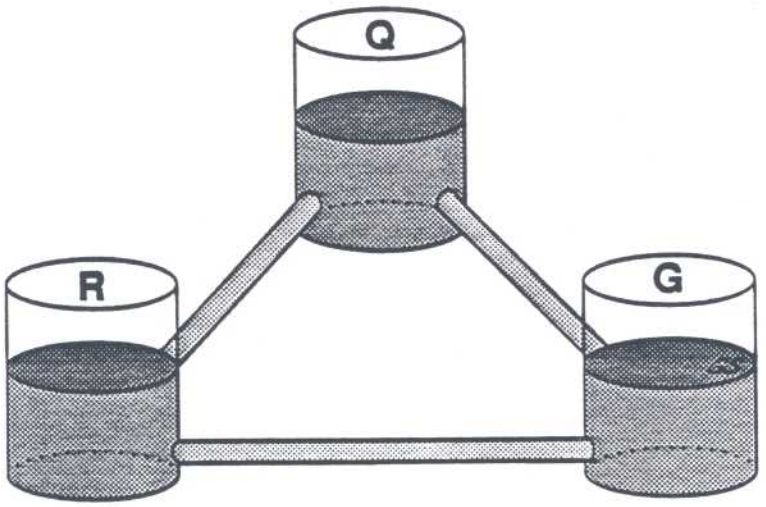
DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN



QUALITATIVE REASONING GROUP

REPORT NO. UIUCDCS-R-90-1572 UILU-ENG-90-1710

DYNAMIC ACROSS-TIME MEASUREMENT INTERPRETATION: MAINTAINING QUALITATIVE UNDERSTANDINGS OF PHYSICAL SYSTEM BEHAVIOR

by

Dennis Martin DeCoste

February 1990

DYNAMIC ACROSS-TIME MEASUREMENT INTERPRETATION: MAINTAINING QUALITATIVE UNDERSTANDINGS OF PHYSICAL SYSTEM BEHAVIOR

BY

DENNIS MARTIN DECOSTE

B.S., University of Illinois at Urbana-Champaign, 1986

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1990

Urbana, Illinois

DYNAMIC ACROSS-TIME MEASUREMENT INTERPRETATION: MAINTAINING QUALITATIVE UNDERSTANDINGS OF PHYSICAL SYSTEM BEHAVIOR

Dennis Martin DeCoste, M.S.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1990
Kenneth D. Forbus, Advisor

Incrementally maintaining a qualitative understanding of physical system behavior based on observations is crucial to real-time process monitoring, control, and diagnosis. This paper describes the DATMI theory for dynamically maintaining a *pinterp-space*, a concise representation of local and global interpretations consistent with the observations over time. Each interpretation signifies alternative paths of states in a qualitative envisionment. Representing a space of interpretations, instead of just a "current best" one, avoids the need for extensive backtracking to handle incomplete or faulty data. Domain-specific knowledge about state and transition probabilities can be used to maintain the best working interpretation as well. Domain-specific knowledge about durations of states and paths of states can also be used to further constrain the interpretation space. When all these constraints lead to inconsistencies, faulty-data hypotheses are generated and then tested by adjusting the pinterp-space. The time and space complexity of maintaining the pinterp-space is polynomial in the number of measurements and envisionment states.

ACKNOWLEDGMENTS

Foremost, I would like to thank my advisor, Ken Forbus, for his much-appreciated encouragement, guidance, and perspective during this research. The diverse, yet highly-interactive, research environment which he has developed for his Qualitative Reasoning Group has greatly facilitated this work. Most notably, I have benefitted greatly from John Collins' work on modelling physical systems and from Brian Falkenhainer's use of my DATMI system in his own work.

Warm thanks are extended to past and present members of QRG for many enlightening and entertaining discussions and for graciously enduring my frequent ramblings. Special thanks go to Brian Falkenhainer, for his inspirations, enthusiasm, and frequent "pep talks", and to John Collins and Gordon Skorstad for their many useful suggestions.

I am also grateful for the comradery of my other officemates, especially Scott Bennett, Steve Chien, and Melinda Gervasio, for making work much more fun and rewarding. And for other friends outside of the lab, especially Alan Yang, Ken Chang, and Maria "Ching" Muyot, for making play much more fun and rewarding.

I also wish to deeply thank my family for their love and support over the years in all that I do. Thanks, Mom and Dad. Thanks, Don and Dave.

This research was supported by the Office of Naval Research, Contract Number N00014-85-K-0225, and by a University of Illinois fellowship.

TABLE OF CONTENTS

1 INTRODUCTION 1

1.1 Thesis 1

1.2 The Role of Qualitative Physics 2

1.3 Overview of DATMI Theory 3

1.4 Overview of This Paper 5

2 MAINTAINING THE OBSERVATIONAL HISTORY 7

2.1 Representing the Observational History 9

2.2 Specifying Observations and Measurements 11

2.3 Converting Measurements into Qualitative Properties 11

2.4 Creating, Merging, and Splitting Global Segments 13

3 MAINTAINING THE PINTERP-SPACE 17

3.1 Consistency Constraints and Classes of Pinterps 18

3.2 Types of Transition Consistency Relations 19

3.3 Representing Transition Consistency Relations With Dependency Paths 21

3.3.1 Caching Exactly Two Dependency Paths Per Pinterp Is Best 21

3.3.2 Some Example Dependency Paths 21

3.4 Maintaining Pinterp Dependencies 23

3.4.1 Processing a New Segment 23

3.4.2 Processing an Updated Segment 23

3.4.3 Propagating the Effects of Pinterp Filterings 26

3.5 Path-Cost Maintenance 26

3.6 Global Interpretation Construction 30

3.7 Finding Dependency Paths 30
3.7.1 Finding Frontier-State, Spanning-State, and Meeting-States Paths 32
3.7.2 Finding Gap-filling Paths 32
3.7.3 Finding Hidden-Transition Paths 33
3.7.4 Improving Hidden-Transition Search 34

4 ADJUSTING THE PINTERP-SPACE TO HANDLE FAULTY DATA . . 36
4.1 Faulty Data 37
4.2 Generating Possible Fixes to the Pinterp-Space 37
4.2.1 Sensor Failure Fix-Hypotheses 38
4.2.2 Conversion Failure Fix-Hypotheses 40
4.3 Applying a Fix-Hypothesis To the Pinterp-Space 41
4.3.1 Propagating Path-Cost Decreases 42
4.3.2 Retracting a Fix-Hypothesis 46
4.4 Improving Fix-Hypothesis Generation 48

5 MAINTAINING INTERPRETATION CREDIBILITIES 50
5.1 Determining Path-Probabilities 50
5.1.1 Using State and Transition Probabilities 51
5.1.2 Using Property Probabilities 52
5.1.3 Locally Composing Path-Probabilities 53
5.2 Normalizing Probabilistic Interpretation Credibilities 54
5.2.1 The Normalization Procedure 55
5.2.2 Issues in Normalizing the Pinterp-Space 56

6 USING DURATION CONSTRAINTS 58
6.1 Representing Duration Estimates 58
6.2 Estimating Durations 59
6.3 Applying Duration Constraints to the Pinterp-Space 60
6.3.1 Checking Spanning-State Dependency Paths 61
6.3.2 Checking Hidden-Transition Dependency Paths 62
6.3.3 Checking Gap-Filling Dependency Paths 64

6.4 Problems with DATMI's Duration Reasoning 64
6.4.1 Incompleteness 64
6.4.2 Unsoundness 65

7 COMPLEXITY ANALYSIS 66
7.1 Definitions 66
7.2 The Size of the Interpretation Space 67
7.3 DATMI Space Complexity 67
7.4 DATMI Time Complexity 67
7.4.1 Determining the Effects of Property Constraints 68
7.4.2 Maintaining the Dependency Paths 68
7.5 Reducing DATMI Time and Space Complexities 69

8 DISCUSSION 71
8.1 Summary 71
8.2 Related Work 72
8.2.1 Qualitative Physics 72
8.2.2 Script-Based Reasoning 74
8.2.3 Connectionism and Pattern Recognition 75
8.3 Future Work 75
8.3.1 Modelling 75
8.3.2 Temporal Reasoning 76
8.3.3 Compact Pinterp-Spaces 77
8.3.4 Active Data Acquisition And Data Selection 78
8.3.5 Handling Faulty Data 79
8.3.6 Reasoning Under Uncertainty 79
8.3.7 Parallel Algorithms 80
8.4 Conclusions 80

REFERENCES 81

APPENDIX 85

A THE QPE PUMP-CYCLE TOTAL ENVISIONMENT 85

B DATMI EXAMPLES FOR THE QP PUMP-CYCLE SYSTEM 88

B.1 Handling Sensor Failures with Property Adjustments 88

B.1.1 Example 1: Failure of the Pump Indicator 88

B.1.2 Example 2: Miscalibrated-Sensors for Water Levels 95

B.2 Handling Sensor Failures with Property Probabilities 104

B.2.1 Example 3: An Unlikely Failure of the Pump Indicator 104

B.3 Noticing State-Duration Conflicts 107

B.3.1 Example 4: Drainage Taking Longer Than Expected 107

C THE FROB SINGLE-WELL TOTAL ENVISIONMENT 109

D DATMI EXAMPLES FOR THE FROB SINGLE-WELL SYSTEM 112

D.1 Maintaining the Most-Probable Interpretation 112

D.1.1 Example 5: Three Collisions 112

D.2 Noticing Reach-Duration Conflicts 117

D.2.1 Example 6: Three Collisions – With Duration Constraints 117

LIST OF FIGURES

1.1 DATMI modules and representations 4

2.1 Alternative history representations for the same behavior 8
2.2 Example quantity-space conversion tables 12
2.3 Translating measurements to CHANGE properties 13
2.4 Translating measurements to ORDER properties 14
2.5 Splitting segments when asserting a property 15
2.6 Merging segments to maintain a concise history 16

3.1 Examples of the five DATMI transition consistency relations 20
3.2 Example of pinterp dependencies 22
3.3 Primitive DATMI definitions 24
3.4 Procedure CREATE-PINTERPS 25
3.5 Procedure REFINE-PINTERPS 25
3.6 Procedure PROPAGATE-FILTERING-EFFECTS 26
3.7 Procedure PROPAGATE-FILTERING-EFFECTS (using path-costs) 28
3.8 Procedure PROPAGATE-PATH-COST-INCREASES 28
3.9 Procedure PROPAGATE-INCREASES-TO-DEPENDENTS 29
3.10 Procedure FIND-DEPENDENCY-PATH 31

4.1 Example of a sensor failure fix-hypothesis 39
4.2 Procedure TRY-FORGETTING-FIX-HYPOTHESIS 42
4.3 Procedure PROPAGATE-NEW-PINTERPS 43
4.4 Procedure BATCH-UPDATE-PINTERP-SPACE 44
4.5 Procedure PROPAGATE-PINTERP-ACTIVATION 45

4.6 Procedure INSTALL-LOWER-COST-DEPENDENCIES 46
4.7 Procedure RETRACT-FIX-HYPOTHESIS 47

5.1 Example of a probabilistic envisionment 51
5.2 Example pinterp-space with path-probabilities 53
5.3 Normalizing the example pinterp-space 56

6.1 Example spanning of state S_s over many segments 61
6.2 Example hidden-transition b-dependency path 63

7.1 Example problem with concise dependencies 70

A.1 The pump-cycle scenario 85
A.2 Summary of state properties for the pump-cycle system 86
A.3 State-transition diagram for the pump-cycle system 87

B.1 Portion of the pump-cycle envisionment 89

C.1 The single-well scenario 110
C.2 State-transition diagram for the single-well system 111

Chapter 1

INTRODUCTION

The problem of interpreting observations of a system over time is fundamental to intelligent reasoning about the physical world. Diagnosis, for example, requires the ability to determine whether a system is operating as expected. Diagnosis also involves determining whether predicted consequences of hypothesized faults could account for the original observations. Monitoring the execution of planned operations requires determining if actions actually have the desired effects. Process control demands, in real-time, an understanding of what is, or might be, happening in the physical system. Finally, model refinement requires determining when and how the predictions of the current model might conflict with the observations.

Several factors make such interpretation tasks difficult:

- An appropriate system model is required to meaningfully interpret the observations. For tracking behavior over time, this model must indicate the temporal ordering constraints on system behavior.
- Typically, the available sensors provide only incomplete, possibly even faulty, data about the state of the system at each time. There may be many alternative interpretations consistent with such incomplete data – and they might all be incorrect if some data are faulty.
- Measurements must be carefully translated into the language of the model. Such translations should strive to minimize the effects of faulty data while preserving useful data.
- Real-time interpretation of incomplete or faulty data requires the ability to incrementally re-interpret, without excessive backtracking, as observations are made.

1.1 Thesis

This paper presents the *Dynamic Across-Time Measurement Interpretation* (DATMI) theory for interpreting observations. It is based on a representation, called the *pinterp-space* (possible interpretation space), that indicates which system states might be occurring at each time according to the observations and the model. Each path of states across time represents an alternative interpretation. This pinterp-space stores the best current global interpretation and efficiently supports queries about alternative global interpretations and partial interpretations.

By indicating every state which might be occurring at each time, instead of just indicating a few consistent sequences of states over time, the pinterp-space concisely represents the entire consistent interpretation space. DATMI uses this implicit representation of the entire interpretation space to:

1. Detect whether a particular state S could have occurred at a particular time t – this is especially useful for monitoring tasks.
2. Interpret incomplete (potentially garden-path) observations – by using the least-commitment strategy of not discounting alternative consistent interpretations.

By dynamically maintaining the pinterp-space as observations are obtained, DATMI also:

1. Quickly provides a best working interpretation at all times.
2. Detects faulty observations as soon as inconsistencies arise.
3. Handles faulty observations by adjusting the pinterp-space to reflect changes in belief for some of the observations.

DATMI can also use the following domain-specific information:

1. Duration estimates for states and paths of states – which provide additional constraint on the pinterp-space.
2. Likelihood or desirability estimates for states and state transitions – that indicate which interpretations are best.
3. Probabilistic distributions over the states possibly indicated by a sensor reading – which help avoid inconsistencies due to faulty data.

DATMI views observations as constraints on the interpretation space of possible behaviors implied by the system model. When the interpretation space implied by the model is large and only incomplete observations are available, an enormous number of interpretations could be consistent. Thus, explicitly generating all and only such valid interpretations is generally intractable and undesirable. The pinterp-space tractably represents an interpretation space which implicitly contains all consistent interpretations of the given observations while not covering any misinterpretations. Local constraint-propagation techniques are shown to be sufficient to ensure the consistency of the pinterp-space.

Although the size of the interpretation space can be exponential in the number of system states, the entire DATMI algorithm involves only polynomial complexity. It requires space which is at most quadratic in the number of system states and linear in the number of observations. Furthermore, processing time is at most cubic in the number of states and quadratic in the number of observations. In practice, processing time is often close to linear – making real-time interpretation feasible.

1.2 The Role of Qualitative Physics

Qualitative physics is well-suited for specifying models for interpretation problems since it stresses the causality and relevancy issues necessary for meaningfully explaining physical behavior. An overview of past and current work in qualitative physics can be found in (Forbus, 1988).

Qualitative simulation generates predictions of what qualitative changes a physical system might undergo. Such simulations can provide an interpreter with the necessary expectations of what behaviors are possible.

A qualitative simulator, unlike a traditional numerical simulator, provides *qualitative states* which represent distinct states of the system's variables. These qualitative states summarize system behavior at some relevant level of detail. Furthermore, qualitative physics is based on the notion of *composibility* – that one can qualitatively simulate the physical system based on general-purpose domain models and additional knowledge of the organization of a particular system. Such composibility is useful for automated generation of models for specific physical systems and for refining models using machine learning.

DATMI relies on the concept of a *total envisionment* (Forbus, 1984):

Definition 1.1 (Total envisionment) *A* total envisionment *represents all the possible qualitative* states *of a system and all the possible* transitions *from one state to another. A* path *represents a sequence of states connected by these transitions.*

Thus, a total envisionment indicates all possible behaviors of the system. Alternatively, an *attainable envisionment* represents only those states and transitions that arise from some initial state.

Although attainable envisioners exist (Forbus, 1981; Kuipers, 1986), the ability to reason about total envisionments is important for interpretation since one may not know the exact state of the system when observations are first available. Unless otherwise noted, the term "envisionment" indicates a total envisionment.

The Qualitative Process Engine (QPE) (Forbus, 1990) is currently used to provide total envisionments that represent the expectations of the model that the interpreter can use. The specific QP model used for our current DATMI examples is described in (Collins-& Forbus, 1990).

1.3 Overview of DATMI Theory

The *Dynamic Across-Time Measurement Interpretation* (DATMI) theory addresses several aspects of the interpretation problem. It builds upon Forbus' *Across-Time Measurement Interpretation* (ATMI) theory (Forbus, 1986a). Both theories share an underlying theme that interpretation involves finding paths of states (*global interpretations*) through a qualitative envisionment which are consistent with the observations. Any simulation technique providing such envisionments can be used to model the system; thus, both theories are *ontology-independent.*

This section presents the basic DATMI framework and compares it to ATMI. Terminology introduced in this overview will be formalized later. As summarized below, DATMI inherits ATMI's basic approach for converting observations into a representation from which global interpretations are generated. As shown in Figure 1.1, initial observations are first converted into *qualitative property* assertions over time intervals using domain-specific conversion rules. These properties each consist of a *property name* and a qualitative *property value.* These properties correspond to ones describing envisionment states. As these property assertions are gathered, *global segments* are maintained to concisely represent the observations qualitatively.

The states of the envisionment that could possibly occur during each of these segments are indicated by the *pinterp-space.* Initially, all states whose properties agree with a segment's properties are considered possible in that segment. Local constraint-propagation techniques adjust the pinterp-space to eliminate state S as a possible state for global segment G whenever

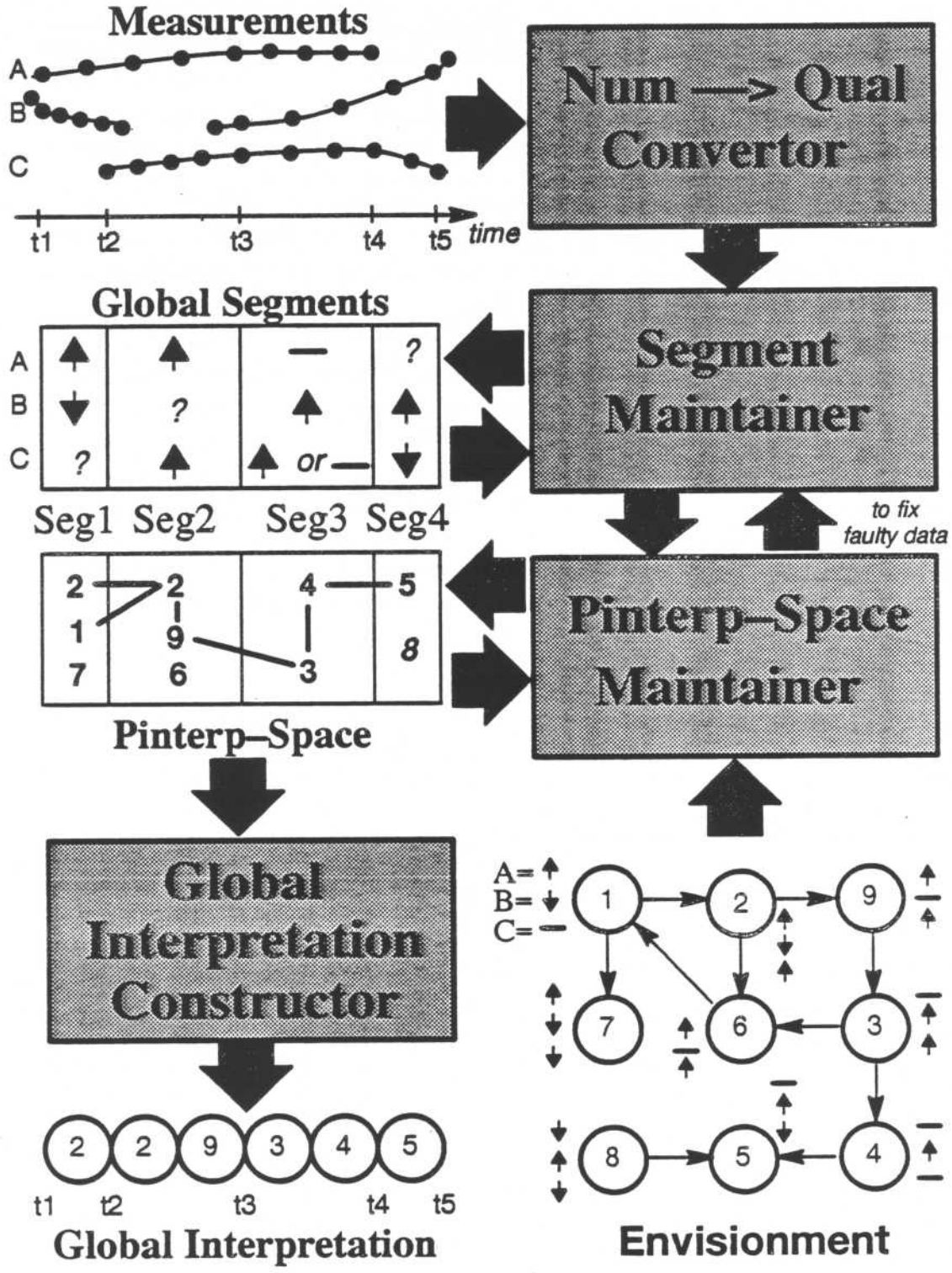


Figure 1.1: DATMI modules and representations

it is inconsistent for S to occur during G. Such inconsistency arises whenever there is no path of possible states of G, using only transitions in the envisionment, which connects S with one of the possible states of a segment temporally adjacent to G. An *inconsistent pinterp-space* results whenever some global segment has no possible states. A pinterp-space being inconsistent indicates that either the observations or the model are faulty.

DATMI extends the ATMI framework in several key ways. First, DATMI dynamically maintains a concise pinterp-space to provide efficient processing and to handle faulty data. Each possible state S of a segment G must have some compatibility relation with each segment temporally adjacent to G. Each relation indicates the (best) path of possible states of G connecting S to some possible state of the adjacent segment. In the pinterp-space of Figure 1.1, for example, the compatibility relation between segment Seg3 and possible state 2 of segment Seg2 indicates a path going through possible state 9 of Seg3 and connecting to possible state 3 of Seg3. Furthermore, state 6 is not a possible state for Seg2 because it cannot have compatibility relations with adjacent segments Seg1 and Seg3.

The (best) working global interpretation can immediately be found at any time by combining these paths across each segment into a global path across all segments. Alternative consistent global interpretations can also be generated or confirmed by searching through this constrained pinterp-space. In contrast, ATMI always searched to find a global interpretation since it lacked the vocabulary of compatibility relations to immediately suggest one.

DATMI addresses the problem of faulty data both at observation time and interpretation time. To reduce the effects of noisy data, DATMI allows conversions from measurements to qualitative data to be conservation (i.e. disjunctive) and probabilistic. Furthermore, DATMI provides a means of recovery when such conversions still lead to an inconsistent pinterp-space. DATMI can dynamically adjust the pinterp-space based on hypotheses of what is wrong: the sensors, the conversions, or the envisionment itself.

DATMI provides other advantages over ATMI as well. It provides efficient techniques for finding *hidden-transition* interpretations and *gap-filling* interpretations to account for incomplete observations. Such interpretations indicate state changes over a single segment that do not seem necessary according to the observations for that segment alone. For example, the global interpretation shown in Figure 1.1 involves a hidden-transition of state 2 followed by state 9 in segment Seg2. Also, as mentioned in Section 1.1, DATMI can use domain-specific duration and probabilistic estimates.

It should be noted up front that some important interpretation problems are not addressed in this work. DATMI assumes that the model (envisionment) is complete and consistent. It ignores the problem of sensor fusion by assuming that only one sensor can indicate the value of a particular system variable at a particular time. It also ignores the problems of active data acquisition (i.e. finding new data to reduce ambiguity) and data selection (i.e. considering the best subset of data that can be processed under current resource constraints). However, the incremental capabilities of DATMI would allow it to incorporate new data or model constraints, or to retract existing ones, at any time.

1.4 Overview of This Paper

Chapter 2 discusses the initial phase of maintaining the observational history. It describes problems and approaches for converting numerical data into concise segments of qualitative

descriptions of the observations, along with how the reliability of sensor readings is taken into account.

Chapter 3 defines the pinterp-space and how it is maintained to concisely represent the possible behaviors consistent with the observations and system model. Furthermore, the notion of *path-costs* is introduced to determine the overall "best" global interpretations.

Chapter 4 explains how faulty data can arise and how they are handled. This chapter also discusses a method for retracting a faulty data hypothesis which is found to be inconsistent with later observations.

Chapter 5 shows how path-costs can be generalized to include probabilistic measures of path likelihood. As shown in this chapter, normalizing these probabilistic measures involves recovering the probabilistic weight that was assigned to interpretations which are inconsistent with the observations. This allows the determination of interpretation likelihoods that are conditional on the observations.

Chapter 6 describes how duration estimates for envisionment states and paths of states can reduce the ambiguity in the pinterp-space. After showing that applying these constraints to the entire pinterp-space is exponentially expensive, it discusses heuristics for applying them to a restricted number of useful cases.

Chapter 7 analyzes the algorithmic complexity of DATMI's maintenance procedures, indicates worst-case complexity, and discusses trade-offs in optimizing expected performance.

Chapter 8 summarizes the DATMI framework, discusses how it relates to other work, and suggests future work.

Finally, the Appendices demonstrate the performance of the LISP DATMI program on various examples.

Chapter 2

MAINTAINING THE OBSERVATIONAL HISTORY

The choice of representation for observations greatly influences the difficulty of interpretation. Overly conservative translations from numerical measurements to qualitative terms can lead to intractably large interpretation spaces. On the other hand, overly precise translations can prevent accurate interpretation. This chapter discusses how DATMI represents observations to try to avoid these two problems. Later chapters show how DATMI recovers from such problems when they still arise.

The behavior of a physical system over time is often represented qualitatively with *parameter histories* (Forbus, 1984; Hayes, 1985):

Definition 2.1 (Parameter history) *A* parameter history *corresponds to a set of predicates describing the relations among object parameters over time. Each* episode *of this history indicates a set of predicates, for a particular spatially-bound collection of objects, whose truth values are constant over the episode's duration. Similarly, an* event *is an episode whose duration is an instant.*

Two episodes *meet* when the time interval of one follows the time interval of the other, with no time in between (Allen, 1983). Each predicate indicates the qualitative status of some system property – such as whether a certain pipe connects two particular containers, whether the water level of one container is greater than, less than, or equal to the water level of another container, or whether the temperature of a furnace is increasing, decreasing, or remaining constant.

The following definition adapts Williams' notion of concise histories (Williams, 1986) in terms of these parameter histories:

Definition 2.2 (Concise history) *A* concise history *merges meeting episodes with identical truth-value assignments for corresponding sets of predicates and globally merges episodes that correspond to identical time intervals for different sets of predicates.*

Figure 2.1 presents several alternative history representations for the same behavior.

In interpretation tasks one must distinguish the *observational history* from the *behavioral history*. The behavioral history corresponds to the traditional use of the term "history":

Definition 2.3 (Behavioral history) *A* behavioral history *represents the actual complete behavior of the physical system over time.*

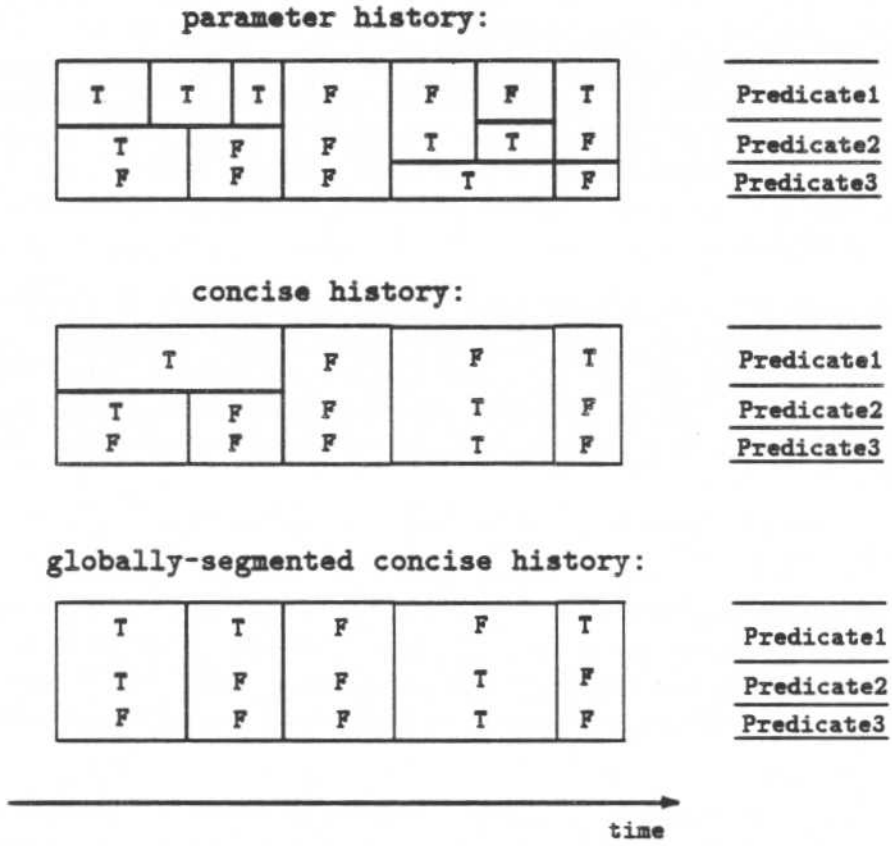


Figure 2.1: Alternative history representations for the same behavior

Boxes represent episodes. The truth values of each predicate on the right are indicated within each box.

Whereas:

Definition 2.4 (Observational history) *An* observational history *represents the observed changes in system variables.*

Thus, unlike a behavioral history, an observational history need not completely specify all the variables. Furthermore, an observational history might even be inconsistent with the behavioral history due to faulty data.

In order to summarize the changes in each predicate over time, while maximizing the temporal extent of individual predicates, the behavioral history is best represented as a concise history. Likewise, an observational history should also be concise. However, a more useful representation for an observational history is a:

Definition 2.5 (Globally-segmented concise history) *A* globally-segmented concise history *is a concise history where episodes never overlap temporally.*

This representation was also used by ATMI. In such a representation, the envisionment states which might be occurring at a particular time can be found by checking which states have properties that are compatible with the property values in the episode covering that time. Forbus formalizes such relationships between histories and total envisionments in (Forbus, 1987).

2.1 Representing the Observational History

Formalizing DATMI's use of globally-segmented concise histories first requires some definitions:

Definition 2.6 (Global segment) *The episodes of globally-segmented observational histories are called* global segments, *or just "segments" for short.*

The function START-TIME(G_g) denotes the time at which segment G_g starts and END-TIME(G_g) denotes the time at which it ends. Furthermore, DURATION(G_g) indicates the length of time over which G_g lasts, where DURATION(G_g) = END-TIME(G_g) - START-TIME(G_g).

Each segment G_g specifies a set of qualitative properties, (SEG-PROPS(G_g)) holding over G's time interval, each defined as follows:

Definition 2.7 (Segment property) *Each* segment property *consists of a* property name, *such as* ORDER(Water-Level(Can1),Water-Level(Can2)), *and a* property value, *such as* GREATER. *The function* PROP-VAL*(P,p) = v denotes property value v for the property named p in the set of properties P.*

Segment properties concisely and more uniformly express the predicates of an episode; a single property with k possible values replaces k different predicates.

The observational history $\mathcal{H}$ is a temporally totally-ordered sequence of these segments, as defined by:

Definition 2.8 (G_l meets G_r) *Two global segments* meet *when the time interval of one follows the time interval of the other, with no time in between. The expression $G_l \mid G_r$ denotes that segment G_l meets G_r.*

Definition 2.9 (G_l leads to G_r) G_l leads to G_r, *denoted $G_l \rightsquigarrow G_r$, exactly when either* END-TIME*(G_l) <* START-TIME*(G_r) or $G_l \mid G_r$.*

Definition 2.10 (Temporally totally-ordered segments) *All segments of $\mathcal{H}$ are* temporally totally-ordered *exactly when, for all pairs of distinct segments G_i and G_j of $\mathcal{H}$, exactly one of the following is true: $G_i \rightsquigarrow G_j$, $G_j \rightsquigarrow G_i$*

Each segment of $\mathcal{H}$ is linked to its ***neighboring segments*** by the following two functions:

Definition 2.11 (B-neighbor) *The backward neighboring (*b-neighboring*) segment of a segment G_g is referred to as* b-neighbor(G_g)*, where:* b-neighbor$(G_g) = G_b \equiv G_b \mid G_g$.

Definition 2.12 (F-neighbor) *The forward neighboring (*f-neighboring*) segment of a segment G_g is referred to as* f-neighbor(G_g)*, where:* f-neighbor$(G_g) = G_f \equiv G_g \mid G_f$.

