DELTAK(STEEL)]
        <A[KT(STEEL)]-->=
    End: (ENV-296)
 Env ENV-296:
  A[DELTAK(STEEL)]=A[KT(STEEL)]
  A[K(STEEL)]=A[KC(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

CLH8: Conjunction of {LH0,LH2}
 LH0:[>=]A[VERTICAL-DIMENSION(STEEL)]
        <A[VERTICAL-MAX(STEEL)]-->=
 LH2:[>=]A[K(STEEL)]
        <A[KC(STEEL)]-->=
    End: (ENV-261)
 Env ENV-261:
  A[K(STEEL)]=A[KC(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

CLH9: Conjunction of {LH0,LH3}
 LH0:[>=]A[VERTICAL-DIMENSION(STEEL)]
        <A[VERTICAL-MAX(STEEL)]-->=
 LH3:[>=]A[DELTAK(STEEL)]
        <A[KT(STEEL)]-->=
    End: (ENV-260)
 Env ENV-260:
  A[DELTAK(STEEL)]=A[KT(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

CLH10: Conjunction of {LH0,LH2,LH3}
 LH0:[>=]A[VERTICAL-DIMENSION(STEEL)]
```

```
        <A[VERTICAL-MAX(STEEL)]-->=
 LH2:[>=]A[K(STEEL)]
        <A[KC(STEEL)]-->=
 LH3:[>=]A[DELTAK(STEEL)]
        <A[KT(STEEL)]-->=
    End: (ENV-297)
 Env ENV-297:
  A[DELTAK(STEEL)]=A[KT(STEEL)]
  A[K(STEEL)]=A[KC(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

CLH11: Conjunction of {LH0,LH1}
 LH0:[>=]A[VERTICAL-DIMENSION(STEEL)]
        <A[VERTICAL-MAX(STEEL)]-->=
 LH1:[>=]A[HORIZONTAL-DIMENSION(STEEL)]
        <A[HORIZONTAL-MAX(STEEL)]-->=
    End: (ENV-298)
 Env ENV-298:
  A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

CLH12: Conjunction of {LH0,LH1,LH2}
 LH0:[>=]A[VERTICAL-DIMENSION(STEEL)]
        <A[VERTICAL-MAX(STEEL)]-->=
 LH1:[>=]A[HORIZONTAL-DIMENSION(STEEL)]
        <A[HORIZONTAL-MAX(STEEL)]-->=
 LH2:[>=]A[K(STEEL)]
        <A[KC(STEEL)]-->=
    End: (ENV-299)
 Env ENV-299:
  A[K(STEEL)]=A[KC(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
  A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
  ENFORCE(QUANTITY-EXISTENCE)

CLH13: Conjunction of {LH0,LH1,LH3}
 LH0:[>=]A[VERTICAL-DIMENSION(STEEL)]
        <A[VERTICAL-MAX(STEEL)]-->=
 LH1:[>=]A[HORIZONTAL-DIMENSION(STEEL)]
        <A[HORIZONTAL-MAX(STEEL)]-->=
 LH3:[>=]A[DELTAK(STEEL)]
        <A[KT(STEEL)]-->=
    End: (ENV-300)
 Env ENV-300:
  A[DELTAK(STEEL)]=A[KT(STEEL)]
  A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
```

```
 A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
 ENFORCE(QUANTITY-EXISTENCE)

CLH14: Conjunction of {LH0,LH1,LH2,LH3}
 LH0:[>=]A[VERTICAL-DIMENSION(STEEL)]
       <A[VERTICAL-MAX(STEEL)]-->=
 LH1:[>=]A[HORIZONTAL-DIMENSION(STEEL)]
       <A[HORIZONTAL-MAX(STEEL)]-->=
 LH2:[>=]A[K(STEEL)]
       <A[KC(STEEL)]-->=
 LH3:[>=]A[DELTAK(STEEL)]
       <A[KT(STEEL)]-->=
   End: (ENV-301)
 Env ENV-301:
 A[DELTAK(STEEL)]=A[KT(STEEL)]
 A[K(STEEL)]=A[KC(STEEL)]
 A[HORIZONTAL-DIMENSION(STEEL)]=A[HORIZONTAL-MAX(STEEL)]
 A[VERTICAL-DIMENSION(STEEL)]=A[VERTICAL-MAX(STEEL)]
 ENFORCE(QUANTITY-EXISTENCE)
```

Table of possible direct influencers.