Notice that neighboring segments of $\mathcal{H}$ always meet each other.

Furthermore, the two end-points of $\mathcal{H}$ are identified as:

Definition 2.13 (Frontier segment) *Segment G_g is a* frontier segment *if either:*

$$\nexists l \in \mathcal{H} \text{ b-neighbor}(G_g) = l \text{ or } \nexists r \in \mathcal{H} \text{ f-neighbor}(G_g) = r.$$

The first and last segments of $\mathcal{H}$ are called frontier segments since they are the outer fringes of $\mathcal{H}$.

Sometimes there are intervals over which no observations are available:

Definition 2.14 (Gap-fill segment) *For any segment G_g,* SEG-PROPS$(G_g) = \emptyset$ *exactly when G_g is a* gap-fill segment.

To minimize complexity, consecutive gap-fill segments are forbidden. Thus, for gap-fill segment G_g, the expression $G_l \mid G_g \mid G_r$, indicates the backward and forward non-gap-fill neighbors of G_g. The shorthand notation $G_l \parallel G_r$ indicates that either $G_l \mid G_r$ or $G_l \mid G_g \mid G_r$, for some gap-fill segment G_g.

It should be noted that the temporal total-ordering of $\mathcal{H}$ precludes general temporal relations, such as "Property X holds sometime during the time period from t1 to t2 in which property Y also holds", allowed in other temporal representations (Allen, 1983) (Williams, 1986). However, for many interpretation tasks, reasonably accurate time-stamps for the measurements are available, providing temporal total-orderings. Furthermore, the computational overhead associated with reasoning about partial temporal-orderings may often be too high [1].

Values for particular properties may be ambiguous due to noisy data. One could simply ignore such noisy data altogether and just use unambiguous observations. However, that would be overly conservative since even noisy data provides some constraint. For example, suppose the numerical values of a series of measurements indicate that a system variable is decreasing or steady. In that case, one at least knows that it is not increasing. DATMI allows disjunctive property values to represent such cases.

[1] Section 8.3.2 discusses some ideas for dealing with partial temporal-orderings.

2.2 Specifying Observations and Measurements

Observations fall into two categories: numerical *measurements* and symbolic *observations*. Measurements are specified as:

Definition 2.15 (Measurement) `MEASURE(n,r,t,i,c)` *indicates the numerical (real) value r for the numerical property named n was measured at time t, by the measuring instrument i, with probability c* [2].

Alternatively, symbolic observations are specified as:

Definition 2.16 (Observation) `OBSERVE(p,v,t1,t2,s,c)` *asserts that the qualitative property named p has value v over the time period from t_1 to t_2, according to source s, with probability c.*

Some states of the envisionment must mention the qualitative properties indicated by the observations to allow comparisons. The numerical properties indicated by measurements are translated into appropriate qualitative properties, as explained in the next section.

2.3 Converting Measurements into Qualitative Properties

To interpret measurements at instants, they must be translated into qualitative properties holding over periods of time. DATMI's translation methods attempt to provide qualitative properties which is neither uselessly weak nor more certain than the data warrants. These methods require domain-specific knowledge, as described below. They also require that the original analog data has been smoothed, using traditional techniques such as Gaussian convolution or least squares methods.

Domain-specific knowledge first specifies *which* sets of numerical properties map into particular qualitative properties. For example, numerical properties for `Water-Level-1` and `Water-Level-2` are both required to determine the qualitative property `ORDER(Water-Level-Can1, Water-Level-Can2)`. Of course, a qualitative property might result from alternative sets of numerical properties, especially if multiple sensors are used. For example, suppose two sensors yield measures $A1$ and $A2$ respectively for some quantity A and two other sensors yield measures $B1$ and $B2$ respectively for some other quantity B. The qualitative property `ORDER(A,B)` could then be determined by any of the four possible comparisons of $A1$ or $A2$ with $B1$ or $B2$. However, the current DATMI implementation does not handle multiple, conflicting, qualitative property assertions.

DATMI also uses domain-specific *quantity-space conversion tables* which specify *how* these numerical measurements are to be mapped into qualitative properties by accounting for the precision and accuracy of the sensors. For instance, for qualitative property `ORDER(Water-Level-Can1, Water-Level-Can2)` with possible values `GREATER`, `LESS`, or `EQUAL`, there must be numerical cutoffs for the relative ratio of the numerical values of `Water-Level-1` and `Water-Level-2` that determine which is actually greater. Figure 2.2 gives an example conversion table for each of the two general classes of properties that can result from measurements. As this figure illustrates, each conversion table can be weighted with discrete probabilities for each alternative value. Those probabilities must be supplied by external domain-specific means, perhaps based on the accuracies of the sensors. Otherwise, DATMI assumes all values are equally likely.

[2] MEASURE is like ATMI's **Measured** predicate, except that it allows some confidence to be specified as well.

```
          ORDER(Water-Level1,Water-Level2):
If relative ratio of Water-Level1 to Water-Level2 is:
- ∞ to -0.1 ⇒  (LESS with probability 1.0)
-0.1 to 0   ⇒  (LESS with prob 0.8) ∨ (EQUAL with prob 0.2)
   0 to 0   ⇒  (EQUAL with prob 1.0)
   0 to 0.1 ⇒  (GREATER with prob 0.8) ∨ (EQUAL with prob 0.2)
 0.1 to ∞   ⇒  (GREATER with prob 1.0)

          CHANGE(Temperature(Can)):
If slope of change in Temperature(Can) is:
- ∞ to -0.1 ⇒  (DECREASING with probability 1.0)
-0.1 to 0   ⇒  (DECREASING with prob 0.7) ∨ (STEADY with prob 0.3)
   0 to 0   ⇒  (STEADY with prob 1.0)
   0 to 0.1 ⇒  (INCREASING with prob 0.7) ∨ (STEADY with prob 0.3)
 0.1 to ∞   ⇒  (INCREASING with prob 1.0)
```

Figure 2.2: Example quantity-space conversion tables

For these mappings, the *relative ratio* of $x1$ to $x2$ is $\frac{x1-x2}{|x2|}$, or just $x1$ if $x2 = 0$. The *slope of change* in property P is simply $(v_{t_2} - v_{t_1})/(t_2 - t_1)$, where $t1 < t2$ and v_t is the value of P at time t.

Another way that DATMI reduces the effects of noisy data is through the use of *noise windows* defined for each property. Each window indicates how many measurements on each end a of measurement sequence are required to feel confident that the inner measurements are qualitatively accurate. Each sequence consists only of:

Definition 2.17 (Close data) *Data which are temporally "close enough" that hidden qualitative changes between any two data points could not occur are called* close data.

With a noise window of size two, six close measurements for times t_1 to t_6 ($M_1, M_2, M_3, M_4, M_5, M_6$), each mapping into an assertion that property p has value v, would only result in a single property assertion: p is v over the interval t_3 to t_4. Thus, the noise window helps ensure that a property is only asserted when enough closely neighboring measurements support it.

Figures 2.3 and 2.4 illustrate how measurements are translated into property assertions for the two general classes of properties. As Figure 2.4 indicates, ORDER properties can even be determined over time intervals where the numerical properties are not measured at corresponding times, by identifying trends over close data. For example, one can see that the value of property A is greater than that of property B from points x to y because at the later point z the value of A is still greater than that of B. However, also note that the ORDER property for A and B from point y to z cannot be determined since the measurement points do not indicate whether point l is before or after point z.

Even if noisy data are not a concern, a window size of one is still required to address the observation correspondence problem. This problem can arise whenever not all properties have measurements at the same times. Figure 2.3 shows an example of this problem. In this example,

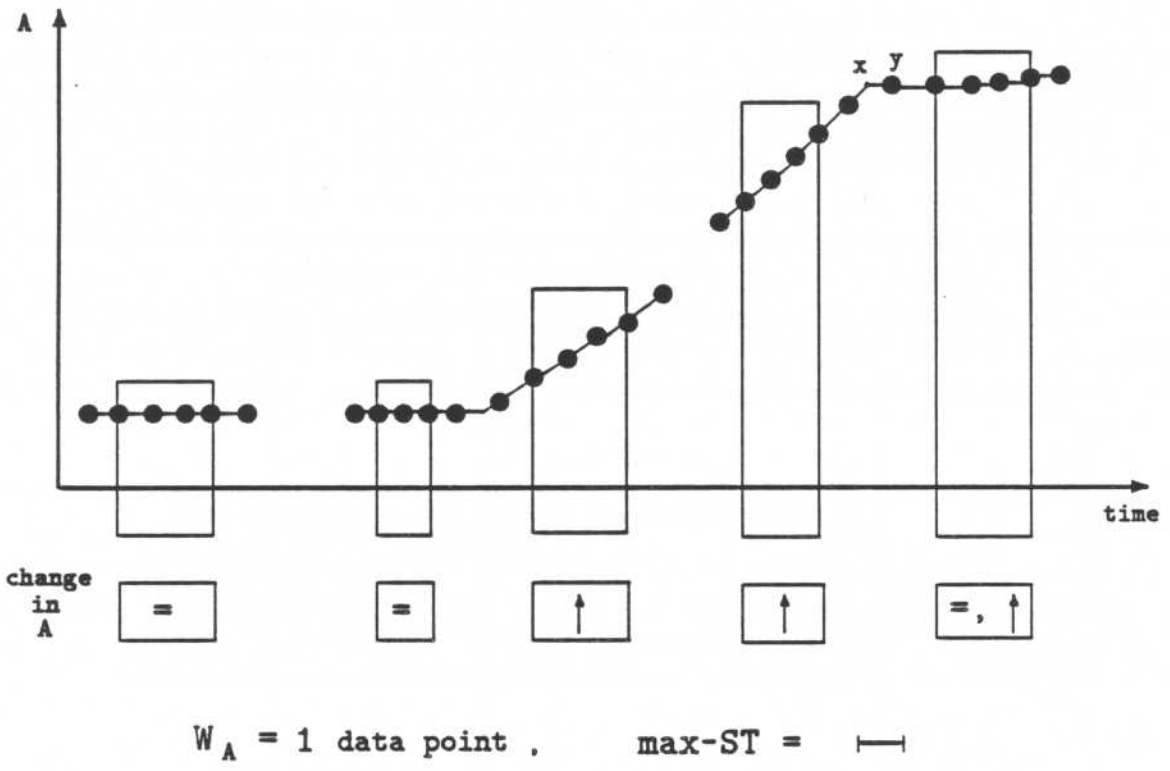


Figure 2.3: Translating measurements to CHANGE properties

change at point x occurs before the point y at which the change is first observed. If the window size was zero, then the preceding increase in property A would be asserted up to point y. The property assertion from point x to point y would then be incorrect. During global segmentation, this could cause correspondence errors with other property assertions between points x and y.

Which data are close is determined by using domain-specific sample-time boundaries for each qualitative property ρ, called MIN-ST(ρ) and MAX-ST(ρ). Data points temporally closer than MIN-ST(ρ) are completely believed to be close. Data points farther apart than MAX-ST(ρ) are not even considered as part of the same qualitative property interval since they aren't nearly close enough to warrant any assumption of continuity [3].

2.4 Creating, Merging, and Splitting Global Segments

If a new property assertion is over the same interval as an existing segment, then the property is just added to that segment's set of properties. However, assertions can also cause the history

[3]Data points whose distances are between MIN-ST(ρ) and MAX-ST(ρ) could have some confidence level of being close which is proportional to the ratio of that distance and the difference between the MAX-ST(ρ) and MIN-ST(ρ). These confidences could be combined with the probabilities of property values given by the quantity-space conversion tables. However, DATMI currently provides no approach for doing this.

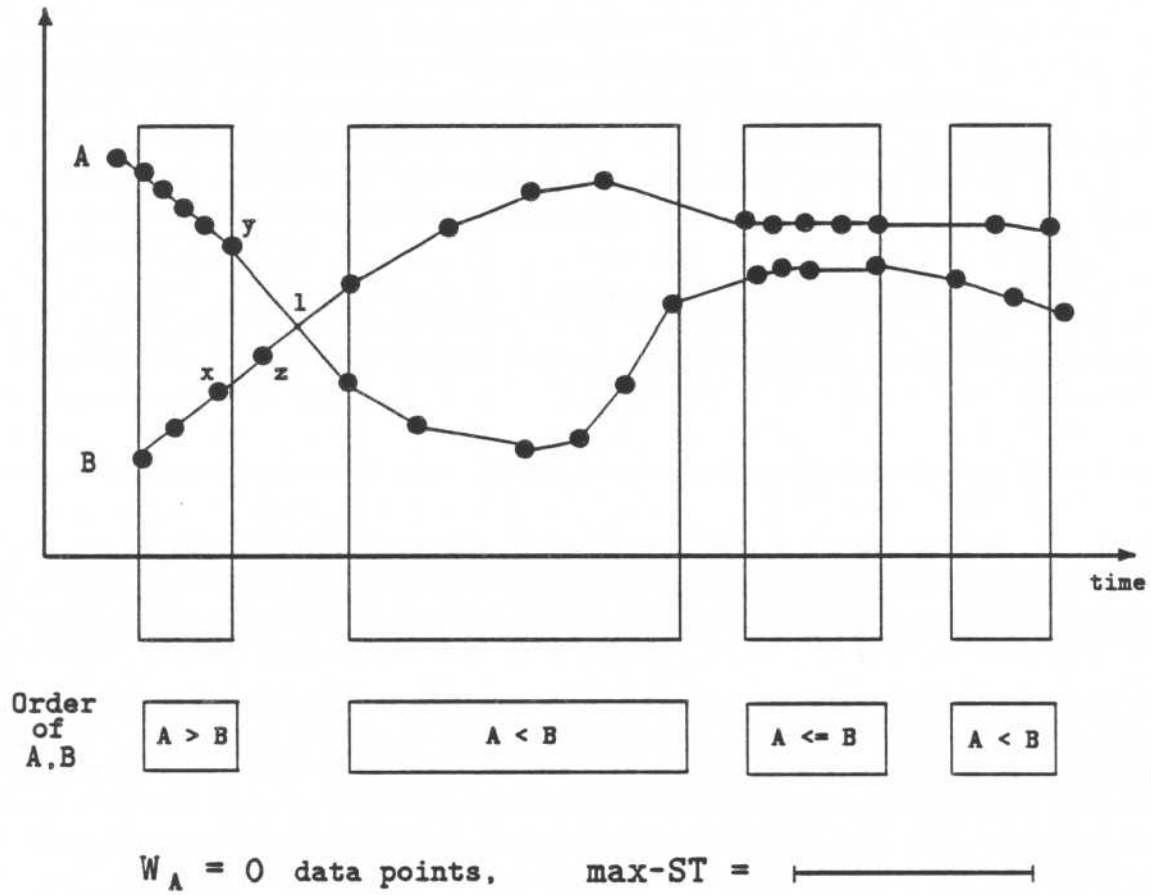


Figure 2.4: Translating measurements to ORDER properties

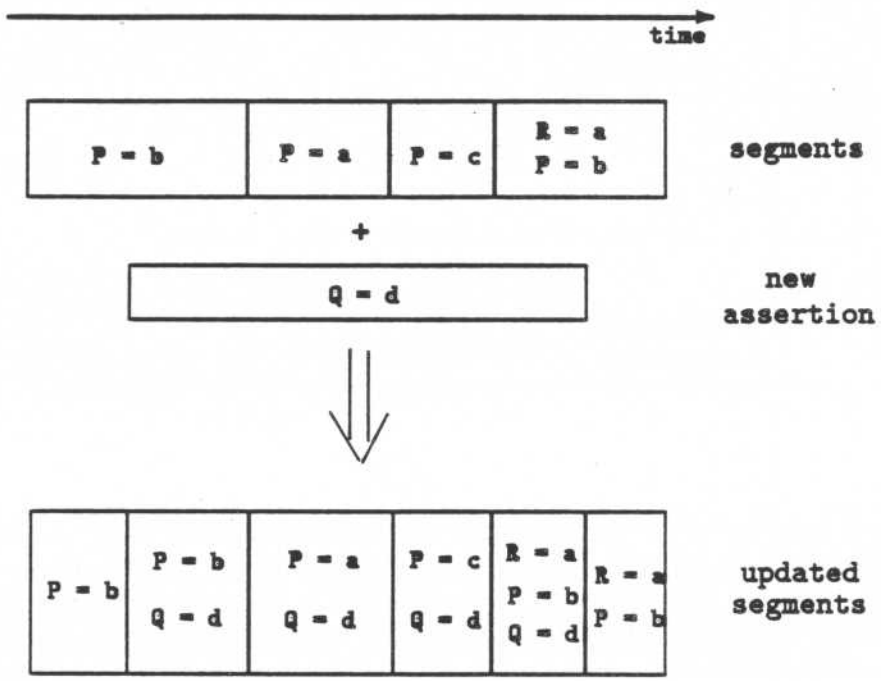


Figure 2.5: Splitting segments when asserting a property

to be reorganized. This reorganization can invoke three kinds of operations: creation, splitting, and merging.

If the assertion interval occurs after (or before) all existing segments, then a new segment G_n must be created with SEG-PROPS(G_n) containing just the new property. If this new interval is outside of the current history (past a frontier segment) then a gap-fill segment is also created to represent the interval between the old frontier segment and the newly created one.

A new qualitative property assertion may temporally overlap a portion of an existing segment. Such overlaps require splitting the overlapped segment into two smaller segments: one containing the new property and one that does not. Furthermore, assertions which cover several segments must have the property added to each of the covered segments, with any partially overlapped segments on either end of the assertion being split appropriately. An example of segment splitting is presented in Figure 2.5.

Neighboring segments which represent the same properties must be merged to keep the observational history concise. Figure 2.6 illustrates the effects of merging segments. Such merging involves temporally extending the earlier of two neighboring segments to cover the time interval of both segments and then discarding the other segment.

Actually, merging should not always occur when two neighboring segments have identical properties. The probabilities of the properties, which are determined when translating measurements into qualitative properties, should also be comparable – not just the property values themselves. Merging two segments which differ greatly in their probabilities for a particular property would lose information essential to later backtracking to handle faulty data. This

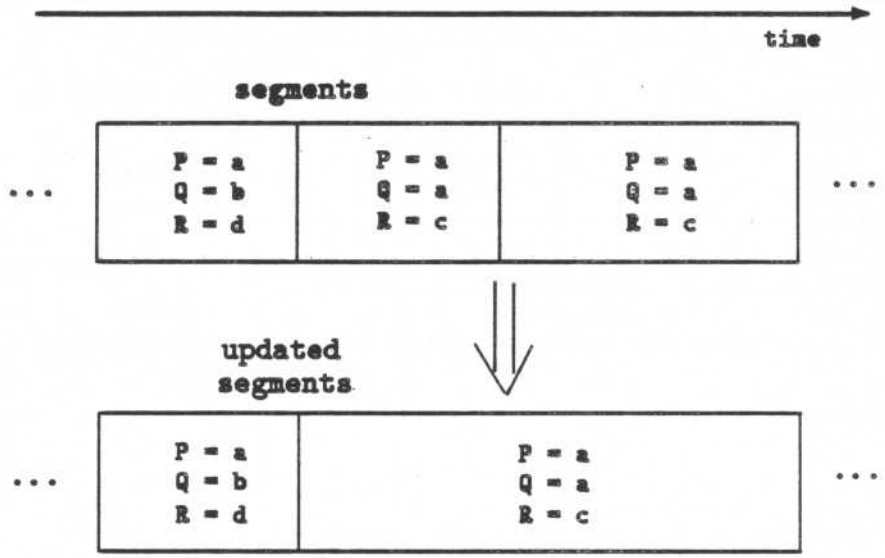


Figure 2.6: Merging segments to maintain a concise history

suggests partitioning a concise history episode into segments with qualitatively distinct probabilities for such properties. This partitioning would require domain-specific criteria that indicate when two probabilities for a property were significantly different. Although DATMI currently does not perform such partitioning, it does ensure that the probability of a merged property is the minimum of the levels in the two merging segments, to avoid overestimating the confidence in the observations.

Chapter 3

MAINTAINING THE PINTERP-SPACE

The interpretation space maintained by DATMI indicates all of the envisionment states that could actually occur during each of the global segments. A particular state that can occur in a particular segment is called a *pinterp* (Forbus, 1986b) because it indicates a "possible interpretation" for that segment. DATMI refers to these pinterps as follows:

Definition 3.1 (Pinterp) *$P(G_g, S_s)$ denotes the* pinterp *which indicates that state S_s can occur some time during segment G_g.*

The interpretation space consisting of these pinterps is referred to as the *pinterp-space.*

For simplicity, this paper will sometimes just refer to a pinterp to indicate its state or segment, if this is unambiguous. Usually, such references indicate a state. For example, a path of states is said to pass through a pinterp if it passes through that pinterp's corresponding state. However, two cases refer to the segment instead:

Definition 3.2 (Neighboring pinterps) *Pinterps $P(G_g, S_a)$ and $P(G_n, S_b)$ are called* neighboring pinterps *exactly when $G_g \mid G_n$ or $G_n \mid G_g$ (ie. they correspond to neighboring segments).*

Definition 3.3 (Same-segment pinterps) *Two pinterps $P(G_g, S_a)$ and $P(G_n, S_b)$ are called* same-segment pinterps *exactly when $G_g = G_n$ (ie. they both correspond to the same segment).*

Maintaining the set of all states that could occur in a segment avoids the unnecessary expense of instead maintaining the set of all state paths spanning that segment that consist only of these states. Indeed, many typical interpretation tasks, such as monitoring, are concerned only with whether a system could be in some particular states. Even tasks requiring global interpretations, such as explanation, typically only need the best current global interpretation. Storing all such paths is typically also impractical. In fact, as noted in (Forbus, 1986a), there may even be an infinite number of such paths if there are cycles in the envisionment.

Nevertheless, DATMI does explicitly represent certain significant paths of states across segments, including those which together form the best current global interpretation. *Pinterp dependencies* are associated with each pinterp $P(G_g, S_s)$ to indicate a possible path of states across G_g which passes through $P(G_g, S_s)$ while connecting a pinterp in `b-neighbor`(G_g) with a pinterp in `f-neighbor`(G_g). So, the pinterp dependencies for a pinterp $P(G_g, S_s)$ indicate one possible behavior for G_g that has S_s occurring in G_g. These dependencies serve two functions:

1. **They explicitly provide, at all times, the current best global interpretation ending at each pinterp.**

2. **They indicate which pinterps must be re-supported when a particular pinterp becomes inconsistent.**

Supporting pinterps with these dependencies is analogous to supporting nodes with justifications in truth-maintenance systems (Doyle, 1979).

This chapter describes how DATMI incrementally maintains pinterp dependencies as new observations are obtained.

3.1 Consistency Constraints and Classes of Pinterps

DATMI uses symbolic relaxation (Waltz, 1972; Mackworth & Freuder, 1985) to construct globally consistent interpretations from local constraints. In such approaches, the constraint network is a graph of nodes, each representing specific variable assignments. These nodes are connected by arcs reflecting local constraints on the values of variables. A complete solution is a connected subgraph with one node for each variable which indicates a set of variable assignments satisfying the constraints.

Each DATMI global segment plays the role of a variable and the states provide the values. The pinterp $P(G_g, S_s)$ represents the variable assignment of a state S_s to a segment G_g. Each variable assignment (pinterp) must satisfy two types of local consistency constraints:

Definition 3.4 (Property contraints) *A pinterp $P(G_g, S_s)$ satisfies the* property constraints *exactly when the properties of state S_s are compatible with the properties of segment G_g. A pinterp which satisfies the property constraints is called* property consistent.

Definition 3.5 (Transition constraints) *A pinterp $P(G_g, S_s)$ satisfies the* transition constraints *exactly when being in S_s during G_g is consistent with being in both: some state during* b-neighbor(G_g) *and some state during* f-neighbor(G_g). *A pinterp which satisfies the transition constraints is called* transition consistent. *The envisionment's state transitions and the set of pinterps satisfying all consistency constraints together define the transition constraints.*

The status of a pinterp $P(G_g, S_s)$ indicates whether the variable assignment of S_s to G_g is consistent, as follows:

1. INCOMPATIBLE – S_s cannot occur in G_g because it is not property consistent [1].

2. INACTIVE – S_s cannot occur in G_g because it is not transition consistent.

3. ACTIVE – S_s can occur in G_g.

4. UNKNOWN – consistency has not been checked.

Definition 3.6 (Filtering and activating pinterps) *The process of changing the status of a pinterp from* ACTIVE *to* INACTIVE *is referred to as* filtering *that pinterp (from the set of* ACTIVE *pinterps). Similarly,* activating *a pinterp means changing its status to* ACTIVE.

[1] The DATMI implementation actually discards all INCOMPATIBLE pinterps to save space. This requires re-creating some pinterps if some segment properties are later discarded as possibly faulty data.

Pinterps which are either ACTIVE or INACTIVE are also called COMPATIBLE, since they satisfy the property constraints. All pinterps satisfying the property constraints are initially assumed ACTIVE and are then later filtered if they violate the transition constraints.

Only ACTIVE pinterps participate in pinterp dependencies. Whenever a pinterp becomes INACTIVE, all the pinterps depending on it must re-satisfy the transition constraints, which gives them new dependencies. Those pinterps which cannot re-satisfy the transition constraints become INACTIVE themselves, propagating the effects of the original filtering.

UNKNOWN pinterps refer to states which are not currently being considered. It may be desirable to ignore some pinterps when the envisionment is very large, observations are very incomplete, or those pinterps are very implausible. The current DATMI implementation does not provide means for labeling pinterps as UNKNOWN.

3.2 Types of Transition Consistency Relations

This section discusses the ways that the transition constraints can be satisfied. The five types of transition consistency relations between a pinterp and a neighboring segment are referred to as: *frontier-state*, *spanning-state*, *meeting-states*, *hidden-transition*, and *gap-filling*. Figure 3.1 illustrates each type. Each transition consistency relation indicates a path of ACTIVE pinterps starting at the given pinterp $P(G_g, S_s)$ and reaching a pinterp in b-neighbor(G_g) or f-neighbor(G_g). A pinterp $P(G_g, S_s)$ satisfies the transition constraints exactly when $P(G_g, S_s)$ has at least one transition consistency relation with f-neighborg and at least one with b-neighborg. If a neighboring segment is a gap-fill segment, then the b-neighbor or f-neighbor, respectively, of that gap-fill segment is instead used for these relations.

Frontier-state consistency refers to the special case where the pinterp's segment is a frontier segment and the neighboring segment does not exist.

Spanning-state consistency for a pinterp $P(G_g, S_s)$ occurs when the pinterp for S_s in the neighboring segment is ACTIVE. S_s can then persist ("span") from G_g to the neighboring segment. This can occur only when the properties of one segment *p-subsumes* the properties of a neighboring segment, according to the following definition:

Definition 3.7 (P-subsumption) *For any disjunctive property values* $v1$ *and* $v2$, $v1$ *is* more general *than* $v2$ *exactly when* PROP-VAL$(P,p) = v2 \Rightarrow$ PROP-VAL$(P,p) = v1$. *Property set* P p-subsumes *property set* Q *exactly when, for all* $p \in P$, PROP-VAL$(P,p) \neq \emptyset \Rightarrow$ PROP-VAL(Q,p) *is more general than* PROP-VAL(P,p).

DATMI prefers spanning-state relations because they indicate paths which best meet the *simplest action assumption* (Forbus, 1984).

Meeting-states consistency occurs when there is a state transition from the given pinterp to an ACTIVE pinterp in the neighboring segment.

Hidden-transition consistency for a pinterp $P(G_g, S_s)$ occurs when there is a path of ACTIVE pinterps of G_i connecting $P(G_g, S_s)$ with an ACTIVE pinterp of the neighboring segment.

Gap-filling consistency for a pinterp $P(G_g, S_s)$ occurs when there is a path, of any envisionment states, which connect $P(G_g, S_s)$ with an ACTIVE pinterp in the neighboring segment.

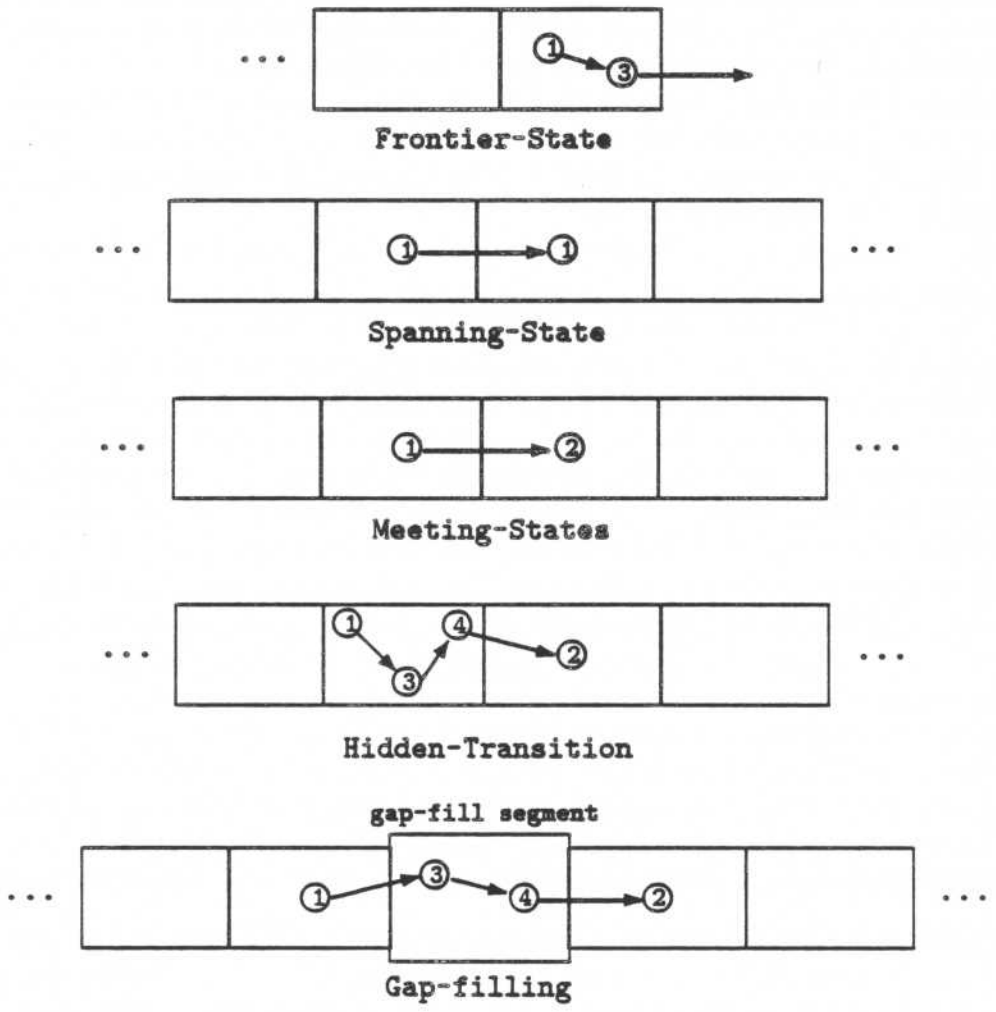


Figure 3.1: Examples of the five DATMI transition consistency relations

An example path from state 1 to a state of the f-neighboring segment, if any, is given for each case. Each box represents a segment and each circle represents an ACTIVE pinterp in that segment corresponding to the numbered state. Each arrow corresponds to a state transition.

3.3 Representing Transition Consistency Relations With Dependency Paths

For each ACTIVE pinterp and a neighboring segment, DATMI caches only the path indicated by the best transition consistency relation between that pinterp and segment. These two paths for each pinterp provide its pinterp dependencies as follows:

Definition 3.8 (Dependency paths) *A* dependency path *for some pinterp $P(G_g, S_s)$ represents the path of pinterps indicated by the best transition consistency relation between $P(G_g, S_s)$ and a neighboring segment. The* f-dependency path *of $P(G_g, S_s)$ connects $P(G_g, S_s)$ with a pinterp in* f-neighbor(G_g) *and the* b-dependency path *of $P(G_g, S_s)$ connects $P(G_g, S_s)$ with one in* b-neighbor(G_g).

Since DATMI only considers acyclic interpretations over each segment, each dependency path consists of no more than N states, where N is the number of envisionment states.

3.3.1 Caching Exactly Two Dependency Paths Per Pinterp Is Best

DATMI caches the path of the best transition consistency relation between each pinterp and neighboring segment because it can directly contribute to the best working global interpretation. Furthermore, by caching these paths as the pinterp dependencies, the pinterps need not be re-satisfied with the transition constraints unless one of the pinterps in those paths becomes INACTIVE.

Caching alternative paths as well, each representing less-optimal transition consistency relations, would not lead to the efficiencies that one might expect. In order to provide best global interpretations, a dependency path that becomes inconsistent with the transition constraints must be replaced with the best consistent path. Selecting which cached alternative to use as the new best path will typically cost as much as full search for a new dependency path. This is because such selection involves search itself, to verify that this alternative is now the best.

Even if the best global interpretation was not required, caching alternatives is still undesirable. Since the search algorithms discussed in Section 3.7 are so efficient, the overhead in always keeping track of which cached alternatives are consistent is typically unnecessary. Also, one cannot expect a manageable-sized set of cached alternatives to even contain a consistent one, since the number of them can be exponential in the size of the envisionment (see Section 7.2). So, even postponing consistency verification until needed is typically not worth the overhead.

Caching alternative paths could only be worth it if they involve many states and one could determine that they were much more likely to be consistent with the expected observations than most other acylic paths through the envisionment. However, such a determination would require search over the exponential number of paths through the envisionment.

3.3.2 Some Example Dependency Paths

To simplify discussions, the following dependency relations among pinterps are defined:

Definition 3.9 (Pinterp dependency relations) *A pinterp $P(G_g, S_s)$* depends *exactly on those pinterps in its two dependency paths. It* f-depends *only on the pinterps in its f-dependency path (except $P(G_g, S_s)$ itself) and* b-depends *only on the pinterps in its b-dependency path (except $P(G_g, S_s)$).*

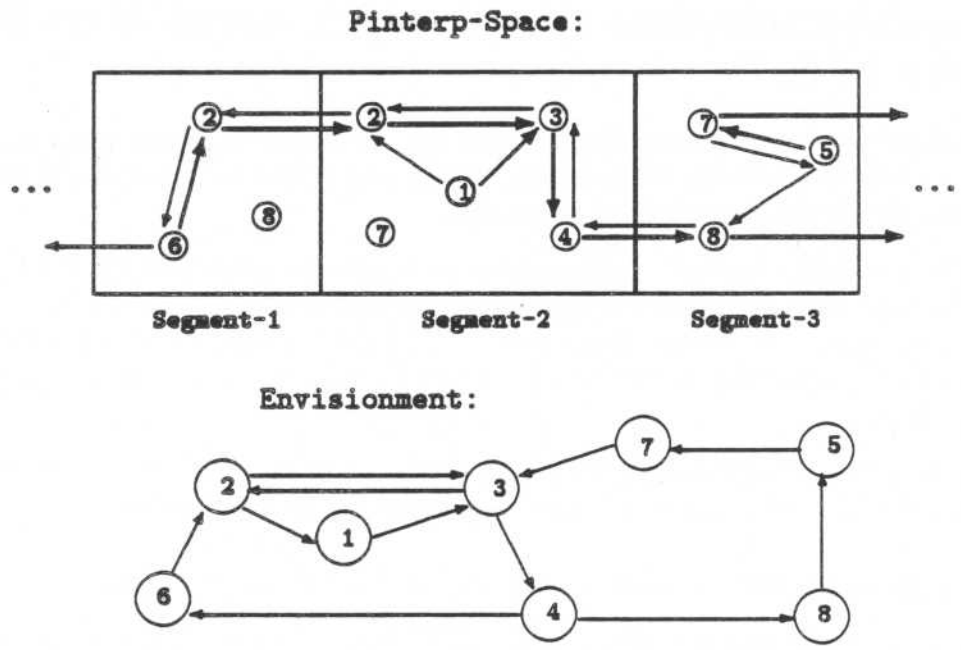


Figure 3.2: Example of pinterp dependencies

For pinterp-space: Boxes represent segments, circles represent COMPATIBLE pinterps corresponding to the numbered states, and thick arrows indicate state transitions in f-dependency paths. Note that the (thin) b-dependency arrows point in the reverse directions of the state transitions, to show the reversed direction of dependency.
For envisionment: Circles indicate states and arrows indicate transitions.

Figure 3.2 shows example dependency paths in part of a pinterp-space. Each path is indicated by a sequence of arrows leading from one pinterp to a pinterp of a neighboring segment. Although pinterps $P(G_1, S_8)$ and $P(G_2, S_7)$ are both COMPATIBLE, they are also INACTIVE because they violate the transition constraints. Furthermore, the ACTIVE pinterp $P(G_2, S_1)$ b-depends on same-segment pinterp $P(G_2, S_2)$ and neighboring pinterp $P(G_1, S_2)$ and f-depends on $P(G_2, S_3)$, $P(G_2, S_4)$, and $P(G_3, S_8)$.

Note that, while each ACTIVE pinterp (except some frontier pinterps) must f-depend and b-depend on some pinterps, some pinterps may not have any pinterps f-depend or b-depend on them. For example, no pinterp in Figure 3.2 depends on $P(G_2, S_1)$ since the transition $P(G_2, S_2) \rightarrow P(G_2, S_3)$ provides a shorter path from $P(G_2, S_2)$ to $P(G_2, S_3)$ than $P(G_2, S_2) \rightarrow P(G_2, S_1) \rightarrow P(G_2, S_3)$.

Also, observe that an interpretation from Segment-1 to Segment-3 can be found simply by following either the chain of f-dependency paths from some ACTIVE pinterp in Segment-1 or the chain of b-dependency paths from some ACTIVE pinterp in Segment-3. For instance, following the chain of f-dependency paths from pinterp $P(G_1, S_6)$ in Figure 3.2 yields a working interpretation $P(G_1, S_6) \rightarrow P(G_1, S_2) \rightarrow P(G_2, S_2) \rightarrow P(G_2, S_3) \rightarrow P(G_2, S_4) \rightarrow P(G_3, S_8)$.

In this case, following the chain of b-dependency paths from $P(G_3, S_8)$ would lead to the same interpretation. However, the backward and forward dependency paths need not be symmetric; in fact, Section 3.5 presents an algorithm for propagating costs of states and transitions which may result in asymmetric dependency paths.

3.4 Maintaining Pinterp Dependencies

The pinterp-space must be updated whenever a new segment is created or properties of existing segments are changed. The basic process of updating the pinterp-space involves two steps:

1. Determine which pinterps become INCOMPATIBLE due to the new segment properties.
2. Determine which pinterps must now be filtered because they no longer satisfy the transition constraints.

Figure 3.3 shows the primitive functions used in pinterp-space maintenance.

DATMI uses essentially the same technique as ATMI to find COMPATIBLE-STATES(ρ), the set of states compatible with a set of properties ρ. It uses the following lookup-table representation:

Definition 3.10 (State lookup-table) *A* state lookup-table *stores for each property name the set of states which are compatible with each possible property value. Given a set of properties ρ, it returns* COMPATIBLE-STATES(ρ).

This table is easily generated from the envisionment off-line. A segment's COMPATIBLE pinterps can be determined by intersecting the set of states compatible with each of the segment's properties. Since DATMI property values can be disjunctive, the set of states compatible with a particular property is the union of the states compatible with each property value disjunct.

3.4.1 Processing a New Segment

A new segment G_g is either a new frontier segment or part of what was a gap-fill segment. G_g is first assigned a set of pinterps compatible with its properties. Each of these COMPATIBLE pinterps starts as ACTIVE and is filtered if transition consistency relations cannot be found between it and its neighboring segments. These filterings are then propagated to the rest of the pinterp-space. Figure 3.4 gives the algorithm for processing a new segment.

As shown in step 5 of this algorithm, care is taken to ensure that any pinterp of G_g depending on a newly filtered pinterp of G_g (due to a hidden-transition dependency path) is also filtered if no alternative dependency path can be found for it. This step continues until no more pinterps of G_g are filtered. Step 6 filters any ACTIVE pinterps of the neighboring segments that have no transition consistency relation with this new segment G_g. Finally, in step 6d, the effects of any such filterings are propagated to the other segments.

3.4.2 Processing an Updated Segment

Whenever additional properties are asserted for a segment, some previously ACTIVE or INACTIVE pinterps can become INCOMPATIBLE. In this case, the ACTIVE pinterps which have become INCOMPATIBLE are treated as newly filtered pinterps of G_g for the sake of propagating the effects of these incompatibilities. However, their statuses are not actually made INACTIVE. Figure 3.5 shows how this is accomplished.

These definitions hold for pinterps P and $P(G_g, S_s)$, global segment G_g, set of properties ρ, and temporal direction d:

SEG-PROPS(G_g) ≡ set of properties asserted for segment G_g.

COMPATIBLE-STATES(ρ) ≡ {$\forall S_i$:
S_i is a state whose properties are compatible with ρ}.

COMPATIBLE?(S_s, ρ) ⟺ $S_s \in$ COMPATIBLE-STATES(ρ).

COMPATIBLE-PINTERPS(ρ, G_g) ≡ { $\forall P(G_g, S_i)$: COMPATIBLE?(S_i, ρ) }

STATUS(P) ∈ {ACTIVE, INACTIVE, INCOMPATIBLE, UNKNOWN}

ACTIVES(G_g) ≡ {$\forall P(G_g, S_i)$: STATUS($P(G_g, S_i)$)=ACTIVE}

INACTIVES(G_g) ≡ {$\forall P(G_g, S_i)$: STATUS($P(G_g, S_i)$)=INACTIVE}

ACTIVE?(P) ⟺ STATUS(P)=ACTIVE

INACTIVE?(P) ⟺ STATUS(P)=INACTIVE

REV-DIR(d) = BACKWARD if d = FORWARD, else FORWARD if d = BACKWARD.

DEPENDENCIES(P,d) = set of pinterps in P's b-dependency path (except P) if d=BACKWARD
set of pinterps in P's f-dependency path (except P) if d=FORWARD

DEPENDING-ON(P,d) = {P_i: $P \in$ DEPENDENCIES(P_i, REV-DIR(d))}

NEIGHBOR-SEGMENT(G_g, d) = G_n where:
If d = BACKWARD then $G_n \mid G_g$.
If d = FORWARD then $G_g \mid G_n$.

NON-GAP-FILL-NEIGHBOR(G_g, d) = G_n where:
If d = BACKWARD then $G_n \parallel G_g$.
If d = FORWARD then $G_g \parallel G_n$.

FRONTIER-SEGMENT?(G_g, d) ⟺ (d=BACKWARD ∧ $\emptyset \mid G_g$) ∨ (d=FORWARD ∧ $G_g \mid \emptyset$).

Figure 3.3: Primitive DATMI definitions

Given: new segment G_g with properties SEG-PROPS(G_g).

(Procedure FIND-DEPENDENCY-PATH is defined in Section 3.7. If no dependency path can be found for a pinterp, it returns fail and filters that pinterp. Otherwise it records the dependency path.)

```
1. C ← COMPATIBLE-STATES(SEG-PROPS(G_g)).
2. For S_s ∈ C do
   a. Create pinterp P(G_g, S_s).
   b. STATUS(P(G_g, S_s)) ← ACTIVE.
3. I ← ∅.          ; the set of newly filtered pinterps
4. For S_s ∈ C do
     For d ∈ {BACKWARD, FORWARD} do
       a. Call FIND-DEPENDENCY-PATH(P(G_g, S_s), d).
       b. If fail, I ← I ∪ P(G_g, S_s).
5. For P ∈ I do
     For d ∈ {BACKWARD, FORWARD} do
       For P(G_k, S_s) ∈ DEPENDING-ON(P, d) do
         ; find alternative support for dependents of newly filtered ones:
         a. Call FIND-DEPENDENCY-PATH(P(G_k, S_s), d).
         b. If fail then I ← I ∪ P(G_k, S_s) else I ← I − P(G_k, S_s).
6. For d ∈ {BACKWARD, FORWARD} do
   a. G_n ← NON-GAP-FILL-NEIGHBOR(G, d).
   b. I ← ∅.
   c. For P(G_n, S_s) ∈ ACTIVES(G_n) do
      i. Call FIND-DEPENDENCY-PATH(P(G_n, S_s), d).
     ii. If fail, I ← I ∪ P(G_n, S_s).
   d. Call PROPAGATE-FILTERING-EFFECTS(G_n, I, d).
```

Figure 3.4: Procedure CREATE-PINTERPS

Given: segment G_g and set of new properties $\rho \in$ SEG-PROPS(G_g).

```
1. I ← ∅.          ; set of newly incompatible pinterps
2. For P(G_g, S_s) ∈ ACTIVES(G_g) do
     If not COMPATIBLE?(S_s, ρ)
       then I ← I ∪ P(G_g, S_s).
3. For d ∈ {BACKWARD, FORWARD} do
     Call PROPAGATE-FILTERING-EFFECTS(G_g, I, d).
```

Figure 3.5: Procedure REFINE-PINTERPS

Given: seed-segment G_g, set of newly filtered pinterps I, and direction d.

```
1. I_n ← ∅        ; set of newly filtered pinterps of neighboring segment
2. For P ∈ I do
       For P(G_k, S_s) ∈ DEPENDING-ON(P, d) do
          a. Call FIND-DEPENDENCY-PATH(P(G_k, S_s), REV-DIR(d)).
          b. If fail then
                If G_k = G_g
                   then I ← I ∪ {P(G_k, S_s)}    ; more for current segment
                   else I_n ← I ∪ {P(G_k, S_s)}.
3. Unless I_n = ∅
     a. G_n ← NON-GAP-FILL-NEIGHBOR(G_g, d).
     b. Call PROPAGATE-FILTERING-EFFECTS(G_n, I_n, d).
```

Figure 3.6: Procedure PROPAGATE-FILTERING-EFFECTS

3.4.3 Propagating the Effects of Pinterp Filterings

A central operation in the incremental maintenance of the pinterp-space is the propagation of pinterp filterings. The key to efficiently handling this propagation is to realize that only the pinterps that depend on one of the newly filtered pinterps can possibly be affected. Figure 3.6 presents this propagation algorithm.

Note that propagation consists of two sweeps, one backwards and one forwards, over the segments. During each sweep, all pinterps of the current segment G_g which have no transition consistency relations with the neighboring segment G_n are filtered before proceeding to G_n. Once propagation reaches G_n, the set of pinterps of G_g that have transition consistency relations with G_n cannot change. Thus, propagation need never examine segments more than once – uni-directional sweeps are sufficient.

During the propagation of pinterp filterings, faulty observations or models may manifest themselves as follows:

Definition 3.11 (Inconsistent segment) *A segment which has no* ACTIVE *pinterps is said to be an* inconsistent segment.

An inconsistent segment indicates that no global interpretation is possible. To fix this problem, either the observations or the model must be modified. Chapter 1 discusses how DATMI performs such fixes.

3.5 Path-Cost Maintenance

The methods discussed so far ensure that the pinterp-space satisfies the property constraints and transition constraints. However, they do not ensure that the global interpretation indicated by the dependency paths is the best one.

DATMI solves this problem by associating with each pinterp a measure of the desirability of the global interpretation it indicates, as follows:

Definition 3.12 (Path-cost) *Suppose the b-dependency path of pinterp $P(G_g, S_s)$ starts in the b-neighboring pinterp $P(G_n, S_b)$. Then, the* path-cost *of $P(G_g, S_s)$ is the sum of the cost of its b-dependency path and the path-cost associated with $P(G_n, S_b)$. The path-cost of a pinterp in the first segment of the history is just the cost of its state.*

A pinterps's path-cost represents the cost of the entire chain of b-dependency paths supporting that pinterp, beginning with a pinterp in the first segment of the history. The cost of a b-dependency path is usually the sum of the costs of all its states and transitions, excluding the cost of the state of the b-neighboring pinterp in that path. The exception is that spanning-state b-dependency paths have no cost, since the state does not change over that path.

The state costs and transition costs should be conditioned on domain-specific preference criteria. For example, one may use these state costs to indicate how much more preferable states having value X for a property at to states having value Y. These costs could also be conditioned on the *a priori* probabilities that these states or transitions might occur. Path-costs based on such heuristic information would reflect reasons for favoring one global interpretation over another.

Maintaining path-costs involves propagating the path-costs of a pinterp forward through the b-dependency paths. Forward propagation is preferred over backward propagation simply because observations typically come in temporal order, so most changes in the pinterp-space will occur in the later segments. In procedure REFINE-PINTERPS (3.5), the backward filtering sweep must be performed before the the forward filtering sweep to ensure that path-costs are accurate when they are propagated in the forward sweep.

Propagating path-costs requires some changes to the PROPAGATE-FILTERING-EFFECTS procedure (Figure 3.6), as shown in Figure 3.7. After finding a replacement b-dependency path for pinterp $P(G_k, S_s)$ in step 2, it determines if $P(G_k, S_s)$'s path-cost is now greater than when using the replaced dependency path. Notice that a pinterp's path-cost never becomes lower here because better b-dependency paths can never result from having filtering some b-neighboring pinterps [2]. Once step 3 determines that no more filtering is necessary, it invokes the forward propagation of those path-costs increases.

Path-cost propagation proceeds segment by segment, from the earliest affected segment to the latest affected segment, as shown in the algorithms of Figures 3.8 and 3.9. Note that step 4 of PROPAGATE-INCREASES-TO-DEPENDENTS ignores pinterps of G_g which b-depend on the current pinterp $P(G_g, S_s)$ with increased path-cost. While PROPAGATE-INCREASES-TO-DEPENDENTS was propagating the effect of a path-cost increase from b-neighbor(G_g) to $P(G_g, S_s)$, it would also propagate the path-cost increases to all other affected pinterps of G_g.

Unfortunately, it is not always adequate to define global path optimality in terms of the sum of the costs of dependency paths. Consider two alternative global interpretations of the same cost: one which has a hidden-transition and one that does not. The latter might be more preferable because it is simpler, by containing less states. In that case, one could get DATMI to prefer that simpler interpretation by slightly penalizing the costs of all hidden-transition

[2] However, *reactivating* some b-neighboring pinterps may result in better b-dependency paths. Since pinterp reactivations occur only due to changes in beliefs in the observations or model, the propagation of path-costs during reactivation is ignored until Chapter 1.

Given: seed-segment G_g, set of filtered pinterps I, and direction d.

```
1. I_n ← ∅, Q ← ∅
2. For P ∈ I do
     For P(G_k, S_s) ∈ DEPENDING-ON(P, d) do
        a. Call FIND-DEPENDENCY-PATH(P(G_k, S_s), REV-DIR(d)).
        b. If fail
             then
               If G_k = G_g
                  then I ← I ∪ {P(G_k, S_s)}    ; more for current segment
                  else I_n ← I ∪ {P(G_k, S_s)}.
             else
               If d = FORWARD and path-cost of P(G_k, S_s) has increased
                  then ENQUEUE P(G_k, S_s) on Q.
3. If I_n = ∅
     then Call PROPAGATE-PATH-COST-INCREASES(Q).
     else
        a. G_n ← NON-GAP-FILL-NEIGHBOR(G_g, d).
        b. Call PROPAGATE-FILTERING-EFFECTS(G_n, I_n, d).
```

Figure 3.7: Procedure PROPAGATE-FILTERING-EFFECTS (using path-costs)

Given: stack S of pinterps having increased path-costs, ordered with pinterps of earlier segments nearer to the head.

```
1. Q ← ∅    ; queue of pinterps affected
2. P(G_s, S_i) ← STACK-HEAD(S).
3. P(G_q, S_j) ← QUEUE-HEAD(Q).
4. If G_q ⇝ G_s
     ; stop propagation at the segment which is after the other:
     then Call PROPAGATE-INCREASES-TO-DEPENDENTS(G_s, Q, Q)
     else Call PROPAGATE-INCREASES-TO-DEPENDENTS(G_q, Q, S).
  ; note that G_s is never the same segment as G_q
5. Unless both S and Q are empty, Goto step 2.
```

Figure 3.8: Procedure PROPAGATE-PATH-COST-INCREASES

Given: stop-segment G_e, queue Q of affected pinterps, and data structure D which can be either a stack or a queue.
Side-effects: changes contents of Q.

```
 1. P(G_g, S_s) ← HEAD(D).        ; a pinterp
with increased path-cost
 2. If G_e ⇝ G_g then exit    ; since passed stop-segment
 3. POP or DEQUEUE head of D.
 4. For P(G_k, S_i) ∈ DEPENDING-ON(P(G_g, S_s), BACKWARD) do
      When G_k ≠ G_g
        a. FIND-DEPENDENCY-PATH(P(G_k, S_i), BACKWARD).
           ; always succeeds, otherwise filter propagation would have
           ;  filtered P(G_k, S_i) already
        b. When the new path-cost is higher than the old one
           i. If D is stack and Q = ∅ then G_e ← G_g.
              ; ensures that cycle from steps 1 to 5 exits at step 2 as soon
              ;  as all depending pinterps are processed at step 4
          ii. Enqueue P(G_k, S_i) onto queue Q.
 5. Unless D = ∅, Goto step 1.
```

Figure 3.9: Procedure PROPAGATE-INCREASES-TO-DEPENDENTS

dependency paths. However, problems arise when there are global preference factors that cannot be handled by penalizing a dependency path, independent of which global interpretation contains it. For example, such global preferences might include partial-ordering constraints on interpretations.

A related (and perhaps more serious) problem is that there could be other global interpretations with same cost as the best ones given by the b-dependency paths. This is due to the fact that only a single best b-dependency path is recorded for each pinterp. Finding all alternative best global interpretations requires full search over all the ACTIVE pinterps.

3.6 Global Interpretation Construction

The construction of consistent global interpretations from the pinterp-space can be performed in a number of ways, depending on which of the following one wants to know:

1. What is the best global interpretation?
2. What are the K best global interpretations?
3. Can this particular path of states explain the data?
4. Can this particular state occur during this particular period of time?

To determine the best current global interpretation from times t_1 to t_2, DATMI simply follows the chain of b-dependency paths from the segment occurring at t_2 backwards to the segment occurring during t_1. The segment for time t_2 may have several ACTIVE pinterps, so the one with the lowest path-cost is used at the start of this chain.

Search over all ACTIVE pinterps and all possible dependency paths is required to find the K best global interpretations. For every f-neighboring pinterp N for the pinterp P at the head of a partial search path, the path-costs of extending that path by each possible b-dependency path between N and P must be considered. Whenever the cost of extending becomes greater than the cost of any of the current K best search paths, search can terminate for that search path.

Verifying that a particular sequence of states is consistent with the observations is much less expensive. Again, search is performed, but this time the given sequence of states strongly constrains what f-neighboring pinterps and dependency paths to consider.

Determining whether a particular state could occur during a certain period of time is even easier: just check if there is an ACTIVE pinterp for that state for any of the segments occurring during that time. Such queries would be useful when monitoring for dangerous or otherwise interesting states.

3.7 Finding Dependency Paths

Finding a dependency path involves finding a transition consistency relation between a pinterp and a neighboring segment. As shown in Figure 3.10, DATMI's algorithm for finding a dependency path first determines which types of relations could possibly hold. When the best b-dependency path is being sought, path-costs are used to indicate which relation is best. When the f-dependency path is being sought, or when path-costs are not being used, DATMI just uses the first relation found.

Given: Pinterp $P(G_g, S_s)$ and direction d

Returns: status of finding a dependency path (ie. succeed or fail)

```
1. G_n ← NEIGHBOR-SEGMENT(G_g, d).
2. If G_n = ∅
      then Call FIND-FRONTIER-STATE-PATH in direction d for P(G_g, S_s)
      else
        if GAP-FILL-SEGMENT?(G_n)
           then Call FIND-GAP-FILL-PATH in direction d for P(G_g, S_s)
3. Otherwise:
  Find a dependency path for P(G_g, S_s) in direction d, by calling:
    a. FIND-SPANNING-STATE-PATH,
    b. FIND-MEETING-STATES-PATH, or
    c. FIND-HIDDEN-TRANSITION-PATH.
  ; when d = BACKWARD and have path-costs, use best path of the three --
  ; else stop when find first one (since calls ordered by simplicity)
4. If fail to find a dependency path for pinterp P(G_g, S_s)
      then STATUS(P(G_g, S_s)) ← INACTIVE.     ; filter it
      else Record new dependency path and path-cost for P(G_g, S_s).
```

Figure 3.10: Procedure FIND-DEPENDENCY-PATH

The rest of this section describes how DATMI searches for each of these types of relations. It is assumed that one is looking for a dependency path for the pinterp $P(G_G, S_S)$ and that G_N is the neighboring segment in question. Since the enforcement of duration constraints is detailed in Chapter 6, this section only hints at how they impact these searches.

3.7.1 Finding Frontier-State, Spanning-State, and Meeting-States Paths

A frontier pinterp automatically has a transition consistency relation with the non-existent neighboring segment, unless duration constraints prevent one. For example, an instantaneous pinterp of a non-instantaneous frontier segment must depend on some hidden-transition path of non-instantaneous pinterps.

A spanning-state relation requires that the pinterp $P(G_N, S_S)$ is ACTIVE. In that case, the spanning of state S_S over the boundary between the neighboring segments G_G and G_N is consistent as long as duration constraints are also satisfied. Since this span may continue over a sequence of contiguous segments, the duration of that whole span is propagated, much as path-costs are. This accumulated duration indicates whether extending the span would exceed the upper-bound on the states duration.

Finding a meeting-states relation is also simple: find some ACTIVE pinterp of G_N whose state has an state transition with S_S, in the required direction.

3.7.2 Finding Gap-filling Paths

To efficiently find gap-filling dependency paths, another type of lookup-table is computed from the envisionment:

Definition 3.13 (Path lookup-table) *The* path lookup-table *indicates the (best) path of states that connects each pair of states in the envisionment.*

Currently, DATMI's path lookup-table is an NxN array, N being the total number of states. Each entry $A[i, j]$ contains the next state in the (best) path from state S_i to state S_j, as well as the cost of that path. The (best) path connecting S_i to S_j can be found by successively gathering the states $A[k, j]$ of the path to S_j from the current state S_k until $k = j$, with $k = i$ initially.

The best non-empty path of states connecting each state to itself is also represented in this table. They provide the best gap-filling path between the same instantaneous state of both neighboring segments of the gap-fill segment.

Generating the *shortest-path lookup-table* requires $\Theta(N^2)$ time to find the shortest path connecting each state pair. For each of the N states, the shortest paths to each of the other states can be found by breadth-first search that only has to reach each state once. If specific state and transition costs are available, then a *least-cost-path lookup-table* can be produced in $\Theta(N^3)$ time using standard graph-search algorithms (Mehlhorn, 1984). As with the state lookup-table, the path lookup-table is generated off-line.

An efficient gap-filling algorithm is crucial to DATMI. The interpretation of a gap-fill segment might be any path of states through the envisionment, since this segment provides no property constraints. Fortunately, an optimal gap-filling dependency path connecting two pinterps can immediately be determined by simply accessing the path lookup-table. Furthermore, whether some state can occur during the gap is indicated by whether there are some paths in the envisionment between that state and the states of ACTIVE neighboring pinterps.

Enforcing duration constraints on gap-filling dependency paths is difficult. The problem arises whenever the duration of the gap conflicts with the estimated duration of the gap-filling path. This problem could sometimes be resolved by considering part of the path to be a hidden-transition over G_G. This would reduce the duration of the portion of the path that is over the gap-fill segment.

DATMI currently does not allow gap-filling dependency paths to include hidden-transition paths in G_g. That assumption allows the gap-filling path to be indicated simply by the pinterp P at the other end of that path from $P(G_G, S_S)$. The complete path can then be quickly reconstructed at interpretation time from the path lookup-table. However, in a more general implementation where global constraints (see Section 8.3.2) might cause some pinterps of a gap-fill segment to become INACTIVE, the entire dependency path across a gap-fill segment would have to be recorded.

3.7.3 Finding Hidden-Transition Paths

Hidden-transition dependency paths are the most complex ones because they may consist of many ACTIVE pinterps from G_G. Yet, unlike for gap-filling paths, a lookup-table for hidden-transition paths would require space exponential in the number of states. This is because the space of possible hidden-transition paths depends on the specific set of ACTIVE same-segment pinterps, as well as the specific set of ACTIVE neighboring pinterps.

The efficiency of finding hidden-transition paths is critical, since every pinterp which is going to be filtered must first be checked for a hidden-transition dependency path. This check requires searching the entire space of possible hidden-transition paths [3]. This space is exponential in the number of ACTIVE pinterps of G_G and G_N.

This search starts with the pinterp $P(G_G, S_S)$, the set A of ACTIVE pinterps of segment G_G, and the set A_N of ACTIVE pinterps of neighboring segment G_N. A path must be found which connects $P(G_G, S_S)$ to some $P(G_N, S_f)$ in set A_N, using only pinterps in set A and transitions in the envisionment.

A simple method for finding such paths is to search breadth-first from $P(G_G, S_S)$ through the envisionment, while only expanding from states which have corresponding pinterps in set A, until some pinterp in set A_N is reached. By marking each state as it is examined, so that it is never examined again, this approach finds a shortest hidden-transition path.

To find the lowest-cost path, search must continue until all promising alternative paths have been explored. DATMI uses best-first search which discards partial search path as soon as its accumulated cost exceeds the cost of the current best path.

Unfortunately, this best-first search could require time exponential in $|A| + |A_N|$. However, simple graph algorithms can find the least-cost paths connecting all pairs of nodes in a graph of N nodes in fixed-cost $\Theta(N^3)$ time (Mehlhorn, 1984). from $P(G_G, S_S)$ fails to terminate after some predetermined $\mathcal{O}((|A| + |A_N|)^3)$ number of pinterp examinations. This exhaustive graph-search among the pinterps in $A + A_N$ ensures that the time complexity of hidden-transition path search remains within $\mathcal{O}((|A| + |A_N|)^3)$ while still finding the best dependency path. In practice, DATMI's best-first search usually finds hidden-transition paths well before the number of pinterp examinations reachs the predetermined cutoff.

[3] Unless, of course, one is willing to overlook some hidden-transition paths that are unlikely or involve too many states. Section 8.3.1.1 discusses the possibility of ignoring some transitions or states indefinitely.

Breath-first search is still performed, even when the best path is desired. This search provides a cheap way of checking that some hidden-transition path exists, before looking for the optimal one.

3.7.4 Improving Hidden-Transition Search

DATMI actually uses an improved version of the hidden-transition path-finding algorithm described above. For example, it removes every pinterp from set A which is also in set A_N. Once search reaches a state of an ACTIVE pinterp of the neighboring segment, that path always has less cost than any extensions.

Additional pinterps are removed from A if they cannot possibly belong to any acyclic hidden-transition path. Some of these useless pinterps can be identified by noting that the length of an acyclic path cannot be more than the number of pinterps from which to choose. So, a pinterp $P(G_G, S_i)$ is removed from A whenever either:

1. The length of the path for shortest-path lookup-table[S,i] > $|A|$.

2. The length of each path for shortest-path lookup-table[i,j] > $|A|$ for all $P(G_N, S_j)$ in A_N.

In the first case, $P(G_G, S_i)$ is removed because there simply are not enough pinterps in A to provide a hidden-transition path from $P(G_G, S_S)$ which goes through $P(G_G, S_i)$. Likewise, pinterps are removed from A_N which have a known shortest path from $P(G_G, S_S)$ which is longer than $|A| + 1$.

Additional techniques provide more efficient expansion from the current state being considered in the breadth-first search. During expansion, one must determine which pinterps in A or A_N correspond to states that are directly connected (via a state transition of the appropriate direction) to the current state in the envisionment. A *contenders-list* represents which pinterps of A and A_N have not yet been explored during expansions. As search progresses, pinterps are removed from this contenders-list.

So, one could expand from the current search state by determining either:

1. which states in the envisionment that are directly connected to the current state also correspond to pinterps of A or A_N, or

2. which pinterps in the contenders-list are directly connected to the current state.

Option 1 requires time proportional to the number of states directly connected to the current state. To achieve that time complexity, the states of the envisionment are tagged at the start of the hidden-transition search with the pinterps of A or A_N to which they correspond. By checking such tags, the determination of whether a state corresponds to a pinterp of A or A_n takes constant time. Alternatively, option 2 can be performed in time proportional to the number of pinterps in the contenders list. This is because whether a pinterp directly connects to the current state can be determined in constant time by referring to the shortest-path lookup-table.

Thus, the selection of an expansion method depends on the relation between the sizes of the contenders-list and the set of state transitions for the current state. If the number of remaining contenders is smaller than the number of states directly connected to the current state, DATMI uses option 2. Such an approach ensures that the hidden-transition search gets quicker as

more and more pinterps are eliminated from the list of contenders. At the same time, it takes advantage of cases where a state has few transitions.

Unfortunately, the hidden-transition path-finding algorithms described above are slightly incomplete because they would not find a dependency path with an instantaneous state occurring between two meeting segments. In theory, an instantaneous behavior, such as the collision of a ball with a wall, might not be captured with the given observations due to insufficient temporal resolution. This problem does not arise in practice because DATMI interprets numerical measurements with conservative translations into qualitative properties. Such translations always blur property value changes over some non-instantaneous period of time. Thus, the DATMI implementation does not worry about this case.

If conservative translations were not possible or desirable, this problem could be solved in a couple of ways. For one, a gap-fill segment of instantaneous time duration could be used between every pair of non-gap-fill segments. Gap-filling search would then try to find an instantaneous state for that gap-fill if some state cannot be found which spans between G_G to the other neighbor of the gap-fill segment. However, maintaining a history where every other segment is an instantaneous gap-fill segment would be cumbersome and expensive.

Alternatively, the basic hidden-transition algorithm could attempt to extend a working hidden-transition path which fails to reach any ACTIVE pinterp of the neighboring segment using only ACTIVE pinterps of G_G. The working path would be extended by some instantaneous states which need not be compatible with any segment properties for G_G. If the extended path can then reach an ACTIVE pinterp of the neighboring segment, that path would be a valid dependency path.