```
 VERTICAL-DIMENSION(STEEL) :
    ACTIVE(PI2) contributes:
    I+(VERTICAL-DIMENSION(STEEL),A[DELTAK(STEEL)])

 HORIZONTAL-DIMENSION(STEEL) :
    ACTIVE(PI1) contributes:
    I+(HORIZONTAL-DIMENSION(STEEL),A[DELTAK(STEEL)])

 KC(STEEL) :
    ACTIVE(PI0) contributes:
    I+(KC(STEEL),A[TEMPERATURE(STEEL)])
```

Table of possible indirect influencers.

```
 NET-INFLUENCE(KC(STEEL),QPE) :
    ACTIVE(PI0) contributes:
    Qprop(NET-INFLUENCE(KC(STEEL),QPE),TEMPERATURE(STEEL))

 NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE) :
    ACTIVE(PI1) contributes:
    Qprop(NET-INFLUENCE(HORIZONTAL-DIMENSION(STEEL),QPE),DELTAK(STEEL))

 NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE) :
    ACTIVE(PI2) contributes:
    Qprop(NET-INFLUENCE(VERTICAL-DIMENSION(STEEL),QPE),DELTAK(STEEL))
```

DELTAK(STEEL) :
  METAL(STEEL) contributes:
  Qprop(DELTAK(STEEL),HORIZONTAL-DIMENSION(STEEL))
  Qprop(DELTAK(STEEL),VERTICAL-DIMENSION(STEEL))

K(STEEL) :
  METAL(STEEL) contributes:
  Qprop(K(STEEL),VERTICAL-DIMENSION(STEEL))
  Qprop(K(STEEL),HORIZONTAL-DIMENSION(STEEL))

USERS GUIDE TO MACINTOSH QUALITATIVE REASONING

Michael Chien 12/90

Two systems (QPE and QSIM) are available for use. These systems will only work for Macintosh Allegro Common Lisp v 1.3.2 running with 4M RAM.

Parameters

Version 2.0 of MACL is due out in 1991 but an upgrade is **not** recommended. Menus, windows, and graphics are built around the object-oriented system of v1.3.2. Version 2.0 uses CLOS (Common Lisp Object System) and is not compatible with the object-oriented system of v1.3.2.

Memory is adjustable although 4M of memory seems to work well and 2M of memory is the absolute minimum.

Filenames should **not** be changed for any folders related to QPE or QSIM and the hard disk name should also not be changed.

It's probably a good idea to restart MACL if you're switching between QSIM and QPE

Files and Folders

1. HD:Qualitative Simulation

   contains MACL startup info and a menu to select QPE or QSIM for startup.

2. HD:QPE files

3. HD:QSIM files

Starting up the systems

1. **Open** the file *Do It* in HD:Qualitative Simulation
2. **Select** from the pull-down menu either QPE or QSIM

Running QSIM

1. **Load** the file that contains the model.
2. **Evaluate** the lisp function that corresponds to the initial state
3. Follow the menu that appears if the model is consistent and the state limit is not reached.

    Behavior tree plot - **type** *t*

    Single behavior plots - **type** <space> or *n* (for next)

    Var-slice plots - **type** *o* (for other) and then **type v** (for var-slice) and **enter** the variable name - all functions work properly for the next menu except for "change layout"

The other functions probably don't work such as phase-slice and any "change layout" function.

Parameters

1. The library of examples can be run from the menu.

    Select the group of demos

    Select the exact example from the pull-down menu

    Select the initialization from the menu that appears on the far right

2. Printing

To get a hardcopy of the plots:

Select from the *Output* menu "Postscript" and then get the plots that you want.

Copy the plot file (your named file) to a disk

Run "Send PS in the Cricket Graph folder" and **download** the file to the LaserWriter

It's not possible to get both screen output and hard output simultaneously.

The default is screen output

3. QSIM parameters

(qsim-cleanup) - not a bad idea to do this once in a while to clean up internal lisp memory

state-limit - this is the maximum number of states that QSIM uses