Chapter 4

ADJUSTING THE PINTERP-SPACE TO HANDLE FAULTY DATA

Sometimes the system model can offer no interpretation of the observations. As noted in Chapter 3, this is heralded by an *inconsistent segment*, a segment without ACTIVE pinterps. General strategies for handling an inconsistent segment include:

- **Giving-up**: Ignore the inconsistency and continue to interpret the other segments.
- **Changing-properties**: Activate pinterps by changing some segment properties, to reflect doubt in the original observations.
- **Model-refinement**: Change the physical model, to provide an alternative envisionment which provides ACTIVE pinterps.
- **Recruiting-unknowns**: Try activating pinterps with UNKNOWN status.
- **Incremental Envisioning**: Enhance a partial envisionment with additional states or transitions which yield new ACTIVE pinterps.

DATMI currently uses only the giving-up and changing-properties strategies, using the following three-step process:

1. Determine a set of changes in segment properties which might fix the inconsistent segment.
2. Apply the effects of those changes to the entire pinterp-space.
3. If inconsistency remains, retract all changes and then either retry step 1 or give up.

Giving-up is a last resort, for when no changes in the observations or model can be made within resource limits. The interpretation construction and filtering algorithms ignore an inconsistent segment. So, inconsistent segments partition the history into sub-histories that are separately interpreted and maintained. Whenever an inconsistent segment is fixed, the two neighboring partitions are merged into one. One could, therefore, postpone the handling of an inconsistent segment until sufficient computational resources are available.

Determining what segment property changes to try is difficult. In the worst case, accurate credit-assignment requires exponential search. How DATMI handles this problem is explained in Section 1.2. Algorithms for efficiently adjusting the pinterp-space to reflect property changes are then described in Section 1.3.

It should be noted that these strategies may be desirable even when no segment is actually inconsistent. For example, if the path-costs for all the ACTIVE pinterps of some segment are too high, then better pinterps for that segment might be requested. Such new pinterps might be provided by recruiting-unknowns, incremental envisioning, or model-refinement strategies. The effects of these new pinterps could be determined by using techniques similar to ones currently used by DATMI to propagate the effects of changing properties.

4.1 Faulty Data

How do inconsistencies arise? Some result from faulty models which give envisionments that are missing possible states or transitions. The other source of inconsistencies is *faulty data*. Faulty data can arise in three ways:

- **true noise**: random deviations in the sensor signal (such as white noise).
- **conversion failure**: an incorrect translation of the analog signal into qualitative values.
- **sensor failure**: the sensor is operating abnormally.

When numerical data are translated into conservative qualitative properties (as discussed in Chapter 2), true noise rarely results in property assertions that disagree qualitatively from the actual behavior. Smoothing sensor signals using traditional engineering techniques, such as Gaussian convolution, typically suffices to capture overall trends.

Unfortunately, such smoothing can sometimes hide qualitative changes which are actually occurring. One might try to first interpret the unsmoothed data and then try interpreting smoothed data if inconsistencies arise. Thus, a type of conversion failure would be detected which might be fixed by changing the segment properties to reflect the smoothed data. However, complex control issues, such as how to efficiently find the right grain-size (perhaps by successive smoothings at different levels), must then be addressed. Because of such difficulties, DATMI does not try such fixes.

Conversion and sensor failures always require adjusting the segment properties to recover from misleading data, since they cannot be prevented by initial preprocessing. Conversion failure occurs when the data sampling rate is too slow to capture every relevant qualitative change. Properties measured as constant, using a suspiciously slow sampling rate, would be candidate sources of inconsistency. Sensor failure occurs when the sensor breaks and produces misleading information.

4.2 Generating Possible Fixes to the Pinterp-Space

Fixing a pinterp-space with an inconsistent segment involves first determining which observations to doubt. Within the DATMI framework, this involves postulating which properties of which segments might have caused a segment G_I to have no ACTIVE pinterps. A candidate set of dubious observations is indicated by a *fix-hypothesis*:

Definition 4.1 (Fix-hypothesis) *A* fix-hypothesis *consists of a set of doubted properties ρ_j for each corresponding segment $G_j \in G$, where G is the set of all segments with doubted properties.*

Several issues complicate the search for plausible fix-hypotheses:

- ***Fracturing*** – one fault can misrepresent the same system variable over many contiguous segments.
- ***Hidden faults*** – the fault may have started before any inconsistency actually arose.
- ***Multiple faults*** – different faults might each cause different segment properties to be incorrect.

Fracturing is caused by the splitting of property assertions during global segmentation. Because of this fracturing, it is not always sufficient to change some properties of just one segment. Furthermore, an inconsistent segment need not have any incorrect properties itself, since faulty data globally affect the pinterp-space. Consider some other segment G_B to have incorrect properties, instead of the inconsistent segment G_I. Even if G_I has some COMPATIBLE pinterps, all of its pinterps will be INACTIVE whenever some INCOMPATIBLE pinterps of G_B must be ACTIVE for some of G_I's pinterps to have dependency paths.

While the current implementation of DATMI handles only sensor failures, the DATMI framework suffices for both sensor failures and conversion failures. For either class of failure, DATMI only generates candidate fix-hypotheses which suggest *forgetting* some segment properties. This reflects strong uncertainty about what is really happening to the system variables of those properties. Such forgetting is conservative because it never leads to any ACTIVE pinterps becoming INACTIVE or INCOMPATIBLE. In contrast, changing the values of some properties would introduce another source of faulty data to worry about – the fix-hypotheses themselves.

So, DATMI generates candidate fix-hypotheses each consisting of a set G of segments, where each segment $G_i \in G$ has some properties ρ_i to be forgotten. Trying all 2^A fix-hypotheses, where A is the total number of property assertions over all segments, is intractable. So, DATMI focuses generation using domain-specific knowledge about:

1. The *plausibility* of doubting each segment property.
2. The *conditions* under which each fix-hypothesis is applicable.

A fix-hypothesis must be retracted whenever its conditions are violated while processing later observations.

The next two sections discuss the generation of fix-hypotheses for recovering from sensor failures and conversion failures. Section 1.4 suggests ways to improve DATMI's generation of fix-hypotheses.

4.2.1 Sensor Failure Fix-Hypotheses

A sensor failure fix-hypothesis suggests some sequence of segments over which a property was incorrectly indicated by a failed sensor. For example, consider an indicator light which is suppose to be ON when a pump is pumping and OFF when is not. A sensor failure occurs when that light burns out and is OFF even though the pump is actually pumping. So, if the light is

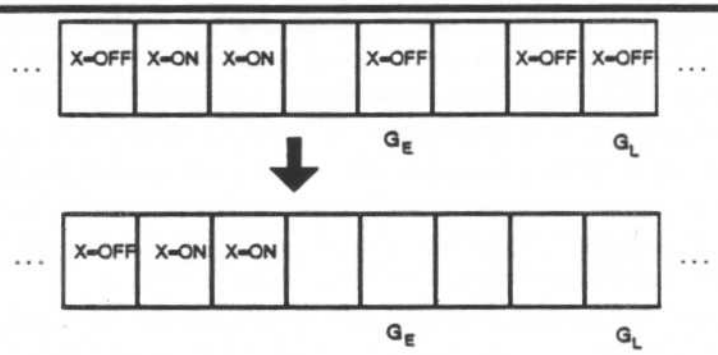


Figure 4.1: Example of a sensor failure fix-hypothesis

This hypothesis H represents doubt that properties of name X has value OFF from G_E to G_L.

OFF and there is an inconsistent segment, DATMI will generate a fix-hypothesis that the indicator has failed.

A simple sensor failure fix-hypothesis H postulates that some properties, all of some name X, are incorrect. Figure 1.1 gives an example of a sensor failure hypothesis for properties of name X. Let v be the value of a property of name X for the latest segment G_L that asserts any value for X. Also, let segment G_E be the earliest segment that asserts v for X without any segment G_k where $G_E \rightsquigarrow G_k$ and G_k asserts X to be something other than v. Then, set G for hypothesis H will consist of all segments from G_E through G_L which assert a value for X, namely value v. For each $G_i \in G$, ρ_i is just the singleton set of the property with value v and name X. One must doubt as far back as G_E to allow for the possibility that there was a hidden fault for X anywhere from G_E to G_L.

The key condition for the fix-hypothesis H is that later observations do not assert a value f of X for some future segment G_F (i.e. $G_L \rightsquigarrow G_F$), such that f differs from the value v of X during G_L. This condition reflects the two simplifying assumptions for sensor failures used by DATMI:

1. A failed sensor produces qualitatively identical values once it fails, regardless of the true behavior.

2. DATMI will be told when and if a sensor is fixed.

When DATMI is told that a sensor has been fixed at time t, this condition is relaxed. The hypothesis will then not be retracted whenever values other than v are observed after t. These two assumptions mean that DATMI considers only non-intermittent sensor failures. Because of these assumptions, DATMI does not attempt to recover from sensor failures that are harder to detect, such as spikes in the data (e.g. an indicator light momentarily disconnected from its power supply due to some vibrations).

Many types of sensor failures do satisfy these assumptions. As previously mentioned, an indicator might stay in the OFF position permanently if it breaks. Similarly, a CHANGE property would always get a value of STEADY if the sensor is jammed. Also, an ORDER property can get faulty values when one of the sensors becomes uncalibrated. However, due to assumption 1

above, DATMI only handles miscalibrations that are great enough to always produce the same qualitative value of GREATER or LESS. This is actually desirable, since otherwise a miscalibration hypothesis would never be retracted.

As these examples suggest, a sensor failure satisfying assumption 1 often results in only some particular subset of possible property values. So, a fix-hypothesis H is only generated when the value v for a property named X could result from that sensor failure. DATMI considers the plausibility of doubting the properties in H to be the *a priori* probability that the constant value v for X is due to the sensors of X failing. Such *a priori* probabilities are either supplied by external, domain-specific means or assumed to be equal for all values.

DATMI also currently assumes that each sensor contributes to only one type of property. If a single sensor provides the values for properties of several different names (such as a CHANGE property as well as part of an ORDER property), then the observed values for all these properties would need to be examined. If recent properties of each name indicated a sensor failure hypothesis for that sensor, then a fix-hypothesis for forgetting some properties of each name should be generated and tested just like a simple fix-hypothesis.

4.2.2 Conversion Failure Fix-Hypotheses

A conversion failure hypothesis identifies some segment properties that might be incorrect due to errors in the conversion of numerical measurements into qualitative properties. Although the DATMI implementation considers only sensor failure hypotheses, this section suggests a natural means for also handling conversion failures in this framework.

As discussed in Section 2.3, DATMI assumes that conversion failures never occur for a property ρ while the sample time ST(ρ) for ρ is less than MIN-ST(ρ). Let the value of ST(ρ) be the maximum of the sample times of all the measurements for a property ρ for a particular segment. By using a simple formula for each segment property ρ, such as:

$$\texttt{plausibility-of-doubting}(\rho) = (\texttt{ST}(\rho) - \texttt{MIN-ST}(\rho)) \cdot \texttt{sensor-unreliability}(\rho),$$

one could easily determine a reasonable ordering for doubting segment properties due to possible conversion failures. The unreliability in the sensors for ρ might be the maximum of the unreliabilities for all the measurements merged into ρ.

An ordered list of the K (for some predefined K) segment properties with the highest plausibility of doubt could be incrementally maintained during segment merges to allow efficient selection of which segment property to doubt next. A simple hypothesis generation technique would be to temporarily forget these properties in order of plausibility until the pinterp-space is fixed. Then, try to re-assert each forgotten property to reduce the number of properties actually forgotten. These re-assertions would proceed with the least dubious properties first. This technique would recover even from multiple conversion failures, assuming K is large enough.

A disadvantage of this technique is that the locality of faults is ignored. All segment properties with large ST(ρ) will be doubted before any properties with almost as large an ST(ρ) which are in segments that happen to be temporally much closer to the inconsistent segment G_I. In practice, properties of segments much closer to G_I are more likely to constrain it. Thus, those closer segment properties are more likely to contribute to the inconsistency. An interleaved scheme where some number of closest properties are also forgotten in order of their measure of doubt is required to enforce some locality.

The key conditions for a conversion failure fix-hypothesis are the values of MIN-ST(ρ) for each segment property ρ it forgets. If MIN-ST(ρ) drastically lowers, then the relative doubt

of ρ compared to other segment properties may also drop drastically. Automated methods for determining MIN-ST(ρ) and MAX-ST(ρ) for the observed properties might make such drastic changes. If the affected segment properties are now more dubious than the forgotten segment properties, these highly dubious ones should be forgotten. Then an attempt to re-establish those previously forgotten properties, in reverse order of doubt, should be made.

4.3 Applying a Fix-Hypothesis To the Pinterp-Space

Given a fix-hypothesis, DATMI propagates its effects and determines if the pinterp-space becomes consistent. If the pinterp-space remains inconsistent, then the effects of this hypothesis are retracted. Alternative hypotheses are likewise tested until the pinterp-space is fixed. If the set of hypotheses is exhausted, DATMI gives up and accepts the inconsistency. Since DATMI uses fix-hypotheses which only forget properties, this propagation process cannot result in any ACTIVE pinterps becoming INACTIVE.

Figure 1.2 presents the algorithm for adjusting the pinterp-space based on a fix-hypothesis which forgets some segment properties. First, those properties are removed from their segments. Next, all pinterps which change from INCOMPATIBLE to COMPATIBLE, due to the reduced property constraints, are considered INACTIVE. Finally, the procedure PROPAGATE-NEW-PINTERPS (as shown in Figure 1.3) is used to activate all pinterps which newly satisfy the transition constraints because of these newly COMPATIBLE pinterps.

Lower-cost dependency paths can become available when pinterps are newly activated. Step 2f of procedure PROPAGATE-NEW-PINTERPS ensures that such better dependency paths are found. Step 2f allows step 2h to propagate path-cost decreases forward from current segment G_i, by updating the path-costs of pinterps in G_i to reflect the changes caused by the backward propagation of step 2c.

The ability to propagate activations of pinterps allows DATMI to easily batch process observations for several segments at once. The ability to interleave batch and incremental processing of observations could support more flexible real-time reasoning. Figure 1.4 gives an algorithm for such batch processing. This algorithm uses PROPAGATE-NEW-PINTERPS to find dependency paths for pinterps of segments having their initial properties asserted while using procedure REFINE-PINTERPS (recall Figure 3.5) to propagate the effects of asserting additional properties for segments. Step 6 propagates the effects of asserting initial properties for segments, by first seeking initial transition consistency relations between those segments and the pinterps of the neighboring segments.

The process of propagating activation is detailed in Figure 1.5. Each propagation sweep proceeds until either: (a) no reactivations occur for some segment or (b) a frontier segment is updated. These sweeps first allow INACTIVE pinterps to be part of dependency paths (in step 5), in case some of the newly activated pinterps can activate those pinterps as well. Then, they check (in step 7) whether all the pinterps of such a path have become ACTIVE. If they are not, the PROPAGATE-FILTERING-EFFECTS of Figure 3.6 is invoked.

Of special interest in this procedure is the need to invoke the propagation of path-costs from the seed-segment G_g to its f-neighbor when propagating activation forward, as noted in step 3. Such a propagation is necessary because some ACTIVE pinterps of the f-neighboring segment might have a lower-cost alternative b-dependency path using the enhanced pinterps of G_g. The

Given: fix-hypothesis H for inconsistent segment G_I.
H suggests forgetting properties ρ_i for each corresponding segment G_i of list G.
(G is ordered from left-most to right-most segments).

```
1. For G_i ∈ G do
     a. SEG-PROPS(G_i) ← SEG-PROPS(G_i) − ρ_i.
     b. If SEG-PROPS(G_i) = ∅
          then
              i. Mark G_i as gap-fill segment.
             ii. Merge G_i with any neighboring gap-fill segments.
          else
            For S_j ∈ COMPATIBLE-STATES(SEG-PROPS(G_i)) do
              If STATUS(P(G_i, S_j))=INCOMPATIBLE
                then STATUS(P(G_i, S_j)) ← INACTIVE.
2. Call PROPAGATE-NEW-PINTERPS(G).
```

Figure 4.2: Procedure TRY-FORGETTING-FIX-HYPOTHESIS

set of enhanced pinterps consists of the newly reactivated pinterps of G_g and any pinterps of G_g which have had their own path-costs lowered.

4.3.1 Propagating Path-Cost Decreases

Figure 1.6 gives the procedure INSTALL-LOWER-COST-DEPENDENCIES to minimize path-costs after propagating activations backwards. Forward propagation of activation automatically propagates the effects of path-cost decreases. This is because the path-costs of all the pinterps of the b-neighboring segment already had their path-costs updated when b-dependency paths were sought for them. However, after backward propagation of activation, some pinterps in the seed-segment may have lower-cost chains of b-dependency paths back to some pinterps in the stop-segment, because newly activated pinterps were not considered. So, the path-costs of the pinterps in the stop-segment must be propagated forward, all the way to the pinterps of the seed-segment.

The main idea is to sweep forward from the stop-segment (INSTALL-LOWER-COST-DEPENDENCIES's start-segment) to the seed-segment (finish-segment) until no more improvements in the path-costs can be made. During this sweep, few of the ACTIVE pinterps of the f-neighboring segment actually look for better b-dependency paths. Let c be the path-cost of a pinterp $P(G_g, S_s)$, minus the cost of the transition to it in its b-dependency path and the cost of S_s. Also, let l be the lowest of the path-costs of the enhanced b-neighboring pinterps. $P(G_g, S_s)$ needs to see if it has a better b-dependency path only if c is higher than l. That is, if the path-cost is already at least as low as the best conceivable lowest-cost b-dependency path would allow, then there is no need to search for a better one.

Note that the forward sweep of INSTALL-LOWER-COST-DEPENDENCIES stops as soon as one of the following occur: (a) the given finish-segment is reached or (b) no more reductions in

Given: list G of segments G_i, where G ordered from left-most to right-most segments.

```
1. For G_i ∈ G do
     When GAP-FILL-SEGMENT?(G_i)
      a. Replace G_i in list G by its b-neighbor.
         ; or the f-neighbor if the backward one does not exist.
      b. For d ∈ {BACKWARD,FORWARD} do
            G_n ← NEIGHBOR-SEGMENT(G_i,d).
            For P ∈ ACTIVES(G_n) do
                 i. STATUS(P) ← INACTIVE.
                ii. Remove P from ACTIVES(G_n).
               iii. Add P to INACTIVES(G_n).        ; will reactivate in step 2
2. For each G_i, in the given order of G do
   a. A_o = ACTIVES(G_i).
   b. Assume that each INACTIVE pinterp of G_i is ACTIVE.
   c. Call PROPAGATE-PINTERP-ACTIVATION(G_i,BACKWARD).
   d. G_s ← PROPAGATE-PINTERP-ACTIVATION's stop-segment G_Z.
   e. A ← PROPAGATE-PINTERP-ACTIVATION's reactives R_Z.
   f. Call INSTALL-LOWER-COST-DEPENDENCIES(G_s,G_i,A).
   g. E ← ACTIVES(G_i) − A_o.   ; remaining reactives
   h. Call PROPAGATE-PINTERP-ACTIVATION(G_i,FORWARD,E).
   i. G_s ← PROPAGATE-PINTERP-ACTIVATION's stop-segment G_Z.
   j. For G_g ∈ G do If G_g ⇝ G_s then G ← G − G_g.
      ; all consequences of G_g have already been propagated
      ;  when activation propagation passes up that segments
```

Figure 4.3: Procedure PROPAGATE-NEW-PINTERPS

Given: new properties ρ_i for each updated segment G_i of list G, where G is ordered from left-most segments to right-most segments.

1. $I \leftarrow \emptyset$. ; segments with initial properties
2. $U \leftarrow \emptyset$. ; updated segments
3. For each segment G_i of G in order of G do
   a. If ρ_i are the initial properties for G_i, then
      i. $C \leftarrow$ COMPATIBLE-PINTERPS(ρ_i, G_i)
      ii. INACTIVES(G_i) $\leftarrow C$.
      iii. Enqueue segment G_i onto queue I.
   b. Else, Enqueue G_i onto queue U.
4. Call PROPAGATE-NEW-PINTERPS(I).
5. For $G_i \in U$ do Call REFINE-PINTERPS(G_i, ρ_i).
6. For each segment G_i of I in given order of I do
   For $d \in$ {BACKWARD, FORWARD} do
      a. $G_n \leftarrow$ NON-GAP-FILL-NEIGHBOR(G_i, d).
      b. $F \leftarrow \emptyset$. ; filtered pinterps
      c. Unless $G_n \in G$
         i. For $P(G_n, S_s) \in$ ACTIVES(G_n) do
            Call FIND-DEPENDENCY-PATH($P(G_n, S_s)$, REV-DIR(d)).
            If fail then $F \leftarrow F \cup P(G_n, S_s)$.
         ii. Call PROPAGATE-FILTERING-EFFECTS(G_n, F).

Figure 4.4: Procedure BATCH-UPDATE-PINTERP-SPACE

Given: seed-segment G_g and propagation direction d.
Also given set E of enhanced pinterps if propagating in forward direction.
The inconsistent segment G_I and current fix-hypothesis H are also known.
Returns: stop-segment G_Z and its reactivated pinterps R_Z.

```
1. R_Z ← ∅.
2. G_n ← NON-GAP-FILL-NEIGHBOR(G_g, d).
3. When d = FORWARD
   a. Call INSTALL-LOWER-COST-DEPENDENCIES(G_g, G_g, E).
   b. B ← INSTALL-LOWER-COST-DEPENDENCIES's enhanced pinterps L.
4. R ← ∅.    ; reactivated pinterps
5. For P(G_n, S_s) ∈ INACTIVES(G_n) do
   a. Call FIND-DEPENDENCY-PATH(P(G_n, S_s), d), allowing
      INACTIVE pinterps of G_n to be used in hidden-transition paths.
      ; All dependencies for newly activated pinterps will be confirmed
      ; later when G_n becomes G_g during the propagation.
   b. If succeed, then
      i. STATUS(P(G_n, S_s)) ← ACTIVE.
     ii. R ← R ∪ P(G_n, S_s).
6. U ← ∅.    ; unconfirmed pinterps
7. For P(G_n, S_s) ∈ R do
   When some P(G_n, S_i) ∈ DEPENDENCIES(P(G_n, S_s), d)
    such that INACTIVE?(P(G_n, S_i))
     a. Call FIND-DEPENDENCY-PATH(P(G_n, S_s), d),
        this time not allowing INACTIVE pinterps (unlike step 5a).
     b. If fail then R ← R − P(G_n, S_s), U ← U ∪ P(G_n, S_n).
8. Call PROPAGATE-FILTERING-EFFECTS(G_n, U, d).
   ; Updates pinterps depending on any unconfirmed pinterps of U.
   If an inconsistent segment G is detected during this propagation,
    then Call RETRACT-FIX-HYPOTHESIS(H, G).
9. If R = ∅ then G_Z ← G_g
     else
       R_Z ← R.
       If FRONTIER-SEGMENT?(G_n) then G_Z ← G_n
         else
           G_g ← G_n.
           If d = FORWARD then E ← B ∪ R.
           Goto step 2.
```

Figure 4.5: Procedure PROPAGATE-PINTERP-ACTIVATION

Given: start-segment G_s, finish-segment G_f, and set of enhanced pinterps E of G_s.
Returns: final enhanced pinterps L.

```
1. G_n ← NON-GAP-FILL-NEIGHBOR(G_s, FORWARD).
2. If G_n = ∅ then exit.
3. C ← lowest path-cost of all pinterps in E.
4. L ← ∅.    ; pinterps with lowered path-costs
5. For P(G_n, S_i) ∈ ACTIVES(G_n) do
   When PATH-COST(P(G_n, S_i)) > C
     ; not counting itself and the immediate transition from it
     a. Z ← PATH-COST(P(G_n, S_i)).
     b. call FIND-DEPENDENCY-PATH(P(G_n, S_i), BACKWARD).
     c. If path-cost of P(G_n, S_i) is now lower than Z
          then L ← L ∪ P(G_n, S_i).
6. Unless L = ∅ or G_s = G_f
     G_s ← G_n.
     E ← L.
     Goto step 1.
```

Figure 4.6: Procedure INSTALL-LOWER-COST-DEPENDENCIES

path-costs can be made. This finish-segment is typically the seed-segment of the sweeps for propagating activation. As mentioned above, stopping at the seed-segment is sufficient since forward propagation of activation automatically propagates path-cost decreases.

It is not sufficient to only propagate path-cost decreases along the existing dependency paths. For example, let $P(G_n, S_q)$ be the b-neighboring pinterp of the current b-dependency path for a pinterp P. If some other neighboring pinterp $P(G_n, S_r)$ gets a reduced path-cost and could allow a better b-dependency path for P, then P should use the b-dependency path containing $P(G_n, S_r)$ instead. However, having $P(G_n, S_r)$ only tell the pinterps that b-depend on it that its cost was reduced would fail to update P. Therefore, one must instead look at all the ACTIVE pinterps of a segment, to see which ones might benefit from any reduced path-costs of the b-neighboring pinterps.

4.3.2 Retracting a Fix-Hypothesis

When applying a fix-hypothesis does not fix an inconsistent segment, DATMI retracts that fix-hypothesis. Figure 1.7 outlines this retraction process. This same procedure is used when future observations indicate that a fix-hypothesis that had been applied successfully violates the conditions of that fix-hypothesis. Retracting a fix-hypothesis will again result in some inconsistent segment, unless sufficient other relaxations to the constraints on the pinterp-space have occurred since it was first applied.

Given: fix-hypothesis H for inconsistent segment G_I.
H suggests forgetting properties ρ_i for each corresponding segment G_i of list G, which is ordered from left-most to right-most segments.

```
1. For each segment G_i, in order of G, do
     ; recover the properties forgotten for fix-hypothesis H
     Call REFINE-PINTERPS(G_i, ρ_i).
2. If no pinterps are ACTIVE for some segment G_B, during
   any propagation sweep used by REFINE-PINTERPS in step 1, then
     ; detected inconsistent segment G_B!
     a. If currently fixing the pinterp-space, then
        i. If G_B ≠ G_I
              then Add G_B to I_G.
                   ; Where I_G is the global set of
                   ; inconsistent segments detected since the first
                   ; inconsistent segment was detected, for the
                   ; current observations.
              else ignore inconsistency  ; it's being worked on ...
       ii. Return to the caller (PROPAGATE-PINTERP-ACTIVATION),
           performing step iii below afterwards (by which
           time the global I_G may have changed):
      iii. For G ∈ I_G do
              When ACTIVES(G) = ∅
                 ;; G didn't happen to get fixed when G_I was.
                 Try to fix G      ; like in step 2b. below
           ; Tries to fix an inconsistent segment that was not
           ; originally handled because PROPAGATE-FILTERING-EFFECTS
           ; stops at the first inconsistent segment it sees.
     b. Otherwise:
        ; fix-hypothesis H previously worked but now needs to be retracted
        Handle G_B like any other inconsistent segment, using
        TRY-FORGETTING-FIX-HYPOTHESIS to test possible alternative
        fix-hypotheses.
```

Figure 4.7: Procedure RETRACT-FIX-HYPOTHESIS

4.4 Improving Fix-Hypothesis Generation

Additional heuristics and domain-specific knowledge could further guide DATMI's ordered generation of fix-hypotheses. Because the search space of possible fixes is so large, strong focus is necessary to allow more robust handling of faulty data. This section considers three especially desirable measures of a fix-hypothesis H that could help improve DATMI's current performance:

- *fix-cost* (C_H) – cost of updating the pinterp-space,
- *fix-hope* ($\mathcal{H}_H$) – likelihood that H will work, and
- *fix-utility* ($\mathcal{U}_H$) – overall desirability of H.

Of course, to be useful, fix-cost and fix-hope must be estimated before actually applying the fix to the pinterp-space. However, fix-utility might first be roughly estimated to provide initial priority and then be adjusted once the fix is applied.

4.4.0.1 Estimating Fix-Cost C_H

The fix-cost C_H estimates the number of changes in the pinterp dependencies required to apply H. Since a change in a pinterp's dependencies might propagate into changes in any other pinterp's dependencies, the fix-cost is extremely difficult to determine accurately without actually propagating the changes. Yet, one would at least like to first try fixes requiring almost no changes in the pinterp-space before trying fixes that would require many changes.

One simple good heuristic is that the fix-cost is likely to be proportional to the number of pinterps which become COMPATIBLE when properties are forgotten. The fix-cost is likely to be further increased when these pinterps are more equally distributed among all segments, since each of those changes are most likely to affect the segments closest to them. Since in practice only a small fraction of the COMPATIBLE pinterps tend to be ACTIVE as well, these heuristics tend to overestimate the fix-costs; these overestimations provide upper bounds on the fix-costs that are useful for real-time processing.

4.4.0.2 Estimating Fix-Hope $\mathcal{H}_H$

For H to succeed, its adjustment must make some pinterp become ACTIVE for the inconsistent segment G_I. A measure for fix-hope should reflect the notion that more property changes in the vicinity of G_I increase the likelihood that the inconsistency will be removed. Thus, a simple estimate of the fix-hope $\mathcal{H}_H$ would be one proportional to the estimate for fix-cost given above.

Inconsistency often arises when none of the pinterps for G_I are even COMPATIBLE. For example, consider the pump-cycle of Figure A.1. No state will be compatible with properties asserting that the water levels of both containers are increasing at the same time. Forgetting a property for a segment other than G_I will not resolve the inconsistency – at least one property over G_I itself must be forgotten. However, even this is not enough to resolve the inconsistency in general. Thus, the fix-hope in these cases should be zero unless H allows some pinterps of G_I itself to become COMPATIBLE.

4.4.0.3 Estimating Fix-Utility u_H

The fix-utility measure u_H for a fix-hypothesis H can be based on two main factors: domain-specific knowledge about the sensors and task-specific preferences on the alternative consistent interpretations. Domain-specific sensor knowledge can influence the initial estimate of fix-utility. For example, each fix-hypothesis's plausibility measure estimates how *a priori* likely it is that the data removed by the fix are actually faulty. Once the inconsistency is fixed, the fix-utility might be adjusted to reflect the path-cost or path-probability for the best global interpretation in the resulting pinterp-space.

4.4.0.4 Discussion of Fix-Hypothesis Generation

The availability of these three measures would allow generation to focus resources on the best trade-offs between expected time of adjustment and quality of interpretations. Real-time processing would typically demand that adjustment costs are small whereas an analysis of long-term behavior might favor interpretations that maximize interpretation credibility in light of possible faults. An overall score for each hypothesis could be based on a task-dependent weighted sum of these c_H, $\aleph_H$, and u_H factors to determine which hypothesis to try next. Search for the fix which maximizes the u_H could proceed until time limitations were reached. The overall score should also depend on how much processing time remains. For example, for real-time reasoning, one might first ensure that some working fix is available and then use the remaining time to find one having much higher fix-utility.

Fix-hypothesis generation should also consider multiple faults. Inconsistencies due to multiple faults might involve both sensor failures and conversion failures. The method for handling multiple conversion failures suggested in Section 1.2.2 – ordered forgetting of segment properties until the inconsistency is removed and then re-asserting as many of them as possible – could also handle multiple faults in general. However, that approach could easily result in compounds fixes which forgot too many different types of properties when each hypothesis suggested forgetting only a couple of segment properties. To help ensure that properties of only a few different names are doubted, one could prefer fix-hypotheses that forget segment properties having the same names as properties already forgotten for other segments by previously applied fix-hypotheses.

Chapter 5

MAINTAINING INTERPRETATION CREDIBILITIES

The algorithms discussed so far have focused on maintaining the space of consistent interpretations. Being aware of these alternative interpretations is essential to conservative monitoring and to handling faulty data. However, to accurately predict the future behavior of the system, one must determine its most likely current state. Similarly, to explain or summarize observations, one may want the most likely or simplest global interpretation.

DATMI provides a means for determining these best interpretations. It uses a generalized notion of the *path-costs* introduced in Section 3.5, as follows:

Definition 5.1 (Interpretation credibility) *A pinterp's* path-credibility *is the real-valued measure of the probability or inverse cost of the chain of b-dependency paths leading to it. An* interpretation credibility *is the path-credibility of the final pinterp in that global interpretation.*

Path-credibilities are maintained using the same algorithms as for path-costs, although they are composed either additively or multiplicatively, as appropriate.

For example, the use of path-costs in Section 3.5 employs additive composition. To provide a basic measure of the simplicity of an interpretation, one could use 1 as the cost of each transition and 0 as the cost of each state. Alternatively, to better detect behaviors which could lead to dangerous states (like those where a factory explodes), one could give the lowest costs to all states that have paths to those dangerous states.