Final notes

There are 4 manuals. The most useful is QSIM User's Manual. Both the QSIM Maintainers Guide and The Postscript Facility are useful if any changes need to be made to QSIM itself.

Addresses: Ben Kuipers kuipers@cs.utexas.edu (512) 471-9561

David Throop throop@cs.utexas.edu (512) 471-9559

(Maintains QSIM)

Adam Farquhar farquhar@cs.utexas.edu (512) 471-9561 (QPC)

Jimmy Crawford (QPC), David Franke (misc), Bess Sullivan(secretary)

Running QPE

1. **Type** (qpe::qpe)

2. **Change** the domain file and the scenario file to the ones that you want.

3. **Envision** (Obtains all of the possible outcomes)

4. Results can be obtained by **Report** (lisp menu choice)

Notes

Menus in QPE- **?** Whatever help is available, **q** quits, **0** redisplay choices, **some integer** performs the function for that integer.

Most things in QPE behave as documented. The documentation for it could be better.

There's one manual.

Addresses

Ken Forbus forbus@cs.uiuc.edu or forbus@ils.nwu.edu

Old phone at UIUC (217) 333-0193

John Collins Very involved with QPE collins@cs.uiuc.edu (217) 244-1372

Gordon Skorstad skorstad@cs.uiuc.edu (217) 244-1372

Johan de Kleer dekleer.pa@xerox.com (Wrote the ATMS) (415) 494-4709

Appendix D: Error Correction in QSIM

Date: Tue, 2 Oct 90 10:43:34 CDT
From: kuipers@cs.utexas.edu
Posted-Date: Tue, 2 Oct 90 10:43:34 CDT
Message-Id: <9010021543.AA29819@ai.cs.utexas.edu>
Received: by ai.cs.utexas.edu (5.59/1.4-Client)
id AA29819; Tue, 2 Oct 90 10:43:34 CDT
To: Chien@csvax.cs.ukans.edu
In-Reply-To: Chien@csvax.cs.ukans.edu's message of Mon, 1 Oct 90 16:07:29 CDT
<9010011607.aa05231@csvax.cs.ukans.edu>
Subject: Modeling in QSIM
Status: R

Mike and Kim,
>Hi. I don't think that I understand the infinity and asymptotic approach that
>QSIM takes. I am trying to model fatigue behavior in metals. The model is as
>follows:
>
```
>(define-QDE crack-model
> (text "Basic qualitative fatigue and fracture behavior.")
> (quantity-spaces
>           (deltaK                (0 kth kt inf))
>           (da/dN                 (0 inf))
>           (a                     (0 a_i thickness)))
>
> (history a)
>
> (constraints
>              ((M+ a deltaK)                    )
>           ((M+ deltaK da/dN)     (kth 0)(kt inf) )
>            ((D/DT a da/dN)                    ))
>
> (print-names (deltaK   "stress intensity range")
>              (da/dN    "crack growth rate")
>              (a        "crack length"))
>
> (layout
>          (nil deltaK)
>          (nil da/dN)
>          (nil a))
> )

>; Model fatigue behavior.
>
>(defun model-FFB ()
> (let ((initial-state
```

```
>          (make-initial-state crack-model
>                              '(
>                        (a (a_i inc))          ;assume an initial crack
>                        )
>                              )))
>   (qsim initial-state)
>  (qsim-display initial-state)
>   ))
```

>The part that I don't understand is why the state

```
>          (when deltaK = [kt inc] and a = [thickness inc]) is being pruned.
```

>This state is being pruned by the rule t < inf -> [f(t) = inf -> f'(t) = inf]
>by the function apply-time-is-finite. I know this because when I trace global
>filters, it tells me "TIME LABEL (finite) is inconsistent with S-6".
>It was suggested that I model this occurance that I want by using region transitions or by using
>time-scale abstraction. But I don't understand why
>this state is being pruned.

Here is the corrected code. Thanks for finding the bug, and your patience on the phone while I fixed it.

Ben