The rest of this chapter describes how DATMI provides a probabilistic measure of interpretation credibility by using multiplicative composition of Bayesian conditional probabilities. In this case, each path-credibility is referred to as a: *path-probability*:

Definition 5.2 (Path-probability) *A pinterp's* path-probability *is the probability that the the chain of b-dependency paths reaching that pinterp represents the actual behavior.*

5.1 Determining Path-Probabilities

As described below, each path-probability is locally composed from two sources:

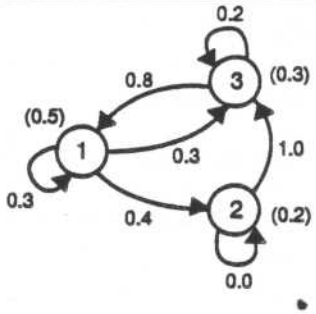


Figure 5.1: Example of a probabilistic envisionment

Each transition is labelled with its conditional probability while each state is labelled with its *a priori* probability (in parentheses). State S_2 is an instantaneous state.

- ***state and transition probabilities*** (model-based) and
- ***property probabilities*** (observation-based).

5.1.1 Using State and Transition Probabilities

For a given domain, there are often common sense or empirical notions of the relative likelihoods that various states or transitions occur. For example, one might know from experience that when a system is in a particular state, it almost always proceeds to a certain other state next. To represent such knowledge, DATMI can use:

Definition 5.3 (State and transition probabilities) *The* state probability *of state S_s is the* a priori *probability $Prob(S_s)$ of being in S_s at the start of the observations. The* transition probability *of a transition $S_s \rightarrow S_f$ from some state S_s to another state S_f is the conditional probability $Prob(S_f \mid S_s)$ (also denoted as $Prob(S_s \rightarrow S_f)$) of being in S_f as soon as S_s ends.*

Figure 5.1 provides an example envisionment which is augmented with state and transition probabilities.

$Prob(S_s \rightarrow S_s)$ is the probability that a system stays in state S_s from one segment to another. To strictly adhere to probability theory, a particular transition probability $Prob(S_s \rightarrow S_f)$ should only indicate the probability of $S_s \rightarrow S_f$ occurring at one particular time. DATMI assumes that it is acceptable to use the same value of $Prob(S_s \rightarrow S_f)$ no matter what time $S_s \rightarrow S_f$ occurs or how long S_s had lasted before this transition. Motivations and consequences of this assumption are discussed in Section 5.1.3. Typically, this simplification is sufficient for at least finding one of the more likely interpretations.

DATMI assumes that some domain-specific sources provide these state and transition probabilities. For example, they might be obtained by stochastic analysis techniques, such as those developed in (Doyle & Sacks, 1989). In any case, the following conditions will hold for any complete and consistent envisionment of N states:

1. $\sum_{i=1}^{N} Prob(S_i) = 1$.
2. $Prob(S_s \rightarrow S_f) = 0$ whenever there no transition $S_s \rightarrow S_f$.

3. $\sum_{i=1}^{N} Prob(S_s \rightarrow S_i) = 1$, for each state S_s.

4. $\sum_{i=1}^{N} Prob(S_i \rightarrow S_s) = 1$, for each state S_s.

5. $Prob(S_s \rightarrow S_s) = 0$ for any instantaneous state S_s.

6. $Prob(S_s \rightarrow S_s) = 1$ for any state S_s with no transitions to others.

In lieu of sufficient domain-specific information, remaining probabilistic weight is uniformly distributed in agreement with these conditions.

5.1.2 Using Property Probabilities

One typically also has some sense of the uncertainty in the observations. As explained in Section 2.3, such uncertainty is represented by the probabilistically-weighted, disjunctive properties given by DATMI's quantity-space conversion tables or the property probabilities given directly by the OBSERVE predicates.

Definition 5.4 (Property probability) *Each* property probability $Prob(p = v)$ *for a segment is the discrete probability that the property named p has the value v for the actual behavior during that segment.*

For example, imagine a segment G_g with k asserted properties of names $p_{i_1}, \ldots, p_{i_k}$. Furthermore, let v_{j_l} be the value in state S_s for the property named p_{i_l}, for $l = 1$ to k. Then, the probability that the actual values of those k observed properties agree with the properties of S_s during G_g is:

$$\texttt{PROPS-PROB}(P(G_g, S_s)) = Prob(p_{i_1} = v_{j_1}) \cdot \quad \ldots \quad \cdot Prob(p_{i_k} = v_{j_k}).$$

This composition assumes that each segment property is independent, which is most reasonable when the observations are never redundant.

As mentioned in Section 2.4, all assertions of properties of a particular name over a given segment must agree both in the property values and the probabilities of those values. This can require a single segment to be partitioned into a sequence of neighboring segments all agreeing in the property values but significantly differing in the probabilities of those values. Such a history is referred to as a:

Definition 5.5 (Globally-segmented certainty-partitioned history) *A* globally-segmented certainty-partitioned history *is a globally-segmented history which is concise so long as each segment property has roughly uniform probability throughout its segment, as determined by domain-specific thresholds.*

A globally-segmented certainty-partitioned history can require significantly more segments than a *globally-segmented concise history* for the same set of observations. However, when the sources of observations are unreliable and noisy, globally-segmented certainty-partitioned histories allow property probabilities to more finely weight the global interpretations according to the varying certainties in the observations.

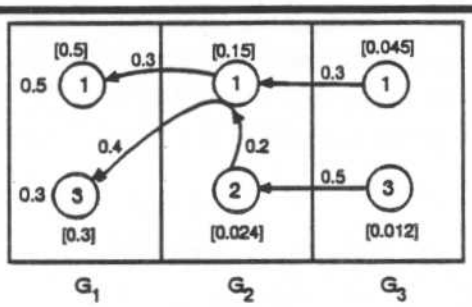


Figure 5.2: Example pinterp-space with path-probabilities

Each transition in a dependency path is labelled with its conditional probability. Note that the probabilities for non-spanning transitions between pinterps are half of the probability for the transition between the corresponding states. Also, the *a priori* probabilities for states in the first segment are given by the over-sized numbers next to those states. The composed path-probability for each pinterp are given in brackets.

5.1.3 Locally Composing Path-Probabilities

Figure 5.2 shows a simple pinterp-space for the envisionment of Figure 5.1. Path-probabilities are shown for each pinterp based on the state and transition probabilities specified for that envisionment. Property probabilities are ignored for this simple example.

The path-probability of a pinterp $P(G_g, S_s)$ depending on $P(G_b, S_d)$ of the b-neighboring segment G_b is defined in terms of the local state, transition, and property probabilities as:

$$\begin{aligned}
\texttt{PATH-PROB}(P(G_g,S_s)) = \texttt{b-path-prob} \cdot\ & \\
& \texttt{PROPS-PROB}(P(G_g,S_{d+1})) \cdot \tfrac{1}{2} \cdot Prob(S_{d+1} \rightarrow S_{d+2}) \cdot \\
& \texttt{PROPS-PROB}(P(G_g,S_{d+2})) \cdot \tfrac{1}{2} \cdot Prob(S_{d+2} \rightarrow S_{d+3}) \cdot \\
& \ldots \\
& \texttt{PROPS-PROB}(P(G_g,S_{s-2})) \cdot \tfrac{1}{2} \cdot Prob(S_{s-2} \rightarrow S_{s-1}) \cdot \\
& \texttt{PROPS-PROB}(P(G_g,S_{s-1})) \cdot \tfrac{1}{2} \cdot Prob(S_{s-1} \rightarrow S_s) \cdot \\
& \texttt{PROPS-PROB}(P(G_g,S_s)).
\end{aligned}$$

If $P(G_g, S_s)$ has a frontier-state best b-dependency path (ie. if there is no $P(G_b, S_d)$ it depends on): then b-path-prob $= Prob(S_g)$, else b-path-prob $=$ PATH-PROB$(P(G_b, S_d))$.

This composition follows from the chain rule of basic Bayesian probability theory (Pearl, 1988). It is based on two conditional independence assumptions:

1. For a given path, a state is only conditionally dependent on the state transitioning to it.

2. Only property constraints and transition constraints are enforced.

Condition 1 holds as long as the envisionment is not globally inconsistent. Condition 2 holds as long as other constraints, such as durations, do not affect which dependency paths are used.

Note the factor of $\frac{1}{2}$ for each transition probability in the computation for PATH-PROB$(P(G_g, S_s))$ shown above. This factor accounts for the fact that there are two possible direct transitions

from some $P(G_g, S_i)$ to some state S_j: one to $P(G_g, S_j)$ and another to $P(G_b, S_j)$ of the b-neighboring segment G_b. So, in lieu of domain-specific knowledge, the value of $Prob(S_i \rightarrow S_j)$ is equally distributed between these two possibilities, yielding: $Prob(P(G_b, S_i) \rightarrow P(G_g, S_j)) = \frac{1}{2} \cdot Prob(S_i \rightarrow S_j)$ and $Prob(P(G_g, S_i) \rightarrow P(G_g, S_j)) = \frac{1}{2} \cdot Prob(S_i \rightarrow S_j)$.

For example, for the pinterp-space given in Figure 5.2:

$$\texttt{PATH-PROB}(P(G_1, S_3)) = Prob(S_3) = 0.3$$

$$\begin{aligned}\texttt{PATH-PROB}(P(G_2, S_2)) &= \texttt{PATH-PROB}(P(G_1, S_3)) \cdot \tfrac{1}{2} \cdot Prob(S_3 \rightarrow S_1) \cdot \tfrac{1}{2} \cdot Prob(S_1 \rightarrow S_2) \\ &= (0.3)(\tfrac{0.8}{2})(\tfrac{0.4}{2}) = 0.024\end{aligned}$$

These compositions overestimate the probabilities for spanning-state and meeting-states paths, relative to hidden-transition paths. Over any one segment, spanning-state and meeting-states paths contribute one transition probability, whereas hidden-transition paths contribute one for each transition in that path. Such overestimation is often acceptable since it leads to simpler interpretations (i.e. ones having fewer state changes).

To more accurately determine the path-probabilities, the duration of each segment must be short enough to eliminate the need for hidden-transition paths. In that case, transitions will occur simultaneously over all chains of b-dependency paths. However, this could require increasing the number of segments by a factor of N, where N is the number of envisionment states. This could increase DATMI's time cost by a factor of N^2, since DATMI has a worst case time cost that is quadratic in the number of segments.

Another problem is that each $Prob(S_s \rightarrow S_s)$ should depend on how long S_s has been spanning. For instance, one might imagine a state-duration probability distribution indicating how the likelihood of a state persisting changes as the observed time span increases. DATMI does maintain upper and lower estimates of the time span of pinterps, as discussed in Chapter 6. However, except when these upper and lower estimates are identical or when the probability is uniform within these estimates, such probabilities could not be used by DATMI. Thus, DATMI currently does not reason about probability distributions over durations.

5.2 Normalizing Probabilistic Interpretation Credibilities

A pinterp's path-probability represents the *a priori* probability that the interpretation path leading to it is the one that occurs. Such probabilities are sufficient for determining which interpretation is best. However, these path-probabilities are actually all underestimates. This is because there will also be *a priori* probabilities implicitly associated with each global path passing through pinterps which are not ACTIVE.

So, finding the probability of a global interpretation which is conditional on the given observations requires knowing the probabilistic weight U assigned to paths passing through pinterps which are not ACTIVE. Pinterps which are not ACTIVE are either INACTIVE or INCOMPATIBLE. Multiplying global path-probabilities by the normalizing factor $F = \frac{1}{1-U}$ ensures that their total sum is 1, as desired. Significantly, this normalization process is not needed to determine the relative orders of interpretations. Thus, the orderings resulting from DATMI's pinterp-space maintenance with path-cost propagation is correct.

Nevertheless, the normalized probability for an interpretation is necessary to determine how much confidence one should have in a particular interpretation. This is especially important because finding the "second best" interpretation in DATMI can require time exponential in the

number of states (see Section 3.5). So, if the best interpretation has a normalized probability of 1/1000, then one might very well suspect that there are many alternative interpretations of similar likelihood. On the other hand, if its normalized probability is well over 0.5 then the best interpretation is clearly much better than the rest.

5.2.1 The Normalization Procedure

The amount of invalid weight U is determined by a segment-wise forward sweep across the pinterp-space, starting at the earliest segment. This sweep computes *pinterp-probabilities* for each pinterp $P(G_g, S_s)$ based on the pinterp-probabilities of the ACTIVE pinterps of G_g and b-neighbor(G_g) which can reach $P(G_g, S_s)$.

Definition 5.6 (Pinterp-probability) *The* pinterp-probability *of an* ACTIVE *pinterp gives the probability that an arbitrary interpretation goes through that pinterp. For an* INACTIVE *or* INCOMPATIBLE *pinterp, the pinterp-probability indicates the probability of that pinterp being the first non-*ACTIVE *pinterp for an arbitrary interpretation.*

Of course, any interpretation containing a non-ACTIVE pinterp is inconsistent with the observations. Thus, the sum of the pinterp-probabilities for non-ACTIVE pinterps gives the total probabilistic weight (U) assigned to inconsistent interpretations by the local path-probabilities compositions.

For each segment G_g, the sum of the pinterp-probabilities for the non-ACTIVE pinterps of b-neighbor(G_g) and the pinterp-probabilities for the ACTIVE pinterps of G_g should be at least 1. It can be greater than 1 since several pinterps can occur during G_g, for hidden-transition interpretations. In practice, however, it is often slightly less than 1 due to underestimations, as explained in Section 5.2.2.

DATMI computes the pinterp-probabilities of each pinterp of a segment G_g as follows, where G_b = b-neighbor(G_g). First, the value of the pinterp-probability of each pinterp $P(G_g, S_s)$ is initialized to 0. Then, for each ACTIVE pinterp $P(G_b, S_j)$, the product of the pinterp-probability of $P(G_b, S_j)$ and $Prob(P(G_b, S_j) \rightarrow P(G_g, S_s))$ is added to each $P(G_g, S_s)$'s pinterp-probability. Alternatively, when G_g is the earliest segment, then $Prob(S_s)$ is added to each $P(G_g, S_s)$'s pinterp-probability.

The contributions of hidden-transition paths over G_g are then added to these pinterp-probabilities. For each ACTIVE $P(G_g, S_i)$ where $i \neq g$, the product of the pinterp-probability of $P(G_g, S_i)$ and $Prob(P(G_g, S_i) \rightarrow P(G_g, S_s))$ is added to each $P(G_g, S_s)$'s pinterp-probability. After iterating over all pinterps of G_g, the contributions of longer hidden-transition paths are incrementally added in the next iteration. The increase for each $P(G_g, S_i)$ in the previous iteration is used as the pinterp-probability of each ACTIVE $P(G_g, S_i)$ when computing the additions to each $P(G_g, S_s)$'s pinterp-probability as above for the current iteration. Finally, when these iterations provide no new increases, each ACTIVE pinterp $P(G_g, S_s)$ has its pinterp-probability multiplied by PROPS-PROB$(P(G_g, S_s))$ to give their final values. DATMI never iterates more times than the number of envisionment states for any one segment, since it ignores all updates that would result from feedback through cyclic hidden-transition paths.

Figure 5.3 shows the pinterp-probabilities computed for the pinterp-space of Figure 5.2. For example, the pinterp-probability for $P(G_2, S_2)$ is:

$$\begin{aligned}\text{PINTERP-PROB}(P(G_2, S_2)) = {} & Prob(P(G_1, S_1) \rightarrow P(G_2, S_2)) \cdot \text{PINTERP-PROB}(P(G_1, S_1)) + \\ & Prob(P(G_2, S_1) \rightarrow P(G_2, S_2)) \cdot \text{PINTERP-PROB}(P(G_2, S_1))\end{aligned}$$

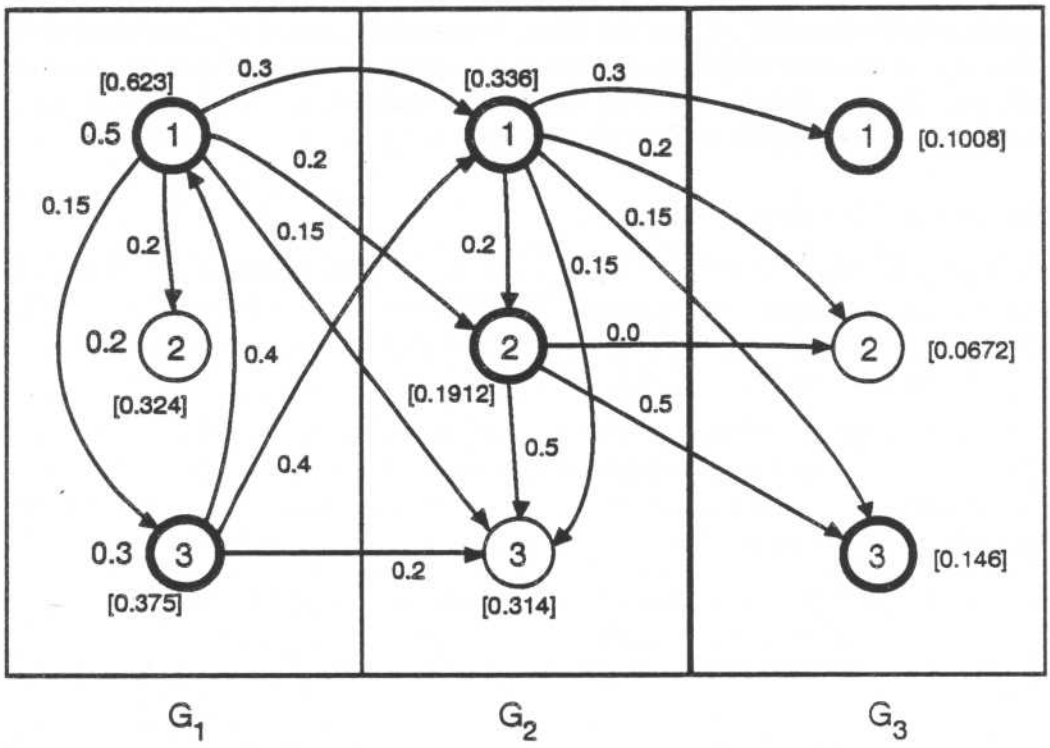


Figure 5.3: Normalizing the example pinterp-space

ACTIVE pinterps are shown in bold circles. All possible transitions from all ACTIVE pinterps to all other pinterps are shown as arrows. The label of each arrow is the conditional probability for that transition (with the probabilities for non-spanning transitions between pinterps being half of the probability for the transition between the corresponding states). The *a priori* probabilities for states in the first segment are given by the over-sized numbers next to those states. The computed pinterp-probabilities are given in brackets. The sum of the pinterp-probabilities computed for non-ACTIVE pinterps gives $U = 0.324 + 0.314 + 0.0672 = 0.7052$ ($F \approx 3.5$).

$$= (0.2)(0.62) + (0.2)(0.336) = 0.1912.$$

5.2.2 Issues in Normalizing the Pinterp-Space

A few key points should be made about this normalization procedure. First, pinterp-probabilities for all ACTIVE pinterps, representing the likelihoods that the system passes through each of those pinterps, are determined as a by-product of this procedure. These pinterp-probabilities can be used for monitoring tasks to indicate the likelihood that interesting states actually occur. Furthermore, the normalizing factor F gives a rough measure of how constraining the given observation set is. If F is very small, then the interpretation space is not being constrained very much. Unless the normalized path-probability for the best interpretation is close to 1, that would suggest the need for more observations.

This normalization procedure never places more confidence in a pinterp or interpretation path than warranted by the specified state, transition, and property probabilities. However, it slightly underestimates the pinterp-probabilities (and thus U as well) because it conservatively avoids feedback due to acyclic hidden-transition paths. When computing the contributions of hidden-transition paths to $P(G_g, S_s)$'s pinterp-probability due to a transition from a pinterp $P(G_g, S_i)$, DATMI considers the the shortest possible cyclic path of the form $S_s \rightsquigarrow S_i \rightarrow S_s$. This path is readily given by DATMI's shortest-path lookup-table; let the number of transitions in that path be k. DATMI avoids feedback to pinterp $P(G_g, S_s)$ from $P(G_g, S_i)$ by ignoring additions in $P(G_g, S_i)$'s pinterp-probability when computing increases for $P(G_g, S_s)$ after iteration $k-1$. Underestimation can also result when duration constraints cause some otherwise consistent dependency path to be pruned. When such a pruned path consists entirely of ACTIVE pinterps, the probability assigned to that path is not recovered during normalization – even though that path is inconsistent.

The complexity of this normalization procedure is within the bounds of DATMI's overall complexity (see Chapter 7). It requires at most $O(M \cdot N \cdot (2N-1) \cdot N) = O(M \cdot N^3)$ time and $O(N^2)$ space, where N is the number of envisionment states and M is the number of measurements. The factor of M accounts for there being at most M segments to sweep across during the segment-wise process. The first factor of N reflects the fact that there can be as many as N ACTIVE pinterps in a segment G_g, each propagating its pinterp-probability along all transitions from it to other pinterps. The factor of $(2N-1)$ indicates that there can be that many pinterps at the start of each of those transitions: N in b-neighbor(G_g) and $N-1$ in G_g itself. And since each segment can have as many as N iterations during each sweep, there is one more factor of N. However, to even further reduce the time costs of normalization, one could cache each pinterp-probability. One would update the cached pinterp-probability for a pinterp $P(G_g, S_s)$ during a normalization sweep only if the status or path-probabilities of some pinterp for an earlier segment, or for G_g itself, had changed since $P(G_g, S_s)$'s pinterp-probability was last updated.

Finally, consider a pinterp $P(G_g, S_s)$ which has a transition to a non-ACTIVE pinterp. It should be noted that a useful alternative to DATMI's normalization procedure is one which redistributes the probability for that transition to only those transitions which also start at $P(G_g, S_s)$. For example, consider a simple case where the envisionment has only two transitions: $S_s \rightarrow S_i$ and $S_s \rightarrow S_j$. Now, assume for G_g and its forward neighboring segment G_f that pinterps $P(G_g, S_s)$, $P(G_g, S_i)$, $P(G_g, S_j)$, $P(G_f, S_s)$, and $P(G_f, S_i)$ are ACTIVE and $P(G_f, S_j)$ is INACTIVE. This alternative redistribution would equally distribute $Prob(P(G_g, S_s) \rightarrow P(G_f, S_j))$ among the other transitions from $P(G_g, S_s)$ (i.e. $Prob(P(G_g, S_s) \rightarrow P(G_g, S_i))$, $Prob(P(G_g, S_s) \rightarrow P(G_g, S_j))$, $Prob(P(G_g, S_s) \rightarrow P(G_f, S_s))$, and $Prob(P(G_g, S_s) \rightarrow P(G_f, S_i))$).

This alternative redistribution would yield the most-probable consistent interpretation, fully conditional on the particular set of ACTIVE pinterps. Maintaining the best working interpretation with such redistribution would require updating the transition probabilities for each pinterp as pinterps cease to be ACTIVE. It seems that such a scheme could be incorporated into DATMI's pinterp-space maintenance without any change in DATMI's worst-case complexity. In contrast, DATMI uniformly normalizes all the ACTIVE pinterps of each segment to give the consistent interpretation which best agrees with the *a priori* expectations; such redistribution does not change the working global interpretation. DATMI's current preference for the *a priori* most likely interpretation which is consistent with the data seems useful because it is simpler and less sensitive to faulty data.

Chapter 6

USING DURATION CONSTRAINTS

Knowing how long things generally take can greatly constrain the interpretation space. For example, evaporation, boiling, draining, and pumping processes could all explain a decrease in the water level of a container. However, knowledge about the time that each process would take to empty a full container might eliminate some interpretations. Thus, evaporation might be ruled out if the container was observed to become empty too quickly. Likewise, if a continuous decrease in the water level takes longer than pumping all the water from a full container would take, then processes other than pumping must be occurring.

This chapter shows how DATMI reasons about such constraints by:

1. Representing estimates of duration and
2. Applying these estimates to the pinterp-space.

6.1 Representing Duration Estimates

To determine how much longer an observed change can last, one must first determine two things:

1. The exact state of the system before the change started.
2. How much change has already occurred since then.

DATMI loses such precise information by using only qualitative properties to describe pinterps. However, precise durations would be elusive even if DATMI could represent the exact states over time, since:

1. Observations often incompletely specify the states anyways.
2. Models often incompletely specify the influences on durations.

For example, one cannot determine exactly how long it would take to pump water out of a container whenever the initial water level is not precisely known or the model does not indicate the exact equation for the pumping rate.

DATMI circumvents these problems by using more-readily available *duration estimates*. It considers two types of duration estimates:

Definition 6.1 (State-duration) *The* state-duration *of state S_s is an estimate of how long the system can remain in S_s.*

Definition 6.2 (Reach-duration) *The* reach-duration *of state pair (S_s,S_d) is an estimate of the time required for the system to reach S_d, starting in S_s. This estimate holds over all acyclic state paths in the envisionment that connect S_s to S_d.*

These duration estimates are expressed as either:

1. *upper/lower-bounds* or
2. *probabilistic distributions*

Upper and lower bounds on the durations of states and paths of states suffice to eliminate many interpretations which grossly violate common sense expectations. Such interpretations might otherwise be preferred by DATMI's probabilistic or simplicity criteria. For example, imagine a state where the only change is water being drained from a container, for a system where such draining never takes more than a few minutes. An interpretation suggesting that this state lasts for hours might be the simplist interpretation consistent with the envisionment, but it would be inconsistent the upper-bound state-duration of a few minutes.

Duration probability distributions generalize these bounds by allowing different probabilities to be associated with the arbitrary ranges of durations for a state or path of states. Duration bounds can themselves be viewed as duration probabilistic distributions having zero probability for the durations outside the bounds and uniform probability for those within. Unlike upper/lower bounds, probability distributions can capture the notion that extreme durations are more unlikely than the durations near the average. Nevertheless, upper/lower-bounds can be more confidently determined and used, as the next two sections show.

A state-duration lower-bound of l seconds for state S_s is denoted $[S_s]_{L=l}$. Likewise, $[S_s]^{U=u}$ indicates a state-duration upper-bound of u seconds for state S_s. The notation $[S_s]_{L=l}^{U=u}$ compactly signifies both of these bounds. Functions $D_L(S_s) = l$ and $D_U(S_s) = u$ are also defined for each state S_s. For the reaching of state S_d from state S_s, notations $[S_s \rightsquigarrow S_d]_{L=l}$, $[S_s \rightsquigarrow S_d]^{U=u}$, and $[S_s \rightsquigarrow S_d]_{L=l}^{U=u}$ and functions $D_L(S_s \rightsquigarrow S_d) = l$ and $D_U(S_s \rightsquigarrow S_d) = u$ are likewise defined. Furthermore, duration probability distributions for state S_s and the reaching of S_d from S_s are represented by the functions $D(S_s)$ and $D(S_s \rightsquigarrow S_d)$ respectively.

6.2 Estimating Durations

Some duration estimates can be obtained directly from the envisionment. For example, for a state S_s specified as instantaneous in the envisionment, $[S_s]_{L=0}^{U=0}$ must be true. Similarly, transition $S_s \rightarrow S_d$ implies $D_U(S_s \rightsquigarrow S_d) \geq D_L(S_s)$, since the system may have just entered S_s. At the very least, each state S_s has $[S_s]_{L=0}^{U=\infty}$ and each transition $S_s \rightarrow S_d$ has $[S_s \rightsquigarrow S_d]_{L=0}^{U=\infty}$, which can be further tightened by additional constraints. Also subject to further constraints, the distributions $D(S_s)$ and $D(S_s \rightsquigarrow S_d)$ can begin as uniform distributions.

Also, some estimates can be derived from other estimates. For example, whenever the upper and lower duration bounds are equal, the probability distribution must assign all probability (1.0) to that time point. More generally, a weak approximation of $[S_s \rightsquigarrow S_d]_{L=l}^{U=u}$ can be

determined by finding two special paths between S_s and S_d. One path identifies the minimal sum (giving l) of state-duration lower-bounds and the other identifies the maximal sum (giving u) of state-duration upper-bounds. However, tighter bounds may exist for $S_s \rightsquigarrow S_d$ since these individual state-duration bounds may themselves be too weak. So, when available, global constraints on the reach-durations should be used instead of these minimal and maximal state-duration sums, to provide tighter bounds.

In any case, further constraints on the duration estimates require domain-specific knowledge not ordinarily available in an envisionment. For example, consider a state S_s where water in a container is boiling. The state-duration upper-bound for S_s cannot be more than the time that it would take to boil away all the water in a full container. Consider a state S_c from which a system transitions exactly when some conjunctive set of independent conditions are met. The state-duration upper-bound for S_c cannot be more than the maximum time that any of these conditions might remain unsatisfied. For instance, a state where water flow through a pipe is stopped by a closed valve has a state-duration upper-bound of ∞ if that valve could remain closed indefinitely. Also, if there are alternative states to which a state S_a can transition, then $D_U(S_a)$ is at most the maximum of the reach-duration upper-bounds for reaching those states from S_a. State-duration lower-bounds can be determined analogously.

6.3 Applying Duration Constraints to the Pinterp-Space

During pinterp-space maintenance, only dependency paths consistent with the duration estimates are allowed. Without any loss of soundness or completeness in the interpretation space, DATMI only applies these constraints to the b-dependency paths. This is sufficient because a pinterp cannot be ACTIVE unless it has both an f-dependency path and a b-dependency path. Duration probability distributions are handled with two distinct processes:

1. Using duration bounds – for time ranges having probabilities of 1.0.

2. Adjusting pinterp path-probabilities – over time ranges of probability less than 1.0.

Adjusting and propagating pinterp path-probabilities for duration probabilities is discussed in Section 5.1.1. Thus, the remainder of this section discusses only the use of duration bounds.

Reach-duration bounds allow verification of whether a candidate b-dependency path for $P(G_b, S_f)$ from pinterp $P(G_g, S_s)$ can occur over segment G_g. Figure 6.2 illustrates such a b-dependency path. The duration bounds for this path from $P(G_b, S_f)$ to $P(G_g, S_s)$ are given by $[S_f \rightsquigarrow S_s]_{L=l}^{U=u}$. The observed duration bounds for reaching $P(G_g, S_s)$ from $P(G_b, S_f)$ are implied by the observed duration of segment G_g and the state-duration bounds for the states in the path. Conservative upper (B_U) and lower (B_L) bounds of the observed duration of $P(G_b, S_f) \rightsquigarrow P(G_g, S_s)$ are:

$$B_U = \texttt{DURATION}(G_g) + D_U(S_s) \quad \text{and} \quad B_L = \max(0, \texttt{DURATION}(G_g) - D_U(S_f)).$$

The state-duration upper-bounds for S_f and S_s are used in these expressions of B_U and B_L to account for possible spanning-state paths involving $P(G_b, S_f)$ or $P(G_g, S_s)$. The b-dependency path from $P(G_b, S_f)$ to $P(G_g, S_s)$ is considered possible as long as these two intervals ($[l,u]$ and $[B_L,B_U]$) intersect each other.

Alternatively, state-duration bounds allow one to determine when a chain of candidate b-dependency paths imply that a state S_s is occurring for too long or too short a time. To assist

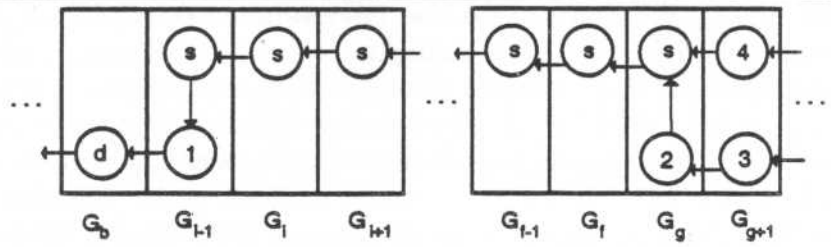


Figure 6.1: Example spanning of state S_s over many segments

State S_s fully spans segments G_i through G_g, and partially spans G_{i-1}, in the chain of b-dependency paths leading to $P(G_g, S_s)$. In this example, S_s is distinct from S_1, S_2, and S_4.

in this determination, DATMI associates a *minimum span-time* (SPAN-TIME-MIN($P(G_g, S_s)$)) and a *maximum span-time* (SPAN-TIME-MAX($P(G_g, S_s)$)) with each pinterp $P(G_g, S_s)$ as follows:

Definition 6.3 (Span-time) *The* span-time *of a pinterp $P(G_g, S_s)$ indicates the minimum and maximum time over which state S_s might be occurring continuously through $P(G_g, S_s)$ in the current best interpretation leading to $P(G_g, S_s)$.*

Figure 6.1 illustrates such a spanning sequence.

DATMI does not globally enforce all duration bounds. The polynomial-sized pinterp-space cannot represent the exponential number of possible reach-duration constraints between pinterps of different segments. Instead, reach-duration constraints are only enforced locally for each individual b-dependency path. Nevertheless, some global constraint on durations is supplied by checking the span-time of each pinterp against the state-duration bounds.

Thus, DATMI handles a subset of duration constraints whose enforcement is necessary but not sufficient. The resulting pinterp-space is complete but unsound since it can suggest interpretations which actually are not consistent with global duration constraints. This should not be surprising, since local constraint-satisfaction methods, like those of DATMI's, are inherently prone to global inconsistences. The approach taken in DATMI is to perform polynomial-time maintenance of a complete pinterp-space which is mostly sound. One can then determine, in time quadratic in the number of pinterps in the interpretation, whether a particular interpretation is indeed globally consistent with all duration estimates – by testing each pair of those pinterps against the reach-duration constraints.

6.3.1 Checking Spanning-State Dependency Paths

During pinterp-space maintenance, the minimum and maximum span-times of each pinterp $P(G_g, S_s)$ are kept up-to-date using the following definitions. The example in Figure 6.1 is referred to throughout this section.

The `base-span-time` is the sum of the durations of all the segments that state S_s fully spans in the best interpretation through $P(G_g, S_s)$. `base-span-time` clearly must include the durations of segments G_i through G_f. However, the `base-span-time` must also include the duration of G_g itself; recall that the interpretation indicated by the chain of b-dependency paths leading to $P(G_g, S_s)$ is defined to have S_s as the last state in G_g.

The span-time bounds should also account for the duration of S_s occurring at the end of the segment G_{i-1} since $P(G_{i-1}, S_s)$ is part of the best interpretation through $P(G_g, S_s)$. This partial span by S_s occurs when $P(G_{i-1}, S_s)$ has hidden-transition b-dependency path.

Let **max-others** be the sum of the maximum state-durations of the pinterps of G_{i-1} that are in the b-dependency path for $P(G_{i-1}, S_s)$, not counting $P(G_{i-1}, S_s)$ itself. Similarly, let **min-others** be the sum of the minimum state-durations of those same pinterps. S_s must last in G_{i-1} for at least as long as DURATION(G_{i-1}) - **max-others**. However, by definition, S_s must last at least as long as $D_L(S_s)$. Thus, the minimum duration of $P(G_{i-1}, S_s)$ is defined as:

$$S_L = \min(D_L(S_s), \texttt{DURATION}(G_{i-1}) - \texttt{max-others}).$$

Similarly, the maximum duration of S_s in G_{i-1} is at least max($D_U(S_s)$,DURATION(G_{i-1}) - **min-others**). However, if the b-dependency path for $P(G_{i-1}, S_s)$ includes a spanning-state paths into G_i, then this upper bound must be increased. This increase accounts for the possibility that the spanning state spent most of its time in the previous segment. Let G_b be the segment **b-neighbor**(G_{i-1}) and let $P(G_b, S_d)$ be the pinterp of G_b on which $P(G_{i-1}, S_s)$ b-depends. If the b-dependency path of $P(G_{i-1}, S_s)$ begins with $P(G_b, S_d) \rightarrow P(G_{i-1}, S_d)$, then state S_d is considered to span from G_b to G_{i-1}. Let **span-adjust** be $D_L(S_d)$ if such a spanning S_d exists, otherwise let **span-adjust** just be zero. Now, the maximum duration of $P(G_{i-1}, S_s)$ is defined as:

$$S_U = \max(D_U(S_s), \texttt{DURATION}(G_{i-1}) - \texttt{min-others} + \texttt{span-adjust}).$$

Using these bounds on the duration of $P(G_{i-1}, S_s)$ yields:

$$\texttt{SPAN-TIME-MIN}(P(G_g, S_s)) = \texttt{base-span-time} + S_L$$
and
$$\texttt{SPAN-TIME-MAX}(P(G_g, S_s)) = \texttt{base-span-time} + S_U.$$

When the span-time interval [SPAN-TIME-MIN($P(G_g, S_s)$),SPAN-TIME-MAX($P(G_g, S_s)$)] fails to intersect the state-duration interval [$D_L(S_s)$,$D_U(S_s)$], the sequence of spanning-state b-dependency paths leading to $P(G_g, S_s)$ is globally inconsistent. To fix such inconsistency, at least one of those spanning-state dependency paths might be replaced with a b-dependency path to a state other than S_s.

Currently, the DATMI implementation only attempts to replace the last spanning-state b-dependency path of a spanning sequence that exceeds the span-time upper-bound. However, one might also try replacing a spanning-state b-dependency path from the front or even the middle of the spanning sequence. For example, a fix for faulty data might suggest extending the front of a spanning sequence backwards over several earlier segments. If the extended span-time becomes greater than the maximum span-time, then one might want to replace the spanning-state b-dependency paths earlier in the sequence if retracting the fix.

When either segment G_i or G_g is a frontier segment, the value of SPAN-TIME-MIN($P(G_g, S_s)$) may be underestimated. In that case the interval [$D_L(S_s)$,$D_U(S_s)$] cannot be directly compared with [SPAN-TIME-MIN($P(G_g, S_s)$),SPAN-TIME-MAX($P(G_g, S_s)$)]. Nevertheless, it must always at least be the case that SPAN-TIME-MIN($P(G_g, S_s)$) $\leq D_U(S_s)$.

6.3.2 Checking Hidden-Transition Dependency Paths

During search for a hidden-transition b-dependency path for a pinterp $P(G_g, S_s)$, the minimum and maximum span-times of the partial hidden-transition path are incrementally updated as

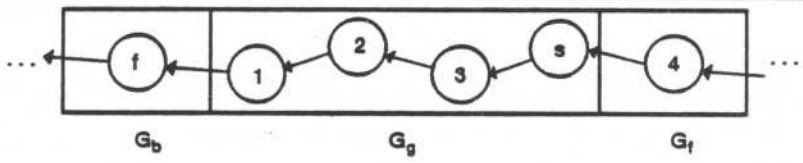


Figure 6.2: Example hidden-transition b-dependency path

The arrows indicate a hidden-transition path from $P(G_b, S_f)$ to $P(G_g, S_s)$.

the path is extended. In discussing how DATMI updates these span-times, this section refers to the example hidden-transition b-dependency path of Figure 6.2.

The maximum span-time of a hidden-transition path is the sum of the $D_U(S_i)$'s for each $P(G_g, S_i)$ in the path. Similarly, the minimum span-time of this path is basically the sum of the $D_L(S_i)$'s for each $P(G_g, S_i)$ in the path. However, to provide a true lower-bound on the span-time, $D_L(S_s)$ and $D_L(S_f)$ must sometimes be omitted from this sum. Those omissions conservatively account for cases where a state at one end of the hidden-transition path spans into a neighboring segment. Such cases occur for a hidden-transition path from $P(G_b, S_f)$ to $P(G_g, S_s)$ whenever either:

1. $P(G_g, S_f)$ is the first pinterp of G_g in the path or
2. f-neighboring $P(G_f, S_s)$ has a spanning-state b-dependency path starting at $P(G_g, S_s)$.

The state-durations for such spanning states are omitted because their durations in G_g can be insignificantly short when they spend all their time in the neighboring segment instead.

If the minimum span-time of a partial hidden-transition path exceeds the DURATION(G_g) as the path is extended during search, then further extensions of that path are avoided. Search is aborted for such paths because any completion of that path across the segment would surely take longer than the segment was observed to occur.

Alternatively, if the maximum span-time of a candidate hidden-transition path is less than DURATION(G_g), then that path cannot last long enough to account for the entire observed time of G_g. Notice that this test can miss some duration violations. For instance, if S_s spans from G_g into G_f and SPAN-TIME-MAX($P(G_f, S_s)$) is greater than DURATION(G_g), then there might still be a violation. In particular, if S_s spends most of SPAN-TIME-MAX($P(G_f, S_s)$) in G_f, then $P(G_g, S_s)$ might not be able to last long enough to allow the hidden-transition path to cross G_g. This example illustrates the inherent unsoundness of the pinterp-space when using duration estimates, as discussed in Section 6.3.

DATMI also ensures that a candidate hidden-transition path from $P(G_b, S_f)$ to $P(G_g, S_s)$ is consistent with the known reach-durations. The duration constraints on such paths that Section 6.3 presented in terms of bounds B_U and B_L can be tightened as follows. If the path actually involves S_f spanning from G_b to G_g, then these constraints must hold:

$$D_L(S_f \rightsquigarrow S_s) \leq B_U = \text{DURATION}(G_g) + D_U(S_s)$$
$$D_U(S_f \rightsquigarrow S_s) \geq B_L = \max(0, \text{DURATION}(G_g) - D_U(S_f)).$$

Otherwise, these tighter constraints must hold:

$$D_L(S_f \rightsquigarrow S_s) \leq B_U = \texttt{DURATION}(G_g) + D_U(S_s)$$
$$D_U(S_f \rightsquigarrow S_s) \geq \max(0, \texttt{DURATION}(G_g)).$$

Note that the bounds on $D_L(S_f \rightsquigarrow S_s)$ must still account for cases where S_s spans from G_g to G_f. This is because $P(G_g, S_s)$ itself cannot indicate such spans into G_f – the b-dependency paths of pinterps in G_f do that.

6.3.3 Checking Gap-Filling Dependency Paths

The gap-filling paths given by DATMI's path lookup-tables are not always consistent with the duration estimates. So, if the lookup-table gives a path violating the above tests for hidden-transition paths, a slightly modified DATMI dependency path search can find a more consistent gap-fill path.

The basic idea is to treat as ordinary segments those gap-fill segments which cannot be consistently interpreted with the paths given by the lookup-tables. Thus, they have pinterps which trivially satisfy the property constraints since there are no segment properties constraining them. As with all segment pinterps, these gap-fill segment pinterps must have dependency paths maintained for them.

Of course, since there are no segment property constraints, there are likely to be many ACTIVE pinterps for such gap-fill segments. Thus, it is more efficient to handle gap-fill segments as ordinary segments only when the conservative duration estimates are contradicted.

6.4 Problems with DATMI's Duration Reasoning

6.4.1 Incompleteness

Incompleteness can arise in the pinterp-space when using duration estimates if either:

1. Acyclic hidden-transition paths are needed to satisfy duration constraints.
2. A pinterp is considered INACTIVE whenever it is inconsistent with the duration constraints.

The polynomial-sized pinterp-space cannot represent cyclic hidden-transition paths. When ignoring duration estimates, acyclic interpretations would always be preferable and more plausible than their cyclic counterparts. However, sometimes a segment might be interpretable under duration estimates only by using repetitions of some path of pinterps. Although DATMI does not currently do so, such cases could be handled by splitting the segments until each repetition would occur as a separate hidden-transition path.

A pinterp which apparently has no b-dependency path consistent with the duration estimates cannot simply be marked as INACTIVE. For example, a spanning-state b-dependency path for a pinterp $P(G_g, S_s)$ which is inconsistent with $D_U(S_s)$ might actually be fixed by making the spanning sequence start at a later segment. It would be a mistake to mark $P(G_g, S_s)$ as INACTIVE if such a fix is possible. On the other hand, actually applying that fix could suddenly make the b-dependency path of some other pinterp conflict with the duration estimates.

To handle this dilemma, DATMI allows pinterps which violate span-time bounds to remain ACTIVE, but explicitly marks them as duration violators. A candidate interpretation containing such a violator must then be specially verified against all duration constraints. These special ACTIVE pinterps will never appear in the working interpretation since they actually have no b-dependency paths.

6.4.2 Unsoundness

As explained in Section 6.3, DATMI's use of duration estimates is inherently unsound. DATMI's limited ability to represent global context adds to this unsoundness. For example, let state S_s represent the draining of a huge tank of water and state S_f be where this tank is completely full. If the system is first in state S_f and then moves directly into S_s, S_s may last for very long time. Now, let state S_e be where this draining has reached equilibrium, with half of the water in this huge tank and the other half in some adjoining destination tank. If S_e occurs just before S_s, then maximum span-time of S_s will be significantly less than if S_f occurs just before S_s.

Such contextual effects are not limited to the state occurring just before S_s. For example, consider this interpretation: a tank drains for 9.99 minutes, a valve blocks this draining for a while, and then the tank continues to drain again for 2 more minutes. DATMI does not realize that this interpretation is inconsistent with a maximum span-time duration of 10 minutes for the state where the tank drains. Although summing the durations of such interrupted states over the chain of b-dependency paths would solve that particular example, things are not always that simple. For instance, a pump may put enough water back into the tank between the two occurrences of the draining state to make both drainings consistent.

One approach to maintaining these global contexts would a scheme where each pinterp could be represented by many alternative, annotated pinterps. The annotations would indicate special global information associated with the chain of b-dependency paths leading up to that pinterp. However, since the number of such pinterps would tend to grow exponentially as the one proceeds forward across the observational history, the number of these special pinterps would have to be limited.

Chapter 7

COMPLEXITY ANALYSIS

This chapter analyzes the DATMI algorithm to determine its time and space complexity. This analysis shows that DATMI requires time at worst cubic in the number of envisionment states and quadratic in the number of measurements. Furthermore, it requires space at worst quadratic in the number of states and linear in the number of measurements.

The worst-case time complexity is cubic in the number of states due to the worst-case complexity of hidden-transition search. However, as discussed in Section 7.5, the space complexity could be reduced to linear in the number of states by using a more concise, but less expressive, representation for dependency paths. In any case, reasons for expecting DATMI's complexity to be much lower than these upper bounds will be suggested.

7.1 Definitions

Let the number of envisionment states be N, the number of measurements be M, and the number of observed properties for any one segment be P. The concise observational history will then have at most $O(M)$ global segments since each segment must have at least one corresponding measurement and there can be at most one gap-fill segment between every non-gap-fill segment. The worst-case overall time complexity of incrementally maintaining the pinterp-space is

$$O(M^2 \cdot P \cdot N^3)$$

and the worst-case space complexity for the pinterp-space is

$$O(M \cdot (P + N^2)).$$

Although the complexities of the state lookup-table and path lookup-tables do depend on the total number of states, these factors are not as significant since these tables can be computed off-line for a given envisionment. In any case, the complexity measures given here for DATMI ignore the separate envisioning process. Envisioning itself might be performed before the interpretation task or incrementally during interpretation.

The effective N for these worst-case complexity measures is the maximum number of states actually compatible with any one segment's properties. For tasks such as process monitoring, most of the properties distinguishing the envisionment states will typically be carefully observed. Thus, the effective N will typically be a small fraction of the total number of envisionment states.

The effect of the number of observed properties on the effective value of N for DATMI complexity can be dramatic. For simplicity, assume that each of the V system variables considered in the envisionment have C possible values; then $N \leq C^V$ since there would be at most C^V distinct envisionment states. Although it is not reasonable to expect that each property assertion for a segment will reduce the number of pinterps by a factor of C, a large number Q of such assertions should result in an average of about N/C^Q COMPATIBLE pinterps per segment. The ratio of possible pinterps to COMPATIBLE pinterps would then be at most C^V/C^Q, or simply C^{V-Q}. Thus, the effective N is expected to drop exponentially in the minimum number of properties asserted for each segment.

Since gap-fill segments do not have any pinterps or properties, the effective M should be lower when there are such segments. Also, the effect P will usually be lower because segments rarely have the same number of properties.

7.2 The Size of the Interpretation Space

For an observational history of M segments, the search space contains $O(N^{N \cdot M})$ possible global interpretations. Any path which has no state occurring more than once during any one segment is a potential interpretation for DATMI. There are $O(N^N)$ acyclic paths consisting of 1 to N states. Each such path could be a local interpretation over a given segment. Although this interpretation space is exponential in N and M, DATMI finds its working global interpretation in polynomial time and space.

7.3 DATMI Space Complexity

The worst-case space complexity of $O(M \cdot P + M \cdot N^2)$ is determined by the structure of the pinterp-space. The factor of $M \cdot P$ arises because each of the M segments can have as many as P properties that must be stored in the observational history. Each segment can also have as many as N ACTIVE pinterps which each can, at worst, have acyclic dependency paths consisting of every envisionment state except that of the pinterp itself; this explains the factor of $M \cdot N^2$.

This space complexity also happens to cover the cost of the lookup-tables; the state lookup-table requires $O(P \cdot N)$ space and the path lookup-table requires $O(N^2)$. Since the original numerical measurements need not be retained after translating them into qualitative properties, no space costs other than those of the pinterp-space and lookup-tables need to be considered.

Usually the space cost of the pinterp-space will be significantly less than the worst-case, because of the factors mentioned in Section 7.1.

7.4 DATMI Time Complexity

Since maintaining a pinterp-space is a constraint-satisfaction problem (CSP), theoretical CSP analyses (Mackworth & Freuder, 1985; Mohr & Henderson, 1986; Han & Lee, 1988) provide some insight into the time complexity of DATMI itself. For example, because the constraint graph formed by the pinterp-space dependencies is a tree-structure, the DATMI constraint propagation algorithms require only time linear in the number of segments to update the pinterp-space for a new observation. Maintaining the pinterp-space involves two fundamental processes that contribute to the time complexity: determining the effects of property constraints, with a cost of

$O(M \cdot P \cdot N^2)$, and maintaining the dependency paths, with a cost of $O(M^2 \cdot P \cdot N^3)$. Together, these processes result in a worst-case time complexity of $O(M^2 \cdot P \cdot N^3)$.

7.4.1 Determining the Effects of Property Constraints

The pinterps satisfying property constraints are determined by intersecting the sets of states given by the state lookup-table for each segment property. Since each of the M segments may have up to P intersections of sets of size $O(N)$, the total time cost for determining COMPATIBLE states is at most $O(M \cdot P \cdot N^2)$.

Although this cost is reasonable, in practice the cost is typically even lower. The effective N here will be even lower than suggested in Section 7.1 because each intersection of current segment states with the states compatible with a new property results in fewer states to check with the next property.

To further reduce the cost of determining segment consistency, the DATMI implementation uses an augmented state lookup-table with a cache indicating which states are compatible with some common subsets of properties asserted for segments during the course of processing new observations. Typically, only a small fraction of the observed system variables will change values between neighboring segments. Therefore, segments with many properties will usually be able to reuse much of the state-intersection work already performed when determining the set of states compatible with earlier segments.

7.4.2 Maintaining the Dependency Paths

The total time cost of incrementally maintaining the dependency paths over the course of asserting observations is at worst $O(M^2 \cdot P \cdot N^3)$. Each update of the pinterp-space requires at most $O(M \cdot N^3)$ time since a propagation sweep can require visiting M segments and searching for the best dependency paths for a segment's $O(N)$ pinterps can require hidden-transition search of at worst $O(N^3)$ time (see Section 3.7.3 for details). By waiting to update the pinterp-space until all properties for a given segment are gathered, only $O(M)$ pinterp-space updatings are required. However, in the worst case, the pinterp-space is updated after each observation assertion; this would require as many as $M \cdot P$ updatings. Thus, the total worst-case cost of maintaining the dependency paths is $O((M \cdot N^3) \cdot (M \cdot P)) = O(M^2 \cdot P \cdot N^3)$.

7.4.2.1 Why Maintaining Dependency Paths Is Often Still Cheaper

This upper bound time complexity for maintaining dependency paths is greatly inflated, even when the factors of Section 7.1 are considered. For one, as discussed in Section 3.7.3, hidden-transition search is only as bad as $\Theta(N^3)$ in those rare cases where exhaustive graph search for least-cost paths is actually required.

Also, updates can often be postponed until all the properties for a segment have been asserted. So, since most of the observations for a segment will typically be asserted before the observations of a future segment, one seldom needs to invoke many more than M pinterp-space updatings during incremental interpretation maintenance.

And since a propagation sweep will rarely require going all the way to a frontier segment, one of the factors of M in the complexity measure is too large. In fact, that factor of M is too large simply because earlier updatings when the observational history has not yet grown to M segments could not possibly require examining M segments.

Furthermore, this upper-bound on the time complexity does not reflect the improved efficiency of using the pinterp dependencies. The efficiency of using dependencies is that only the fraction of neighboring pinterps actually depending on a changed pinterp need to have alternative dependency paths determined. Although using dependencies certainly improves the expected time complexity, there is no clear order of magnitude improvement in the worst-case time complexity.

Intuitively, one would expect the overall DATMI time complexity to be greater when there are fewer observations at each time because the interpretation space will be larger when there are less observations constraining it. Since DATMI implicitly maintains this entire interpretation space, it should be more costly to handle a smaller set of observations. Yet, the given time complexity measure of $O(M^2 \cdot P \cdot N^3)$ may seem to indicate that a smaller P leads to a lower worst-case time cost. However, in reality, changing P also changes the effective N. A lower P results in a higher effective N since more states will be compatible with a smaller, less constraining, set of properties. Thus, the intuition is correct – DATMI's performance does improve with increased observations. In any case, DATMI still performs well with sparse data due to its polynomial worst-case complexity.

7.5 Reducing DATMI Time and Space Complexities

The DATMI algorithms for finding dependency paths, as described in Section 3.7.3, are somewhat inefficient. In particular, they find dependency paths for each pinterp independently of those for other same-segment pinterps, unless exhaustive graph search gets invoked. Since a hidden-transition path H of $|H|$ pinterps also immediately indicates valid dependency paths for the $|H| - 2$ other same-segment pinterps, much of the independent search for hidden-transition paths can be redundant. For example, consider ACTIVE pinterps $P(G_g, S_1)$, $P(G_g, S_2)$, and $P(G_g, S_3)$ of segment G_g with a pinterp $P(G_n, S_4)$ ACTIVE in neighboring segment G_n. A dependency path $P(G_g, S_1) \rightarrow P(G_g, S_2) \rightarrow P(G_g, S_3) \rightarrow P(G_n, S_4)$ for pinterp $P(G_g, S_1)$ would also imply a dependency path $P(G_g, S_2) \rightarrow P(G_g, S_3) \rightarrow P(G_n, S_4)$ for $P(G_g, S_2)$ and $P(G_g, S_3) \rightarrow P(G_n, S_4)$ for $P(G_g, S_3)$.

The redundancy in the existing search algorithms could be greatly reduced by integrating the search for dependency paths for all pinterps of a segment. Such integrated search would note the smaller dependency paths discovered when longer paths were found. It would also try extending known hidden-transition dependency paths. Furthermore, all hidden-transition dependency paths could be reduced from an explicit entire pinterp path to a concise representation where only the next pinterp in the path is given. Using the transition paths of the previous example, pinterp $P(G_g, S_1)$ would have a concise dependency of just $P(G_g, S_2)$, pinterp $P(G_g, S_2)$ would in turn have a concise dependency of $P(G_g, S_3)$, and so on. The complete hidden-transition path for each pinterp would be immediately recovered by following these concise dependencies until a pinterp of the neighboring segment is reached. Using concise dependencies would reduce the space complexity of the pinterp-space by a factor of N since each of the N pinterps would then require only constant space for a dependency instead of $O(N)$ space.

Unfortunately, such concise dependencies and integrated hidden-transition search are not always desirable. As the example of Figure 7.1 shows, dependency paths cannot always be represented as concise dependencies when special constraints such as duration estimates and other global path constraints are considered. Note that while $P(G_g, S_2)$ b-depends on $P(G_g, S_1)$,

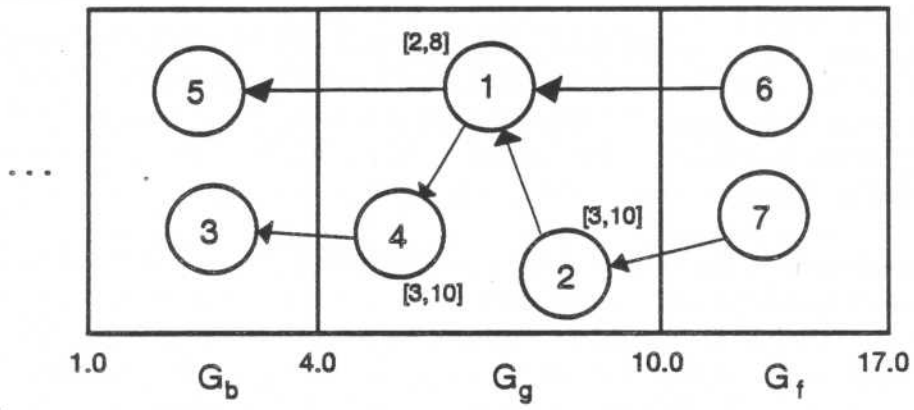


Figure 7.1: Example problem with concise dependencies

The ranges [u,l] next to some states represent state-duration upper/lower bounds (in seconds) for those states. Assume the best b-dependency path for $P(G_g, S_1)$ is $P(G_b, S_3) \rightarrow P(G_g, S_4) \rightarrow P(G_g, S_1)$ and the best one for $P(G_g, S_2)$ is $P(G_b, S_5) \rightarrow P(G_g, S_1) \rightarrow P(G_g, S_2)$. A concise dependency for $P(G_g, S_2)$ on $P(G_g, S_1)$ would indicate a dependency path of $P(G_b, S_3) \rightarrow P(G_g, S_4) \rightarrow P(G_g, S_1) \rightarrow P(G_g, S_2)$, which isn't even consistent with the state-duration lower-bounds since DURATION(G_g) is only 6.0 seconds.

$P(G_g, S_1)$ has a b-dependency path which $P(G_g, S_2)$ cannot inherit from $P(G_g, S_1)$. Doing so would violate the state-duration lower-bound constraints on the pinterps of G_g.

Such conflicts may be resolved by changing the dependency path of $P(G_g, S_1)$ to some other path of equal path-cost or path-probability. However, the current dependency path for $P(G_g, S_1)$ might be uniquely optimal, especially if the path-costs or path-probabilities are defined rather precisely. Besides, such resolution could require search time exponential in the number of ACTIVE pinterps for G_g. Nevertheless, some hybrid scheme might be useful, when storage space is tight, which allows concise dependencies except for pinterps that have special constraints preventing them from using the path which is indicated by a concise dependency.

Chapter 8

DISCUSSION

8.1 Summary

This paper has presented the DATMI framework for solving a wide variety of interpretation problems. DATMI works on interpretation tasks that meet the following two basic conditions:

- A total envisionment for the physical system can potentially be generated at the level of detail desired.
- Domain-specific knowledge is available for translating the measurements into the qualitative properties of the envisionment states.

The model can be of any ontology which satisfies these conditions.

In addition to the above conditions, DATMI suffers from two other limitations:

1. It provides interpretations as linear sequences of states, which are not always the most appropriate representations. For example, partial orderings can provide more general explanations, but DATMI's use of global segmentation and local dependency paths precludes them.
2. Its pinterp-space is complete but can be globally unsound because:
   (a) It enforces reach-duration constraints only local to each segment.
   (b) It utilizes only observations which are temporally totally-ordered; global segmentation cannot represent global trends or partial ordering constraints.

Nevertheless, DATMI offers many key contributions by:

1. Allowing conservative, probabilistic translations of numerical measurements into qualitative observations, which reduces the effects of faulty data.
2. Maintaining a concise observational history, which helps generalize interpretations.
3. Incrementally interpreting observations as they are obtained.
4. Providing key types of interpretations:
   (a) The best global interpretation, based on either costs or probabilities.

(b) A constrained space of ACTIVE pinterps, to simplify search for alternative interpretations.

(c) Estimated probabilities of each state occurring during each segment (i.e. the pinterp-probabilities found during normalization).

5. Finding hidden-transition and gap-filling paths as needed.

6. Using duration estimates to constrain the pinterp-space.

7. Detecting faulty data and testing possible fixes.

8. Requiring time and space at most polynomial in the number of measurements and envisionment states.

8.2 Related Work

8.2.1 Qualitative Physics

8.2.1.1 Q2

Integrating quantitative and qualitative observational data is clearly desirable for constraining the interpretation space. DATMI, for example, uses the durations of segments to constrain the pinterp-space. Q2 (Kuipers & Berleant, 1988) also utilizes some types of quantitative information. It propagates quantitative intervals during history generation to prune inconsistent histories.

However, using sufficiently detailed envisionments, the appropriate qualitative properties can often provide nearly as much useful constraint in DATMI as the numeric information that Q2 handles. Consider, for example, an envisionment which differentiates states by comparisons of the rates of water flowing into and out of a container. Qualitative values for this rate property can then be obtained from the original numeric measurements to constrain the interpretation space.

To ensure that it can always offer some consistent interpretation, Q2 must be able to follow every branch during history generation, which can be exponential in the number of states. Although there are also an exponential number of paths through an envisionment, DATMI never needs to consider more than a cubic number of them (for dependency path search) because of the factorization provided by its global segmentation. Thus, Q2 is less suitable than DATMI for conservative monitoring tasks.

Q2 uses a bottom-up process of extending its working histories to account for new observations. In contrast, DATMI uses a top-down process of refining the expectations of the envisionment by the new observations. Q2's explicit representation of all global interpretations allows global constraints to be fully utilized. DATMI limits the use of global constraints in return for polynomial-cost means for identifying the best interpretation and the states which can possibly occur during each segment. This trade-off generally distinguishes DATMI from other approaches as well.

8.2.1.2 ATMS

Attempts have been made to apply assumption-based truth-maintenance systems (ATMS) (de Kleer, 1986) to interpretation tasks, such as interpreting seismic events (Johnson et al., 1987). Sometimes,

an ATMS is indeed appropriate for measurement interpretation. By representing observations as ATMS assumptions, backtracking to handle incomplete data or faulty data can be avoided by using the ATMS to generate environments for each alternative interpretation.

Unfortunately, using an ATMS to maintain a space of interpretations can be exponential in the number of assumptions. For interpreting steady-state behavior of a physical system with only a few uncertain observations, as in (de Kleer & Williams, 1986), this complexity might not be prohibitive.

The time and space complexity of an ATMS seems too high for solving the across-time interpretation tasks for which is DATMI designed. The main problem is that each DATMI pinterp would correspond to an ATMS node. For an observational history of M segments and an envisionment of N states, the ATMS could require $O(2^{M \cdot N})$ time and space. This follows from the fact that there would be $M \cdot N$ pinterps, each being either ACTIVE or not depending on the activity of other pinterps. In contrast, the time complexity for DATMI is at most quadratic M and cubic in N.

8.2.1.3 GDE

Another ATMS-based approach, GDE (de Kleer & Williams, 1986), provides an alternative means for handling inconsistencies between the measurements and the model. It is not directly suited for our problem because its focus is on determining the minimal set of faults in the system itself, not in the observations. Although it acknowledges sensor failure rates, it does not attempt to reason about the nature of such failures, as DATMI does with sensor failure hypotheses. Also, GDE does not reason over time. The consequences of using TCP (Williams, 1986) with GDE to allow across-time reasoning, which de Kleer and Williams suggest as future work, are not clear. Although TCP's concise histories could represent partially-ordered interpretations, that approach would suffer from overhead that DATMI's globally-segmented pinterp-space avoids.

8.2.1.4 PREMON

The *predictive monitoring* (PREMON) framework (Doyle et al., 1988) shares DATMI's emphasis on using an explicit system model to provide expectations that can be compared with observations over time. It addresses the data selection problem of determining which sensor readings to focus on when one cannot process them all. However, PREMON does not attempt to maintain a space of consistent interpretations while it performs causal simulation. Thus, backtracking to handle faulty data would typically require complete resimulation using modified data. By not extending each working state with its many alternative next states during causal simulation, PREMON can often fail to detect anomalous behavior by being ignorant of the true state of the system.

8.2.1.5 GTD

Simmons and Davis provide another alternative framework for interpretation tasks based on the generate, test, and debug (GTD) paradigm (Simmons, 1988; Simmons & Davis, 1987). The GTD control flow divides reasoning among three components:

- Generator – Applies rules associating observation patterns with possible system behavior to obtain a hypothesis event sequence.

- Tester – Simulates the hypothesis event sequence using the system model to see if this behavior indeed results in the given observations.
- Debugger – Determines problems with the hypothesis based on causal explanations given by the tester and then either:
  1. Suggests fixes to the hypothesis sequence and tests the new hypothesis, or, alternatively,
  2. Quits trying to fix the hypothesis and instead reinvokes the generator to create a new hypothesis.

By combining the efficiency of rules for creating initial hypotheses with the robustness of causal models to ensure that hypotheses are consistent with the observations and model, GTD appears to provide a solid foundation for solving interpretation problems. However, their work does not address the problem of translating numeric sensor readings to qualitative terms nor the problem of handling faulty data. Furthermore, GTD does not incrementally generate its hypothesis interpretation, handle observations at many times, nor maintain a space of consistent alternative hypotheses.

8.2.1.6 Others

DATMI's quantity-space conversion tables are similar to mappings used in the O[M] system (Mavrovouniotis & Stephanopoulos, 1987) that maintains order of magnitude relations. For example, using O[M] notation, measurements indicating $A \sim> B$ (A is slightly greater than B) translate into $A > B$ and $A = B$ properties. However, DATMI's mappings can also have probabilities associated with them, to reflect sensor reliabilities.

Work in the closely related areas of diagnosis, process monitoring and plan recognition address the same basic problems confronting measurement interpretation. The problem of selecting a sensor failure hypothesis explaining data conflicts has been addressed in work on diagnosis. For example, one can use specifications of possible faults and their symptoms based on deep-level model expectations (Chandrasekaran & Punch III, 1987; Scarl et al., 1987). DATMI theory provides a means for incorporating such generated hypotheses into the working interpretation space. Monitoring the handling of detected sensor or system component failures by maintaining contextual information of the status of problem recovery in an augmented transition network is suggested in (Kaemmerer & Allard, 1987). Such an approach might be integrated with DATMI to manage which sensor failure and conversion failure hypotheses are currently being imposed on the interpretation space.