```
(defun apply-time-is-finite (state)
 (cond ((every     #'(lambda (qvalue)
                       (let* ((qval (cdr qvalue))
                              (var  (qval-variable qval)))
                         (cond ((eq  (qmag qval) *inf-lmark*)
                                (let ((explicit-deriv (explicit-derivative-qmag var state)))
                                  (or (null explicit-deriv)
                                      (eql explicit-deriv *inf-lmark*))))
                               ((eq  (qmag qval) *minf-lmark*)
                                (let ((explicit-deriv (explicit-derivative-qmag var state)))
                                  (or (null explicit-deriv)
                                      (eql explicit-deriv *minf-lmark*))))
                               (t t))))
                   (cdr (state-qvalues state)))   ; skip over time qval
          state)
         (t (if *trace-time-label*
                (format *QSIM-Trace* "~%TIME LABEL (finite) is inconsistent with ~a."
                        (state-name state)))
            (prune-inconsistent-state state "with finite time")
            nil)))

; fixed bug:  if explicit derivative does not exist, state is not (yet) inconsistent. (-BJK)
```

Appendix E: Notes on Porting QPE to MACL

1) I added some system-defining facilities in the walker.lisp file in ATMS
   by adding the hooks provided in a later version of PCL.
2) In ADB, made *user-package* and *lisp-package* definitions correct for MACL.
   (defsys.lisp)
3) In ADB (db.lisp), I commented out the line
   ; (declare(special *number-of-classes))
   I do not think this is valid by itself. Maybe, proclaim instead of declare
   is correct in this situation.
4) In QPE2P6 (events.lisp):function run-qpe-events, I commented out the line
   ; (unless entry (return-from EXECUTE-EVENT-THUNKS))
   I do not think that this line is syntactically correct and couldn't
   figure out what was trying to be done with this line.
5) In QPE2P6 (bench.lisp), Commented out all of the garbage-collection
   "features".

Appendix F: Notes on Porting QSIM to MACL

Mike Chien August 1990

This version of QSIM is written for MACL v1.3.2 under finder 6.1.5 and system 6.0.5. The loop macro is needed for running. All graphics and menus are written using Object Lisp and not using CLOS. Many parts of the code were written for my particular machine i.e. some pathnames are hard-coded.

This version is clearly "as is" and is provided through the University of Kansas, Civil Engineering Department.

Any questions on this version can be directed toward
Mike Chien chien@csvax.cs.ukans.edu

This version of QSIM was written for Macintosh Allegro Common Lisp version 1.3.2. MACL uses Object Lisp and does not use CLOS. CLOS will be available with MACL 2.0 due out at the end of 1990. This version has many dependencies on Object Lisp. Windows, menus and graphics are all done with Object Lisp. Also, the loop.lisp extension is needed.

All files are in source code form. If this version is to be used, it would greatly help in efficiency to compile the files and to use the compiled version.

There is a read-me file in the postscript section. Most of the changes for MACL are in this section of QSIM. There was only one significant change in the q section of QSIM. This is in structures.lisp. This is in the ldb-test macro. A comma was removed from in front of the global variables. MyFrontEnd.lisp provides the catalog of examples through a menu and control of the output. Any other changes to the files were inconsequential (i.e. commenting out lines that caused errors).

To prepare to use this version of QSIM. Go through the postscript read-me file and make any necessary changes. It is also necessary to make the files locateable and to make a directory for postscript output to be sent. These changes are in mac-system.lisp in the postscript directory and in mac-system.lisp in the q directory and in the init.lisp file at the top-level. If the catalog section (located in MyFrontEnd.lisp) is to be used, then changes need to be made in this section also.

To run this version of qsim, you can open the init.lisp file directly or from inside MACL. There must be a change to the qsim package (in-package :qsim) to run QSIM. Then, you can select from the available catalog of examples or load files and run them individually.
The only other option implemented is whether the output goes to the screen or to a postscript output file. This option is selected through a menu item. The default is to the screen.