8.2.2 Script-Based Reasoning

Script-based reasoning is often a useful approach for interpreting the behavior of physical systems (Laskowski & Hoffman, 1987; Schaefer, 1987). The appeal of scripts is that they can provide explanations which are especially well-suited for the expected kinds of observations and behaviors. However, qualitative physics research strives to provide the deep models necessary to account for novel behaviors that were not originally considered. In that spirit, approaches based on qualitative physics models, such as DATMI, promise better coverage than script-based approaches.

8.2.3 Connectionism and Pattern Recognition

Connectionist approachs, especially parallel distributed processing (PDP), have been successful at perceptual tasks such as classification and pattern recognition with uncertain numeric data (Rumelhart et al., 1987a; Rumelhart et al., 1987b). However, interpretation across time requires the ability to reason sequentially about context, which is not necessary for other perceptual tasks. Although supporting sequential reasoning with PDP frameworks is still largely an open research problem, the existing PDP TRACE model (McClelland & Elman, 1987) has performed well on speech interpretation tasks involving single speakers uttering monosyllables. A key characteristic shared by both TRACE and DATMI is their dynamic maintenance of a space of possible interpretations consistent with the domain knowledge and currently available observations.

In order to avoid segmentations which later contextual information would show to be incorrect, TRACE makes no effort to maintain concise histories. Although concise segmentation may not be required for short speech utterances, interpreting system behavior over long periods of time could become excessively costly if the TRACE representation was used. DATMI avoids this problem by allowing segments to split as necessary when segment properties are changed to reflect faulty-data hypotheses or new observations. Furthermore, TRACE interpretations can only consist of one state (i.e. node in the connectionist network) per segment; thus, TRACE would misinterpret behaviors involving hidden-transitions due to incomplete data. However, when the observations are sufficiently complete that hidden-transitions are not required and the observation period is short enough that the non-concise segmentations are not prohibitively expensive, TRACE might be applicable. In those cases, TRACE's massively parallel relaxation methods might maintain a pinterp-space representation by assigning one pinterp to each node in its network.

Alternatively, model-based measurement interpretation can be viewed as a form of traditional structural pattern recognition. Interpretation based on an envisionment is similar to structural pattern recognition based on some grammar. The use of weightings in stochastic grammars to reflect uncertainty in the grammar or the data is analogous to DATMI's use of probabilistically weighted envisionments.

8.3 Future Work

Much further work is required to realize the full potential of the broad DATMI framework. Further developments in modelling, temporal reasoning, pinterp-space representation, data gathering, handling faulty data, dealing with uncertainty, and parallel algorithms will be needed.

8.3.1 Modelling

Advances in modelling could greatly improve measurement interpretation. For example, *order of magnitude reasoning* (Raiman, 1987; Mavrovouniotis & Stephanopoulos, 1987) is essential for integrating quantitative and qualitative knowledge as well as for handling differences in time-scale (Kuipers, 1987). Also, modelling the relative likelihoods of various physical processes (D'Ambrosio, 1987) and conditioning the plausibilities of conjunctive behaviors on these likelihoods would provide additional interpretation preference criteria. Modelling systems at several levels of abstraction (Falkenhainer & Forbus, 1988) would allow interpretations which

only make significant distinctions among properties. Such abstractions can also make the total envisioning process more tractable.

DATMI could perform system identification tasks by combining total envisionments of several systems into one *composite envisionment*. Since DATMI's time and space complexity is at worst only cubic in the number of envisionment states, the larger number of states in such composite envisionments need not be prohibitive.

8.3.1.1 Incremental Envisioning

Totally envisioning a physical system can lead to a number of states and transitions which is exponential in the number of system variables. To ensure tractable temporal reasoning during interpretation, the DATMI implementation currently assumes that an existing total envisionment is available before interpretation begins. We suspect that such clean separation of envisioning from interpretation is more efficient for cases where a large fraction of envisionment states are likely to occur or where the observations are very sparse. This intuition is based on the efficient techniques developed for total envisioning, such as those described in (Forbus, 1990).

The other cases would perhaps be best handled by using some kind of *incremental envisioning* during the interpretation process. Such incremental envisioning would augment a working, partial envisionment with additional states and transitions as needed. New states and transitions would be requested at least whenever DATMI detected an inconsistent segment. These new states and transitions could then be incorporated into the pinterp-space using the adjustment operations that DATMI currently uses for recovering from faulty data. We are currently exploring this direction.

8.3.1.2 Concise Envisionments

A *concise envisionment* could be formed by partitioning an envisionment into sub-envisionments which each indicate the possible behaviors of non-interacting sets of system components. Irrelevant temporal orderings of the behaviors of non-interacting system components would not be represented in a concise envisionment. Such envisionments would have many fewer states and transitions than the total envisionment representing the same behaviors. As a first start, a system modelled in QP theory could have each sub-envisionment indicate the behaviors of one *p-component* (Forbus, 1984).

Such concise envisionments would provide smaller envisionments, reducing DATMI's complexity. However, handling those rare states where such components could actually interact would require extending the DATMI framework to allow some pinterps to represent those *interaction states*. Pinterps representing interaction states would require multiple dependency paths which simultaneously reach the various partial states that represent the behaviors of non-interacting sets of components during a neighboring segment. Thus, each global interpretation might involve some simultaneous paths of partial states across some of the segments.

8.3.2 Temporal Reasoning

Partial temporal orderings among DATMI property assertions could further constrain the interpretation space. By supporting the full range of temporal relations defined in Allen's temporal logic (Allen, 1983), even a very incomplete observation could greatly constrain the working interpretation space. For example, noting that a property p_1 occurs some unspecified time after

some p_2, one could eliminate all interpretations where p_2 actually occurs before p_1. Unfortunately, only one instantiation of these temporal orderings can be represented by DATMI since it globally segments the observations.

An alternative to global segmentation is provided by Williams' temporal constraint propagator (TCP) (Williams, 1986), which allows such partial temporal-ordering relations. However, it is unclear how a pinterp-space could be maintained with TCP. It seems that each alternative global segmentation of TCP's concise history would have to be represented to allow pinterps to be maintained. Only a subset of this intractably large set of alternative global segmentations could be considered. Without explicitly reasoning about all these alternative global segmentations, the interpretation space may fail to cover the actual behavior or even become inconsistent due to faulty data. As Dean and Boddy (Dean & Boddy, 1987) point out, determining whether two actions are independent may, in fact, require considering all total orderings to ensure there are no significant interactions.

Much of the constraint imposed by observations is local to each segment since the properties of each segment locally determine which pinterps are COMPATIBLE. Yet, for most real interpretation tasks, some of the available observational constraints will not be local. For example, one may note that there is a change in some system variable over some interval of time without being able to determine exactly when the change occurs. Call such a global constraint an *occurs-within* constraint. These constraints are especially useful for noting that a change occurred between two measurement points that are not close enough to confidently make any property assertion.

Occurs-within constraints could be expressed using weak temporal orderings. One could represent them in a pinterp-space by selecting one ordering and allowing backtracking. Unfortunately, the resulting pinterp-space would no longer cover all consistent interpretations. Alternatively, if there were only a small number of occurs-within constraints, a candidate global interpretation could be tested against them.

One may also wish to forbid certain paths of envisionment states from the interpretations. A need for forbidding such paths may result from a problem with envisionments first identified by Kuipers (Kuipers, 1986): some paths through an envisionment may not be globally consistent with the underlying physics. One means of handling such global constraints would be to use techniques such as *logic of occurrence* (Forbus, 1986b) to propagate global information such as changes in energy. Furthermore, the envisioning process itself might be able to identify particular paths of states which are found to be globally inconsistent but which cannot be explicitly denoted as so by the envisionment's state-transition graph representation. As with other such global constraints, DATMI could then test candidate global interpretations against them.

8.3.3 Compact Pinterp-Spaces

As discussed in Chapter 2, a *globally-segmented concise history* is maintained by merging segments which have identical segment properties and similar confidence levels for those properties. One might consider also merging any two neighboring segments whose sets of ACTIVE pinterps correspond to the same states, since those segments differ only in seemly insignificant property values. The segment properties for the merged segment would be the intersection of the properties of those two segments. For example, conservative translation of measurements may have given segment G_g a disjunctive property value of INCREASING $\vee$ STEADY for its property named

p while a neighboring segment G_n asserts only the value INCREASING for its property named p. If none of the ACTIVE pinterps of G_g would become INCOMPATIBLE if the value of STEADY was not asserted for G_g's property named p, then that value of STEADY could be discarded as needlessly conservative. The above condition holds if the ACTIVE pinterps of segments G_g and G_n correspond to the same states.

Although the space of valid interpretations would not change, merging segments G_g and G_n if they had corresponding sets of ACTIVE pinterps would better summarize the possible behaviors during those times. Of course, the risk of this method is that faulty data can mislead one into believing property values are unnecessary when in fact they are not. However, conservative translations of measurements into segment property assertions have been empirically noted to result in many neighboring segments that differ in unnecessary property values. Thus, such mergers might be required in practice to avoid very fragmented global segmentations when conservatively interpreting many measurements for many system variables.

Further reduction in the size of the pinterp-space can be obtained by discarding long-past segments. Indeed, if one is monitoring a system over a period of weeks with rapid sampling, it would be of little utility, and perhaps even infeasible, to maintain a full pinterp-space over the entire observation period – unless the measured system variables are slow to change qualitatively. At the very least, one might summarize very old segments by discarding seemly insignificant property values for segments that have not had any changes in their pinterp dependencies after observing many segments after them. Any effects on those old segments due to faulty data would probably have been detected, if ever, by the time that so many later segments have been observed.

If only the best global interpretation is being sought, then the pinterp-space can also be compressed by discarding alternative ACTIVE pinterps for all segments before a *convergence segment*. Each convergence segment has either zero or one ACTIVE pinterps – zero if it is also an inconsistent segment. Status changes for pinterps of segments on one side of a convergence segment G_C cannot cause status changes for pinterps of segments on the other side. If no new property assertions are going to be made over time periods before G_C, then the best global interpretation up to G_C cannot change. Thus, one could discard all ACTIVE pinterps in the earlier segments not contributing to that best interpretation.

This method might require recovering some of those discarded pinterps when the data observed so far was actually faulty. For example, G_C may get alternative ACTIVE pinterps due to the application of some fix-hypothesis. Then the previously sole ACTIVE pinterp for G_C may be filtered due to some later segment property assertions. In such a scenario, some discarded ACTIVE pinterp of a segment G_P preceding G_C may be required to provide dependencies for some pinterp of G_C.

As this section indicates, future work towards a more compact pinterp-space must balance the advantages of compression with the difficulties of recovering discarded information to handle faulty data.

8.3.4 Active Data Acquisition And Data Selection

Active data acquisition is crucial for many important interpretation tasks where the number of potential observations is great but the number which can be made at any one time is small. For example, an airplane pilot must constantly decide in real-time which of the many gauges to read to adequately determine the status of the plane. Because dynamic data which are not

acquired are typically lost, deciding which data to gather at any particular time is of great urgency. Alternatively, *data selection* involves selecting among recorded observations to find those which best constrain the working interpretation space.

Data selection is easier than data acquisition because the space of possible data is much more restricted and there is no need to decide how to actively search for new data. Nevertheless, data acquisition and data selection are both difficult problems. In the DATMI framework, these problems might be addressed by trying to reduce the number of ACTIVE pinterps for each segment. Segments with many ACTIVE pinterps indicate times over which the behavior is most ambiguous. For example, one might prefer observations at times nearest the segments having the most ACTIVE pinterps, since those observations are most likely to affect those segments. In any case, much future work is required to provide the data selection skills of a human nuclear powerplant operator or the active data acquisition skills of an experimental scientist.

8.3.5 Handling Faulty Data

A blackboard architecture (Hayes-Roth, 1985) would make the generation and selection of DATMI fix-hypotheses more robust and adjustable. The original numeric measurements, translated properties, and the pinterp-space itself would be the main data structures upon which the many knowledge sources could operate. Knowledge sources specializing in translating from numeric to qualitative properties and assigning confidence levels to these translations could be used. Knowledge sources for each fix-hypothesis class could help determine which of the competing hypotheses to try next.

Also, instead of simply forgetting properties when trying to fix an inconsistent pinterp-space, one could try to actually reevaluate the original numeric data after changing some parameters. For example, one could retranslate some property ρ using different sample time parameters MIN-ST(ρ) and MAX-ST(ρ). Alternatively, one could smooth the data signals with a different window size to change the sensitivity to the original fluctuations in the data. Indeed, one might even determine the entire qualitative space of smoothed signals (Witkin, 1983) and then select one that provides a consistent interpretation. However, the computational cost of computing the entire scale-space or even selecting a scale that leads to a consistent pinterp-space could easily become prohibitive.

8.3.6 Reasoning Under Uncertainty

DATMI provides some means for incorporating measures of certainty into interpretations to reflect confidences in observational data and *a priori* likelihoods of particular behaviors. However, the integration of the wide-variety of alternative methods for dealing with uncertain information into the DATMI framework warrants further exploration. For example, qualitative measures of uncertainty (such as endorsements (Cohen, 1985), partially ordered certainties (Rosen-Krantz, 1981), and modal logics of likelihood (Halpern & Rabin, 1987)) might allow even weak knowledge about the relative certainties in observations and state and transition likelihoods to be used to further bias the pinterp-space. Path-probability intervals based on the concept of certainty intervals (Shafer, 1985) might allow imprecision and ignorance about the certainties in observations and state and transition likelihoods to be reflected in the path-probabilities. These path-probability intervals might be tightened as additional information focuses the range of path-probabilities most consistent with all available information.

8.3.7 Parallel Algorithms

Although DATMI is currently implemented as a sequential LISP program, parallel algorithms for several key processes could greatly increase the overall efficiency of DATMI. Naturally, the initial data smoothing and segmenting steps could be done in parallel. Since the DATMI propagation procedures proceed in a segment-wise fashion, dependency paths for each pinterp of the current segment could be sought by individual processors simultaneously. With M segments, P properties in any segment, and N envisionment states, N processors could reduce the time complexity of DATMI propagation from $O(M^2 \cdot P \cdot N^3)$ to $O(M^2 \cdot P \cdot N^2)$. Furthermore, the activation propagation sweeps for each seed-segment during pinterp-space adjustment could each be performed simultaneously.

Parallel algorithms could also improve the efficiency of generating and using the DATMI lookup-tables. The state lookup-table could be generated for each state independently, in time linear in P and constant in N, using N processors. The shortest-path lookup-table could also be generated for each initial state independently, in linear time in N (or in quadratic time for the least-cost-path lookup-table), again by using N processors. Accessing the state lookup-table to gather the states compatible with P properties using P processors would cost $O(N^2 \cdot lg(P))$ time versus the current $O(N^2 \cdot P)$ cost since parallel intersections of sets of states could be performed.

8.4 Conclusions

DATMI provides a modular framework for interpreting the behavior of physical systems. Since it separates the interpretation process into distinct modules for modelling, translation, segmentation, pinterp-space maintenance, and fault recovery, further research can focus on these areas individually. For example, advances in qualitative modelling would provide better envisionments without requiring the other modules to change. Likewise, more efficient, perhaps parallel, hidden-transition path-finding algorithms would only affect the pinterp-space maintenance stage.

By viewing the interpretation task as a process of incrementally constraining an interpretation space, all interpretations currently consistent with the given observations and system model are available. Thus, DATMI is well-suited for process monitoring tasks where it is important to detect the possibility of undesirable behaviors occurring. Also, having immediate access to alternative interpretations, via the dependency paths, reduces the amount of backtracking required to handle incomplete or faulty observations.

REFERENCES

Allen, J. F. Maintaining knowledge about temporal intervals. *Communications of the ACM*, **26**(11):832–843, November 1983.

Chandrasekaran, B and Punch III, W. F. Data validation during diagnosis, a step beyond traditional sensor validation. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 778–782, July 1987.

Cohen, P. R. *Heuristic Reasoning about Uncertainty: An Artificial Intelligence Approach.* Pitman Publishing Inc., 1985.

Collins, J. W and Forbus, K. D. *Building Qualitative Models of Thermodynamic Processes.* Technical Report, University of Illinois at Urbana-Champaign, 1990. (In preparation).

D'Ambrosio, B. Extending the mathematics in qualitative process theory. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 595–599, July 1987.

Dean, T and Boddy, M. Incremental causal reasoning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 196–201, July 1987.

de Kleer, J and Williams, B. Reasoning about multiple faults. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 132–139, August 1986.

de Kleer, J. An assumption-based TMS. *Artificial Intelligence*, **28**(2), March 1986.

Doyle, J and Sacks, E. P. Stochastic analysis of qualitative dynamics. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1187–1192, August 1989.

Doyle, J. A truth maintenance system. *Artificial Intelligence*, **12**:231–272, 1979.

Doyle, R. J, Sellers, S. M, and Atkinson, D. J. Predictive monitoring based on causal simulation. 1988. Presents research done at JPL on automated monitoring of physical systems.

Falkenhainer, B and Forbus, K. D. Setting up large-scale qualitative models. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 301–306, August 1988.

Falkenhainer, B. *Learning from Physical Analogies: A Study in Analogy and the Explanation Process.* PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, December 1988. (Technical Report UIUCDCS-R-88-1479, University of Illinois at Urbana-Champaign, December 1988).

Forbus, K. D. *A Study of Qualitative and Geometric Knowledge in Reasoning about Motion.* Technical Report TR-615, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA., 1981.

Forbus, K. D. Qualitative process theory. *Artificial Intelligence*, **24**:85–168, 1984.

Forbus, K. D. Interpreting measurements of physical systems. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 113–117, August 1986. (Technical Report UIUCDCS-R-86-1248, University of Illinois at Urbana-Champaign, March 1986).

Forbus, K. D. *The Logic of Occurrence.* Technical Report UIUCDCS-R-86-1300, University of Illinois at Urbana-Champaign, December 1986.

Forbus, K. D. The logic of occurrence. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, August 1987. (Technical Report UIUCDCS-R-86-1300, University of Illinois at Urbana-Champaign, December 1986).

Forbus, K. D. Qualitative physics: past, present, and future. In *Exploring Artificial Intelligence*, Morgan Kaufmann, 1988.

Forbus, K. D. Introducing actions into qualitative simulation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1273–1278, August 1989. (Technical Report UIUCDCS-R-88-1452, University of Illinois at Urbana-Champaign, August 1988).

Forbus, K. D. The qualitative process engine. In D. S Weld and J de Kleer (Eds.), *Readings in Qualitative Reasoning about Physical Systems*, pages 220–235, Morgan Kaufmann, 1990. (Technical Report UIUCDCS-R-86-1288, University of Illinois at Urbana-Champaign, December 1986).

Halpern, J and Rabin, M. A logic to reason about likelihood. *Artificial Intelligence*, (32):379–405, 1987.

Han, C and Lee, C. Comments on Mohr and Henderson's path consistency algorithm. *Artificial Intelligence*, **36**:125–130, 1988.

Hayes, P. J. The second naive physics manifesto. In J Hobbs and R Moore (Eds.), *Formal Theories of the Commonsense World*, Ablex, 1985.

Hayes-Roth, B. A blackboard architecture for control. *Artificial Intelligence*, **26**:251–321, 1985.

Johnson, R. R et al. Interpeting signals with an assumption-based truth maintenance system. In *SPIE Vol. 786 Applications of Artificial Intelligence V*, pages 332–337, 1987.

Kaemmerer, W. F and Allard, J. R. An automated reasoning techique for providing moment-by-moment advice concerning the operation of a process. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 809–813, July 1987.

Kuipers, B and Berleant, D. Using incomplete quantitative knowledge in qualitative reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 324–329, August 1988.

Kuipers, B. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

Kuipers, B. Abstraction by time-scale in qualitative simulation. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 621–625, July 1987.

Laskowski, S. J and Hoffman, E. J. Script-based reasoning for situation monitoring. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 819–823, July 1987.

Mackworth, A. K and Freuder, E. C. The complexity of some polynomial newtwork consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25:65–73, 1985.

Mavrovouniotis, M and Stephanopoulos, G. Reasoning with orders of magnitude and approximate relations. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 626–630, July 1987.

McClelland, J. L and Elman, J. L. Interactive processes in speech perception: the TRACE model. In J. L McClelland et al. (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2*, chapter 15, pages 58–121, MIT Press, Cambridge, Massachusetts, 1987.

Mehlhorn, K. *Graph Algorithms and NP-Completeness.* Volume 2 of *Data Structures and Algorithms*, Springer-Verlag, 1984.

Mohr, R and Henderson, T. C. Arc and path consistency revisited. *Artificial Intelligence*, 28:225–233, 1986.

Pearl, J. *Probabilistic Reasoning In Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

Raiman, O. Order of magnitude reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 100–104, August 1987.

Rosen-Krantz, P. *Foundations and Applications of Inductive Probability*, chapter 8. Ridgeview Publishing Co., 1981.

Rumelhart, D. E, McClelland, J. L, et al. *Foundations.* Volume 1 of *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, The MIT Press, 1987.

Rumelhart, D. E, McClelland, J. L, et al. *Psychological and Biological Models.* Volume 2 of *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, The MIT Press, 1987.

Scarl, E. A, Jamieson, J. R, and Delaune, C. I. Diagnosis and sensor validation through knowledge of structure and function. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):360–368, May/June 1987.

Schaefer, P. Belief functions for real-time script processing. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 502–508, August 1987.

Shafer, G. Probability judgement in artificial intelligence. In *Workshop on Uncertainty and Probability in Artificial Intelligence*, pages 127–135, August 1985.

Simmons, R and Davis, R. Generate, test and debug: combining associational rules and causal models. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1071–1078, August 1987.

Simmons, R. *Combining Associational and Causal Reasoning to Solve Interpretation and Planning Problems.* PhD thesis, Massachusetts Institute of Technology, Artificial Intelligence Lab, Cambridge, Cambridge, Massachusetts, September 1988.

Waltz, D. L. *Generating Semantic Descriptions from Drawings of Scenes with Shadows.* Technical Report MAC AI-TR-271, Massachusetts Institute of Technology, Artificial Intelligence Lab, Cambridge, 1972.

Williams, B. C. Doing time: putting qualitative reasoning on firmer ground. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 105–112, August 1986.

Witkin, A. Scale-space filtering. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 1019–1022, 1983.

APPENDIX A

THE QPE PUMP-CYCLE TOTAL ENVISIONMENT

This appendix describes the pump-cycle model used for the DATMI examples shown in Appendix B. As shown in Figure A.1, this pump-cycle consists of two identical containers of water connected by a pump and a valved-pipe. When the pump is ON and the valve is OPEN, water flows through the pump from CAN1 to CAN2 and through the pipe from CAN2 back to CAN1, forming a cycle.

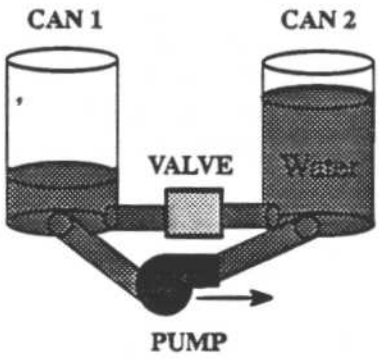


Figure A.1: The pump-cycle scenario

This pump-cycle system was modelled with Collins' and Forbus' thermodynamic domain QP models (Collins & Forbus, 1990). The total envisionment for DATMI was given by QPE (Forbus, 1990) using these domain models and a description of the pump-cycle scenario. Augmented envisioning techniques (Forbus, 1989) provided the transitions between states which differ in the statuses of the pump and valve. Thus, changes in the pump and valve were modelled as the results of actions by external agents and other external causes, such as component failures.

The table of Figure A.2 indicates the values of each interesting property for each of the 42 states of the total envisionment. Each column of this table represents the particular state indicated by the number given at the top of that column. Likewise, each row indicates the

value for the property of the name given at the left-hand side of that row. For example, this table shows that in state S_1 the system consists of two empty containers, the pump is OFF, and the valve is CLOSED.

```
Q1 = (AMOUNT-OF-IN WATER LIQUID CAN1)        Q5 = (FLUID-LEVEL CAN1)
Q2 = (AMOUNT-OF-IN WATER LIQUID CAN2)        Q6 = (FLUID-LEVEL CAN2)
Q3 = (FLOW-RATE PATH)                        Q7 = (MAX-PRESSURE PUMP)
Q4 = (FLOW-RATE PUMP)                        Q8 = (PRESSURE PUMP)

Property/state table:
                                                                1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 4 4 4
                                1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
P1 = ORDER(Q1,0) ..             = = = = = = = = = = = = = = = = = > > > > > > > > > > > > > > > > > > > > > > > > > > >
P2 = ORDER(Q2,0) ..             = = = = > > > > > > > > > > > > > = = = = > > > > > > > > > > > > > > > > > > > > > > >
P3 = ORDER(Q5,Q6) .             = = = = < < < < < < < < < < < < < > > > > < < < < < < < < < < < < < = = = = > > > >
P4 = (ON PUMP) ....             F F T T F F F F F F T T T T T T T F F T T F F F F F F T T T T T T T T F F T T F F T T
P5 = (OPEN VALVE) .             F T F T F F F F T T T F F F T T T F T F T F F F F T T T F F F T T T T T F T F T F T F T
P6 = ORDER(Q7,Q8) .             > > > > > < = > < = > < = > < = > > > > > > < = > < = > < = > < = > > > > > > > > > > >
P7 = CHANGE(Q5) ...             0 0 0 0 0 0 0 + + + + 0 0 + + + 0 - - - 0 0 0 + + + + 0 - + + - 0 + 0 0 - - 0 - - -
P8 = CHANGE(Q6) ...             0 0 0 0 0 0 0 - - - - 0 0 - - - 0 + + + 0 0 0 - - - - 0 + - - + 0 - 0 0 + + 0 + + +
P9 = CHANGE(Q4) ...             0 0 0 0 0 0 0 0 0 0 0 + 0 0 + + 0 0 0 0 - - 0 0 0 0 0 0 + 0 - + + - 0 + 0 0 - - 0 0 - -
P10 = CHANGE(Q3) ...            0 0 0 0 0 0 0 - - - 0 0 0 - - - 0 + 0 + 0 0 0 - - - 0 0 0 - - + 0 - 0 0 0 + 0 + 0 +
```

Figure A.2: Summary of state properties for the pump-cycle system

In addition to defining the state properties, the total envisionment also indicates the valid transitions among these states. The graph of Figure A.3 shows all of these state transitions. For example, if the system is in state S_1 then it can transition into state S_2, where the two containers are still empty but the valve is now OPEN instead of CLOSED.

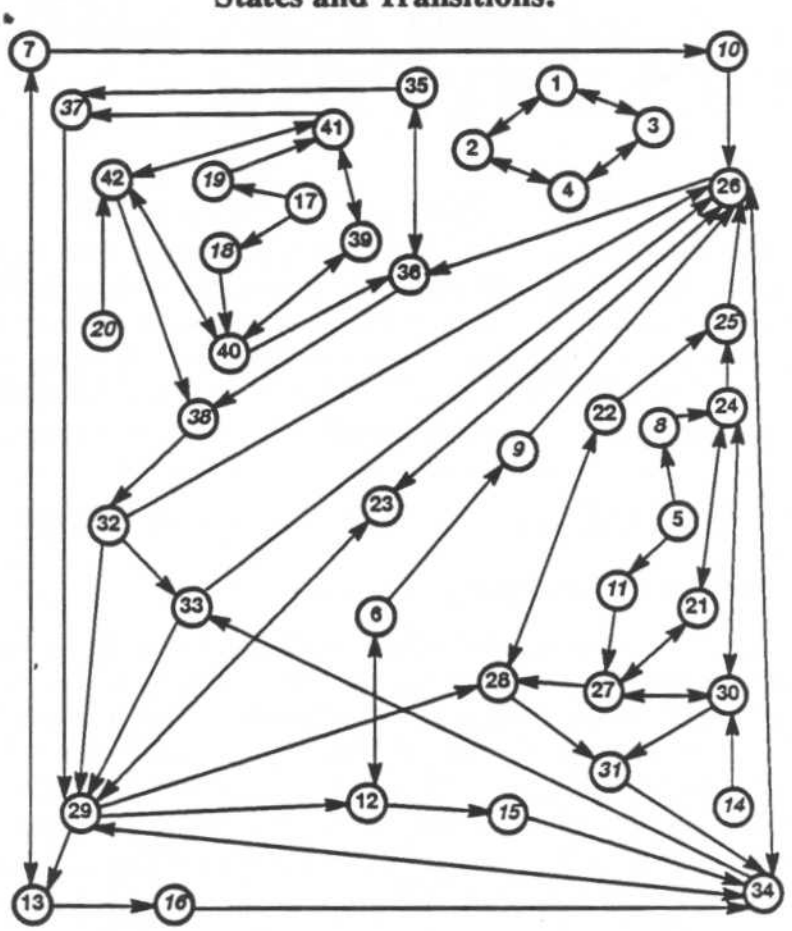


Figure A.3: State-transition diagram for the pump-cycle system

States 8, 9, 10, 11, 14, 15, 16, 18, 19, 20, 25, 31, 37, and 38 are instantaneous states.

APPENDIX B

DATMI EXAMPLES FOR THE QP PUMP-CYCLE SYSTEM

For these examples, DATMI used the total envisionment described in Appendix A. The measurements for each example were obtained from numerical simulations of the pump-cycle. For each example, this appendix describes the status of the pinterp-space at various key points in the dynamic process of interpreting the stream of measurements. Although DATMI has been used for many interpretation tasks, including the PHINEAS (Falkenhainer, 1988) project, a representative set of examples is provided by this appendix along with Appendix D.

B.1 Handling Sensor Failures with Property Adjustments

B.1.1 Example 1: Failure of the Pump Indicator

This example illustrates how DATMI handles faulty data which arise from sensor failures. An inconsistency is detected when water flow through the path is observed even though the water levels seem equal and the pump is observed to be OFF. The envisionment indicates that states where the water level of CAN1 is not greater than the water level of CAN2 cannot have water flowing from CAN1 to CAN2 unless the pump is ON. As will be shown, DATMI resolves this inconsistency by doubting observations of the PUMP status. Figure B.1 illustrates the portion of the envisionment that provides the best working interpretations over the course of this example.

The first *snapshot*, Snapshot 1-1, of the pinterp-space shows that an inconsistency is detected when the pump is observed to be OFF (for the property named $P4$) during segment Seg32 (G_{32}). Each row of these snapshots indicates the segment properties and pinterp statuses of the particular segment indicated at the left-hand side. INCOMPATIBLE pinterps are indicated by a ".", ACTIVE pinterps by a "#" or a digit, and INACTIVE pinterps by a "-" or "=".

The property name abbreviations $P3$, $P4$, $P7$, $P8$, and $P10$ used in these snapshots are defined in Figure A.2. For example, all the pinterps for segment G_1 (Seg1) are INCOMPATIBLE except for ACTIVE pinterps $P(G_1, S_{17})$ and $P(G_1, S_{39})$. Also, the observed values for each property of G_1 are:

- $P3$: "The water level is greater in CAN1 than in CAN2."
- $P4$: "The pump is OFF."

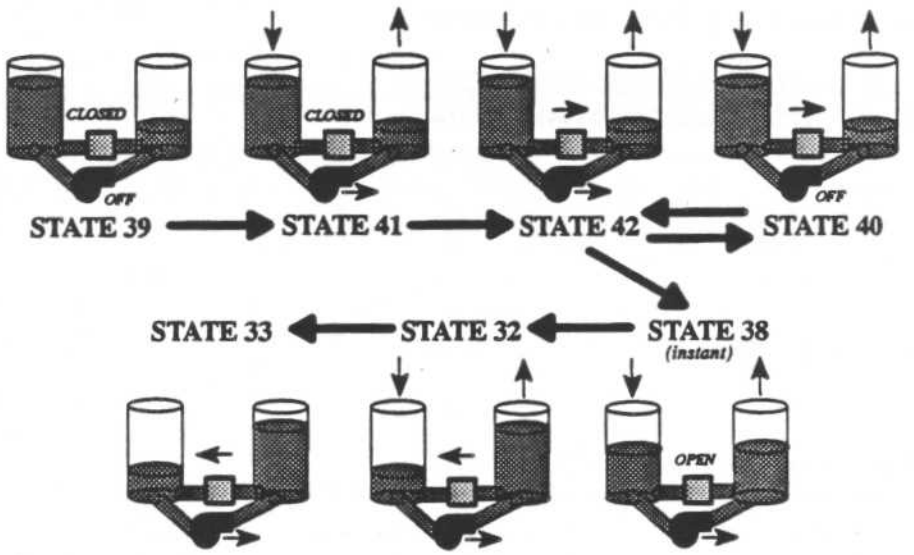


Figure B.1: Portion of the pump-cycle envisionment

(Arrows indicate direction of change in the water levels and the water flow)

- *P*7: "The water level in CAN1 is steady."
- *P*8: "The water level in CAN2 is steady."
- *P*10: "There is no change in the rate of water flow through the pipe."

During G_{32}, the properties for *P*7, *P*8, and *P*10 indicate that water is moving from CAN1 to CAN2. However, since *P*3 suggests that the water level of CAN1 is not greater than that of CAN2, it is impossible for such changes in the water levels to occur in this system without the pump being ON. Thus, G_{32} has no COMPATIBLE pinterps, which means it has no ACTIVE pinterps. So, G_{32} is an inconsistent segment that must be fixed.

Snapshot 1-1 also illustrates a few other aspects of DATMI. First, the property for *P*3 has the disjunctive value of "less-than or equal-to" over G_{32} because the numerical measurements of the water level in CAN1 were just slightly less than the measurements of the water level in CAN2. DATMI's quantity-space conversion table allowed imprecise sensor readings to be conservatively interpreted. Similarly, no value for *P*3 during G_{30} is asserted because the measurements for G_{30} did not satisfy the noise window (recall Section 2.3) for *P*3. This occurred because the change from ">" to "< or =" for *P*3 occurred at some indeterminate point during G_{30}.

Also, the "=" for $P(G_{15}, S_{19})$ indicates that S_{19} cannot occur during G_{15} because of conflicts with duration estimates. In particular, $P(G_{15}, S_{19})$ could only be ACTIVE if S_{19} spanned from G_{11} to G_{15}. No other ACTIVE pinterp of G_{11} other than $P(G_{11}, S_{19})$ could reach $P(G_{15}, S_{19})$ using state transitions through ACTIVE pinterps of G_{11} and G_{15}. However, S_{19} cannot span because S_{19} is an instantaneous state with $D_U(S_{19}) = 0$. Note that $P(G_{11}, S_{19})$ itself does not similarly conflict with the duration estimates because it can b-depend on $P(G_{10}, S_{17})$ and f-depend on $P(G_{11}, S_{41})$ and $P(G_{15}, S_{41})$.

Snapshot 1-1: *INCONSISTENT*

```
***** Inconsistent segment <Seg 32> detected! *****

                  11111111112222222222333333333344
        1234567890123456789012345678901234567890123456789012    P3    P4   P7   P8   P10
  Seg1: ................#.....................1...              >     F    0    0    0
  Seg9: ................#.....................1...              >          0    0    0
 Seg10: ................#.#...................1.#.              >                    0
 Seg11: ..................#.....................1.              >     T    -    +    0
 Seg15: ..................=-....................#1              >     T    -    +
 Seg18: ..................-.....................1              >     T    -    +    +
 Seg22: ................-.-....................#.1              >          -    +    +
 Seg24: ................-.....................1..              >     F    -    +    +
 Seg30: ................-.....................1..                   F    -    +    +
 Seg32: ..........................................           (< =)    F    -    +    +
 Seg33: ??????????????????????????????????????????????????                 -
```

To fix an inconsistent segment, DATMI tries to forget all the segment properties of some name that have the most recent value. As Snapshot 1-2 shows, forgetting $P10$ for its most recent value of "+" does not remove the inconsistency, though it does make some more pinterps COMPATIBLE, like $P(G_{18}, S_{19})$ and $P(G_{22}, S_{41})$. Note that no pinterps COMPATIBLE in Snapshot 1-1 are now INCOMPATIBLE in Snapshot 1-2. This a useful conservative feature of forgetting segment properties instead of actually changing their values.

Snapshot 1-2: *INCONSISTENT*

```
Try SENSOR-FAILURE FIX-HYPOTHESIS:
  Forget P10 from <Seg 15> to <Seg 32> ...
  ***** Pinterp-space still inconsistent at <Seg 32>! *****

                  11111111112222222222333333333344
        1234567890123456789012345678901234567890123456789012    P3    P4   P7   P8   P10
  Seg1: ................#.....................1...              >     F    0    0    0
  Seg9: ................#.....................1...              >          0    0    0
 Seg10: ................#.#...................1.#.              >                    0
 Seg11: ..................#.....................1.              >     T    -    +    0
 Seg15: ..................=-....................#1              >     T    -    +
 Seg18: ..................--....................#1              >     T    -    +
 Seg22: ................---....................##1              >          -    +
 Seg24: ................-.....................1..              >     F    -    +
 Seg30: ................-.....................1..                   F    -    +
 Seg32: ..........................................           (< =)    F    -    +
 Seg33: ??????????????????????????????????????????????????                 -
```

Having failed to make the pinterp-space consistent, the forgetting of $P10$ must be retracted, resulting in Snapshot 1-3. G_{32} again becomes inconsistent since the pinterp-space is now as it was before $P10$ was forgotten. Forgettings of other properties are similarly tried and retracted until one works or until forgettings of the recent properties of each name have been tried.

Snapshot 1-3: *INCONSISTENT*

```
Retract fix-hypothesis <Fix-1 {Forget P10}> ...
  ***** Retracted fix-hypothesis <Fix-1 {Forget P10}>! *****

                1111111111222222222233333333334444
       1234567890123456789012345678901234567890123456789012     P3    P4   P7   P8   P10
  Seg1: ................#.....................1...              >     F    0    0    0
  Seg9: ................#.....................1...              >          0    0    0
 Seg10: ................#.#...................1.#.              >                    0
 Seg11: ................#.....................1.               >     T    -    +    0
 Seg15: ................=-....................#1               >     T    -    +
 Seg18: ................-.....................1               >     T    -    +    +
 Seg22: ................-.-...................#.1              >          -    +    +
 Seg24: ................-.....................1..              >     F    -    +    +
 Seg30: ................-.....................1..                   F    -    +    +
 Seg32: ........................................              (< =)     F    -    +    +
 Seg33: ????????????????????????????????????????????????                   -
```

The forgetting of $P4$ succeeds in giving G_{32} some ACTIVE pinterps, as Snapshot 1-4 shows. Forgetting all the most recent occurences of $P4$ having value "F" acknowledges that the pump status indicator light may have burnt out sometime after G_{18} and falsely indicated that the pump has been OFF since G_{24}.

The best global interpretation across the segments at this point is given by the chain of pinterps noted as ACTIVE in Snapshot 1-4 by the digits instead of "#". Such digits indicate the temporal order of pinterps in hidden-transition paths. Thus, the global interpretation crosses over G_{30} with $P(G_{30}, S_{38})$ followed by $P(G_{30}, S_{32})$. Note that domain-dependent probabilistic knowledge is used to determine the best global interpretation. For example, the best global interpretation starts at $P(G_1, S_{39})$ instead of $P(G_1, S_{17})$. This is partly because S_{17} is the *a priori* unlikely state where *all* of the water is in CAN1. S_{39} is a more likely state where both have some water.

Having removed the inconsistency, DATMI is able to interpret the rest of the observations, as shown in Snapshot 1-5. All observations of $P4$ after G_{32} suggest value "F" for $P4$ because the pump indicator has burnt out by then. These later "F" values are ignored because the fix-hypothesis for $P4$ is active since G_{32}. This acknowledges that this indicator cannot be used to determine the status of the pump once that indicator is assumed to have failed.

Note that over segments G_{75} to G_{149} there are many segments differing only in their disjunctive values for $P7$, $P8$, and $P10$. This arises when the values of these CHANGE properties are close to zero and approaching it. These disjunctive values indicate that the actual values may be significantly positive or negative, or they may be zero but the imprecise sensor is not giving an exact zero value.

Snapshot 1-4:

```
Try SENSOR-FAILURE FIX-HYPOTHESIS:
  Forget P4 from <Seg 22> to <Seg 32> ...
  ***** <Fix-2 {Forget P4}> worked! *****

                 11111111112222222222333333333344
       1234567890123456789012345678901234567890123456789012     P3    P4    P7    P8   P10
  Seg1: ................#.....................1...      >     F     0     0     0
  Seg9: ................#.....................1...      >           0     0     0
 Seg10: ................#.#...................1.#.      >                       0
 Seg11: ................#.....................1.      >     T     -     +     0
 Seg15: ................=-....................#1      >     T     -     +
 Seg18: ................-.....................1      >     T     -     +     +
 Seg22: ................-.-...................#.1      >           -     +     +
 Seg24: ................-.-...................#.1      >           -     +     +
 Seg30: ................-.-...........2.....1.#.#                 -     +     +
 Seg32: ..............................1.....#....  (< =)           -     +     +
 Seg33: ................---.........#..1....-=.---                 -
Interpretation credibility = 1.369565E-10 (1.9281808E-6 normalized)
{Normalization under-estimated due to cycles}
Best interpretation ending in state other than 32 ends in state 29:
   It's credibility is 1.00 times smaller.
   {cred = 1.3695513E-10  (normalized: 1.9281615E-6)}
```

Snapshot 1-5: ***Status after processing all observations***

```
                  1111111111222222222233333333334444
        12345678901234567890123456789012345678901234     P3     P4     P7     P8    P10
  Seg1: ................#.....................1...      >      F      0      0      0
  Seg9: ................#.....................1...      >             0      0      0
 Seg10: ................#.#...................1.#.      >                           0
 Seg11: .................#.....................1.      >      T      -      +      0
 Seg15: .................=-.....................#1      >      T      -      +
 Seg18: .................-.....................1      >      T      -      +      +
 Seg22: ................-.-....................#.1      >             -      +      +
 Seg24: ................-.-....................#.1      >             -      +      +
 Seg30: ................-.-...........2.....1.#.#                    -      +      +
 Seg32: ..............................1.....#....  (< =)             -      +      +
 Seg33: ..............................1.........      <              -      +      +
 Seg62: ............................#..1.........      <              -      +  (+ 0)
 Seg74: ............................#..1.........      <              -  (+ 0)  (+ 0)
 Seg75: ....-##....##.......-##....##..#1........      <         (- 0)  (+ 0)  (+ 0)
Seg111: ....-##....##.......-##....##...1........      <         (- 0)  (+ 0)      0
Seg112: ....-##....##.......-##....##...1........      <         (- 0)  (+ 0)  (- 0)
Seg113: ....-##....##.......-##....##..-1........      <         (- 0)  (+ 0)
Seg115: ....-##....##.......-##....##..-1........      <         (- 0)  (+ 0)  (+ 0)
Seg116: ....-##....##.......-##....##...1........      <         (- 0)  (+ 0)      0
Seg120: ....-##....##.......-##....#....1........      <         (- 0)      0  (+ 0)
Seg121: ....-##....##.......-##....##..-1........      <         (- 0)  (+ 0)  (+ 0)
Seg122: ....-##....##.......-##....#....1........      <             0  (+ 0)  (+ 0)
Seg123: ....-##....##.......-##....#....1........      <             0      0
Seg124: ....-##....##.......-##....#....1........      <         (- 0)      0  (- 0)
Seg125: ....-##....##.......-##....##...1........      <         (- 0)  (+ 0)      0
Seg126: ....-##....##.......-##....#....1........      <             0  (+ 0)      0
Seg127: ....-##....##.......-##....#....1........      <             0      0      0
Seg129: ....-##....##.......-##....##...1........      <         (- 0)  (+ 0)      0
Seg131: ....-##....##.......-##....#....1........      <             0      0      0
Seg137: ....-##....##.......-##....##...1........      <         (- 0)  (+ 0)      0
Seg139: ....-##....##.......-##....#....1........      <             0      0      0
Seg147: ....-##....##.......-##....#....1........      <             0      0  (+ 0)
Seg149: ....-##....##.......-##....#....1........      <             0      0      0
Interpretation credibility = 2.0934093E-26 (2.9532617E-22 normalized)
{Normalization under-estimated due to cycles}
Best interpretations ending in states other than state 33:
   End in one of states: (23 28).
   All have credibility 1.06 times smaller.
   {cred = 1.9802324E-26  (normalized: 2.7935983E-22)}
```

Final Summary at Snapshot 1-5:

```
================ Global Interpretation: ================
From    0.0   <Seg 1> to    5.0  <Seg 10>:     39
From    5.0  <Seg 11> to    7.0  <Seg 11>:     41
From    7.0  <Seg 15> to   14.0  <Seg 24>:     42
From   14.0  <Seg 30> to   15.0  <Seg 30>:     38   32
From   15.0  <Seg 32> to   36.5  <Seg 74>:     32
From   36.5  <Seg 75> to   79.5 <Seg 149>:     33
========================================================
Number of numerical measurements:  644
--- Single active fix-hypothesis: ---
<Fix-2 {Forget P4}> for value FALSE
   type: SENSOR-FAILURE; time: 10.0 to 79.5; inconsist-seg: <Seg 32>
System Run Time = 6.87 seconds
```

B.1.2 Example 2: Miscalibrated-Sensors for Water Levels

In this example, DATMI changes an earlier hypothesis, that a pump indicator has failed, to a new hypothesis that the sensors of the water levels in the two cans are miscalibrated. The hypothesized failure of the pump indicator must be retracted when the indicator lights up, suggesting that the pump is ON, after the indicator light was assumed to have burnt out earlier. This retraction is necessary because such a sensor failure is assumed to not be intermittent.

Snapshot 2-1 – Snapshot 2-4 are the same as Snapshot 1-1 – Snapshot 1-4, except that $P10$ is not observed after G_{24}. Thus, DATMI again initially decides to doubt the pump indicator to fix the inconsistency at G_{32}.

Snapshot 2-1: *INCONSISTENT*

```
***** Inconsistent segment <Seg 32> detected! *****

                 11111111112222222222333333333344
       1234567890123456789012345678901234567890123456789012     P3    P4    P7    P8   P10
  Seg1: ..................#.....................1...      >     F     0     0     0
  Seg9: ..................#.....................1...      >           0     0     0
 Seg10: ..................#.#...................1.#.      >                       0
 Seg11: ...................#....................1.      >     T     -     +     0
 Seg15: ...................=-...................#1      >     T     -     +
 Seg18: ....................-...................1      >     T     -     +     +
 Seg22: ..................-.-...................#.1      >           -     +     +
 Seg24: ..................-....................1..      >     F     -     +     +
 Seg30: ..................-....................1..            F     -     +
 Seg32: ........................................... (< =)     F     -     +
 Seg33: ???????????????????????????????????????????                 -
```

Snapshot 2-2: *INCONSISTENT*

```
Try SENSOR-FAILURE FIX-HYPOTHESIS:
  Forget P10 from <Seg 15> to <Seg 24> ...
  ***** Pinterp-space still inconsistent at <Seg 32>! *****

                 11111111112222222222333333333344
       1234567890123456789012345678901234567890123456789012     P3    P4    P7    P8   P10
  Seg1: ..................#.....................1...      >     F     0     0     0
  Seg9: ..................#.....................1...      >           0     0     0
 Seg10: ..................#.#...................1.#.      >                       0
 Seg11: ...................#....................1.      >     T     -     +     0
 Seg15: ...................=-...................#1      >     T     -     +
 Seg18: ...................--...................#1      >     T     -     +
 Seg22: ..................---..................##1      >           -     +
 Seg24: ..................-....................1..      >     F     -     +
 Seg30: ..................-....................1..            F     -     +
 Seg32: ........................................... (< =)     F     -     +
 Seg33: ???????????????????????????????????????????                 -
```

Snapshot 2-3: *INCONSISTENT*

```
Retract fix-hypothesis <Fix-1 {Forget P10}> ...
  ***** Retracted fix-hypothesis <Fix-1 {Forget P10}>! *****

                1111111111222222222233333333334444
       12345678901234567890123456789012345678901 2     P3    P4    P7    P8   P10
 Seg1: ................#.....................1...      >     F     0     0     0
 Seg9: ................#.....................1...      >           0     0     0
Seg10: ................#.#...................1.#.      >                       0
Seg11: ................#.....................1.       >     T     -     +     0
Seg15: ................=-....................#1       >     T     -     +
Seg18: ................-.....................1       >     T     -     +     +
Seg22: ................-.-...................#.1      >           -     +     +
Seg24: ................-.....................1..      >     F     -     +     +
Seg30: ................-.....................1..            F     -     +
Seg32: ..........................................  (< =)     F     -     +
Seg33: ??????????????????????????????????????????                  -
```

Snapshot 2-4:

```
Try SENSOR-FAILURE FIX-HYPOTHESIS:
  Forget P4 from <Seg 22> to <Seg 32> ...
  ***** <Fix-2 {Forget P4}> worked! *****

                1111111111222222222233333333334444
       12345678901234567890123456789012345678901 2     P3    P4    P7    P8   P10
 Seg1: ................#.....................1...      >     F     0     0     0
 Seg9: ................#.....................1...      >           0     0     0
Seg10: ................#.#...................1.#.      >                       0
Seg11: ................#.....................1.       >     T     -     +     0
Seg15: ................=-....................#1       >     T     -     +
Seg18: ................-.....................1       >     T     -     +     +
Seg22: ................-.-...................#.1      >           -     +     +
Seg24: ................-.-...................#.1      >           -     +     +
Seg30: ...............---.....#..2....#1.###                      -     +
Seg32: .........................#..1....##....  (< =)              -     +
Seg33: ...............---.....#..1....==.---                      -
Interpretation credibility = 1.369565E-10 (1.9256245E-6 normalized)
{Normalization under-estimated due to cycles}
Best interpretation ending in state other than 32 ends in state 29:
   It's credibility is 1.00 times smaller.
   {cred = 1.3695513E-10  (normalized: 1.9256051E-6)}
```

Snapshot 2-5 shows the situation when $P4$ is observed to have value "T" just after G_{40}. This conflicts with the assumption of the active fix-hypothesis that $P4$ can only be "F" after G_{32}. Thus, this fix-hypothesis must be retracted.

Snapshot 2-5:

```
***** Active sensor-failure hypothesis contradicted! *****
  Property P4 observed with value "T" at time 20.5.
  <Fix-2 {Forget P4}> expected value "F".

                 11111111112222222222333333333344
       1234567890123456789012345678901234567890123456789012     P3    P4   P7   P8   P10
  Seg1: ..................#.....................1...      >     F    0    0    0
  Seg9: ..................#.....................1...      >          0    0    0
 Seg10: ..................#.#...................1.#.      >                    0
 Seg11: ...................#.....................1.      >     T    -    +    0
 Seg15: ...................=-....................#1      >     T    -    +
 Seg18: ....................-.....................1      >     T    -    +    +
 Seg22: ..................-.-....................#.1      >          -    +    +
 Seg24: ..................-.-....................#.1      >          -    +    +
 Seg30: ..................---........#..2....#1.###                  -    +
 Seg32: ..............................#..1....##....  (< =)          -    +
 Seg33: ..............................#..1.........     <            -    +
 Seg40: ..............................#..1.........     <            -    +
Interpretation credibility = 1.9565214E-11 (2.7568046E-7 normalized)
{Normalization under-estimated due to cycles}
Best interpretation ending in state other than 32 ends in state 29:
   It's credibility is 1.00 times smaller.
   {cred = 1.9565018E-11  (normalized: 2.756777E-7)}
```

During the course of retracting the fix-hypothesis of forgetting $P4$, the originally inconsistency at G_{32} is again detected at Snapshot 2-6. Now, alternative fix-hypotheses must be tried, as Snapshot 2-7 - Snapshot 2-12 illustrate.

Finally, forgetting the ORDER relation ($P3$) between the water levels of the two containers succeeds in fixing the pinterp-space at Snapshot 2-13.

Having fixed the inconsistent pinterp-space, the observation of $P4$ being "T" at the segment after G_{40} is now asserted, as Snapshot 2-14 shows.

Snapshot 2-6: *INCONSISTENT*

```
Retract sensor-failure hypothesis <Fix-2 {Forget P4}> ...
***** Inconsistent segment <Seg 32> detected! *****

                 1111111111222222222233333333334444
       12345678901234567890123456789012345678901234    P3    P4    P7    P8   P10
 Seg1: ................#.....................1...      >     F     0     0     0
 Seg9: ................#.....................1...      >           0     0     0
Seg10: ................#.#...................1.#.      >                       0
Seg11: .................#.....................1.      >     T     -     +     0
Seg15: .................=-....................#1      >     T     -     +
Seg18: .................-.....................1      >     T     -     +     +
Seg22: .................-.-...................#.1      >           -     +     +
Seg24: .................-.....................1..      >     F     -     +     +
Seg30: .................-.....................1..            F     -     +
Seg32: ............................-..-....--....   (< =)           -     +
Seg33: ............................#..1..........      <           -     +
Seg40: ............................#..1..........      <           -     +
```

Snapshot 2-7: *INCONSISTENT*

```
Try SENSOR-FAILURE FIX-HYPOTHESIS:
  Forget P10 from <Seg 15> to <Seg 24> ...
  ***** Pinterp-space still inconsistent at <Seg 32>! *****

                 1111111111222222222233333333334444
       12345678901234567890123456789012345678901234    P3    P4    P7    P8   P10
 Seg1: ................#.....................1...      >     F     0     0     0
 Seg9: ................#.....................1...      >           0     0     0
Seg10: ................#.#...................1.#.      >                       0
Seg11: .................#.....................1.      >     T     -     +     0
Seg15: .................=-....................#1      >     T     -     +
Seg18: .................--....................#1      >     T     -     +
Seg22: .................---...................##1      >           -     +
Seg24: .................-.....................1..      >     F     -     +
Seg30: .................-.....................1..            F     -     +
Seg32: ............................-..-....--....   (< =)           -     +
Seg33: ............................#..1..........      <           -     +
Seg40: ............................#..1..........      <           -     +
```

Snapshot 2-8: *INCONSISTENT*

```
Retract fix-hypothesis <Fix-3 (Forget P10)> ...
  ***** Retracted fix-hypothesis <Fix-3 (Forget P10)>! *****

                1111111111222222222233333333334444
       12345678901234567890123456789012345678901 2    P3    P4    P7    P8   P10
 Seg1: ................#.....................1...     >     F     0     0     0
 Seg9: ................#.....................1...     >           0     0     0
Seg10: ................#.#...................1.#.     >                       0
Seg11: ..................#.....................1.     >     T     -     +     0
Seg15: ..................=-....................#1     >     T     -     +
Seg18: ...................-....................1     >     T     -     +     +
Seg22: ..................-.-...................#.1     >           -     +     +
Seg24: ..................-.....................1..     >     F     -     +     +
Seg30: ..................-.....................1..           F     -     +
Seg32: ............................-..-....--....  (< =)           -     +
Seg33: ............................#..1.........     <           -     +
Seg40: ............................#..1.........     <           -     +
```

Snapshot 2-9: *INCONSISTENT*

```
Try SENSOR-FAILURE FIX-HYPOTHESIS:
  Forget P8 from <Seg 10> to <Seg 40> ...
  ***** Pinterp-space still inconsistent at <Seg 32>! *****

                1111111111222222222233333333334444
       12345678901234567890123456789012345678901 2    P3    P4    P7    P8   P10
 Seg1: ................#.....................1...     >     F     0     0     0
 Seg9: ................#.....................1...     >           0     0     0
Seg10: ................#.#...................1.#.     >                       0
Seg11: ..................#.....................1.     >     T     -           0
Seg15: ..................=-....................#1     >     T     -
Seg18: ...................-....................1     >     T     -           +
Seg22: ..................-.-...................#.1     >           -           +
Seg24: ..................-.....................1..     >     F     -           +
Seg30: ..................-.....................1..           F     -
Seg32: ............................-..-....--....  (< =)           -
Seg33: ............................#..1.........     <           -
Seg40: ............................#..1.........     <           -
```

Snapshot 2-10: *INCONSISTENT*

```
Retract fix-hypothesis <Fix-4 {Forget P8}> ...
  ***** Retracted fix-hypothesis <Fix-4 {Forget P8}>! *****

                 11111111112222222222333333333344
      .          12345678901234567890123456789012345678901234567890123456789012      P3    P4    P7    P8    P10
  Seg1: ................#.....................1...                                     >     F     0     0     0
  Seg9: ................#.....................1...                                     >           0     0     0
 Seg10: ................#.#...................1.#.                                    >                       0
 Seg11: .................#.....................1.                                     >     T     -     +     0
 Seg15: .................=-....................#1                                     >     T     -     +
 Seg18: .................-.....................1                                     >     T     -     +     +
 Seg22: .................-.-...................#.1                                    >           -     +     +
 Seg24: .................-.....................1..                                    >     F     -     +     +
 Seg30: .................-.....................1..                                          F     -     +
 Seg32: ............................-..-....--....                                  (< =)       -     +
 Seg33: ............................#..1.........                                     <           -     +
 Seg40: ............................#..1.........                                     <           -     +
```

Snapshot 2-11: *INCONSISTENT*

```
Try SENSOR-FAILURE FIX-HYPOTHESIS:
  Forget P7 from <Seg 10> to <Seg 40> ...
  ***** Pinterp-space still inconsistent at <Seg 32>! *****

                 11111111112222222222333333333344
      .          12345678901234567890123456789012345678901234567890123456789012      P3    P4    P7    P8    P10
  Seg1: ................#.....................1...                                     >     F     0     0     0
  Seg9: ................#.....................1...                                     >           0     0     0
 Seg10: ................#.#...................1.#.                                    >                       0
 Seg11: .................#.....................1.                                     >     T           +     0
 Seg15: .................=-....................#1                                     >     T           +
 Seg18: .................-.....................1                                     >     T           +     +
 Seg22: .................-.-...................#.1                                    >                 +     +
 Seg24: .................-.....................1..                                    >     F           +     +
 Seg30: .................-.....................1..                                          F           +
 Seg32: ............................-..-....--....                                  (< =)             +
 Seg33: ............................#..1.........                                     <                 +
 Seg40: ............................#..1.........                                     <                 +
```

Snapshot 2-12: *INCONSISTENT*

```
Retract fix-hypothesis <Fix-5 {Forget P7}> ...
  ***** Retracted fix-hypothesis <Fix-5 {Forget P7}>! *****

                1111111111222222222233333333334444
       12345678901234567890123456789012345678901 2    P3    P4    P7    P8   P10
 Seg1: .....................#.....................1...     >     F     0     0     0
 Seg9: .....................#.....................1...     >           0     0     0
Seg10: .....................#.#...................1.#.     >                       0
Seg11: .....................#.....................1.     >     T     -     +     0
Seg15: .....................=-....................#1     >     T     -     +
Seg18: .....................-.....................1     >     T     -     +     +
Seg22: .....................-.-...................#.1     >           -     +     +
Seg24: .....................-.....................1..     >     F     -     +     +
Seg30: .....................-.....................1..           F     -     +
Seg32: ............................-..-....--.....  (< =)           -     +
Seg33: ............................#..1...........     <           -     +
Seg40: ............................#..1...........     <           -     +
```

Snapshot 2-13:

```
Try SENSOR-FAILURE CALIBRATION FIX-HYPOTHESIS:
  Forget P3 from <Seg 1> to <Seg 40> ...
  ***** <Fix-6 {Forget P3}> worked! *****

                1111111111222222222233333333334444
       12345678901234567890123456789012345678901 2    P3    P4    P7    P8   P10
 Seg1: --..---.........#...---...........--..1...           F     0     0     0
 Seg9: -------....--...#...---....-....-.--..1...                 0     0     0
Seg10: -------...---...#.#.---...---...-.---.1.#.                                0
Seg11: .....................#.........-.......-...1.           T     -     +     0
Seg15: .....................=-.........-..-....--..#1           T     -     +
Seg18: .....................-...........-.....-...1           T     -     +     +
Seg22: .....................-.-...........-.....-.#.1                 -     +     +
Seg24: .....................-.....................1..           F     -     +     +
Seg30: .....................-.....................1..           F     -     +
Seg32: .....................---.........#..#....##.1##                 -     +
Seg33: .....................---.........#..#....##.1##                 -     +
Seg40: .....................---.........#..#....##.1##                 -     +
Interpretation credibility = 5.8700926E-11 (8.3307356E-7 normalized)
{Normalization under-estimated due to cycles}
Best interpretation ending in state other than 40 ends in state 42:
   It's credibility is 1.00 times smaller.
   {cred = 5.870034E-11  (normalized: 8.330653E-7)}
```

Snapshot 2-14:

```
***** Retracted contradicted fix-hypothesis <Fix-2 {Forget P4}>! *****

                 11111111112222222222333333333344
        1234567890123456789012345678901234567890123     P3    P4    P7    P8   P10
  Seg1: --..---.........#...---...........--..1...           F     0     0     0
  Seg9: -------....--...#...---....-....-.--..1...           0     0     0
 Seg10: -------...---...#.#.---...---...-.---.1.#.                       0
 Seg11: ..................#.........-.......-...1.           T     -     +     0
 Seg15: ..................=-........-..-....--..#1          T     -     +
 Seg18: ..................-...........-.....-...1           T     -     +     +
 Seg22: .................-.-...........-.....-.#.1                 -     +     +
 Seg24: .................-.....................1..           F     -     +     +
 Seg30: .................-.....................1..           F     -     +
 Seg32: .................-.....................1..           F     -     +
 Seg33: .................-.....................1..           F     -     +
 Seg40: .................---.........#..#....##.##1                 -     +
 Seg43: .................--.........#..#....##..#1           T     -     +
Interpretation credibility = 8.385762E-12 (1.1906554E-7 normalized)
{Normalization under-estimated due to cycles}
Best interpretation ending in state other than 42 ends in state 41:
   It's credibility is 1.00 times smaller.
   {cred = 8.385678E-12  (normalized: 1.19064346E-7)}
```

The rest of the observations are asserted without difficulty, resulting in Snapshot 2-15. Note that G_{121} is a gap-fill segment since no observations are observed during that time interval. However, even with no property constraints for G_{121}, almost half of its pinterps are INACTIVE because there are no valid dependency paths for them.

Snapshot 2-15: *Status after processing all observations*

```
                    1111111111222222222233333333334444
          12345678901234567890123456789012345678901234     P3    P4    P7    P8   P10
  Seg1: --..---.........#...---...........--..1...             F     0     0     0
  Seg9: -------....--...#...---....-....-.--..1...                   0     0     0
 Seg10: -------...---...#.#.---...---...-.---.1.#.                                0
 Seg11: ..................#.........-.......-...1.             T     -     +     0
 Seg15: ..................=-........-..-....--..#1            T     -     +
 Seg18: ..................-...........-.....-...1             T     -     +     +
 Seg22: .................-.-...........-.....-.#.1                   -     +     +
 Seg24: .................-.......................1..          F     -     +     +
 Seg30: .................-.......................1..          F     -     +
 Seg32: .................-.......................1..          F     -     +
 Seg33: .................-.......................1..          F     -     +
 Seg40: .................---........#..#....#2.##1                   -     +
 Seg43: .................--........#..1....##..##            T     -     +
 Seg44: .................--........#..1....##..##            T     -     +
 Seg75: .................--........#..1....##..##            T     - (+ 0)
 Seg76: ..--.......##.....--.......##..#1...##..##            T (- 0) (+ 0)
Seg120: ..--.......##.............#....1.........            T (- 0)     0
seg121: -----##-##-##-##-----##-##-##-##1#####----
Seg122: ..--.......##.............#....1.........            T     0     0
Seg126: ..--.......##.....--.......##..-1...--..--           T (- 0) (+ 0)
Seg128: ..--.......##.............#....1.........            T     0     0
Interpretation credibility = 1.3037649E-17 (1.8511554E-13 normalized)
{Normalization under-estimated due to cycles}
Best interpretation ending in state other than 33 ends in state 28:
   It's credibility is 1.06 times smaller.
   {cred = 1.2332789E-17  (normalized: 1.7510755E-13)}
```

Final Summary at Snapshot 2-15:

```
========== Global Interpretation: ==========
From    0.0   <Seg 1> to    5.0  <Seg 10>:    39
From    5.0  <Seg 11> to    7.0  <Seg 11>:    41
From    7.0  <Seg 15> to   11.0  <Seg 22>:    42
From   11.0  <Seg 24> to   19.0  <Seg 33>:    40
From   19.0  <Seg 40> to   20.0  <Seg 40>:    42  38
From   20.0  <Seg 43> to   36.5  <Seg 75>:    32
From   36.5  <Seg 76> to   59.0 <Seg 120>:    33
   { ----- no state transitions during gap ----- }
From   65.0 <Seg 122> to   69.0 <Seg 128>:    33
============================================================
Number of numerical measurements:  420
--- Single active fix-hypothesis: ---
<Fix-6 {Forget P3}> for value (< > = UNREL ?)
   type: SENSOR-FAILURE; time: 0.0 to 69.0; inconsist-seg: <Seg 32>
System Run Time = 13.45 seconds
```

B.2 Handling Sensor Failures with Property Probabilities

B.2.1 Example 3: An Unlikely Failure of the Pump Indicator

The same observations are used for this example as for Example 1. However, the quantity-space conversion tables for this example assign some non-zero probability to every possible value for property $P4$. They also avoid asserting that $P7$ or $P8$ ever have value 0 without also admitting that they might be somewhat postive or negative instead.

These quantity-space conversion tables indicate that when the pump indicator says the pump is `OFF`, then believe that the pump is actually `OFF` with confidence 0.99 and that the pump is `ON` with confidence 0.01. Thus, when the pump must actually be `ON` at some point where the indicator claims it is `OFF`, there will still be a global interpretation, of rather low *a priori* probability, that the pump is `ON`.

Snapshot 3-1 shows the pinterp-space resulting from using this alternative quantity-space conversion table. In G_{32}, the value of "T" for $P4$ allows `ACTIVE` pinterps $P(G_{32}, S_{32})$ and $P(G_{32}, S_{38})$. "F" is the *a priori* more plausible value for $P4$ in G_{32} because the indicator claims the pump is `OFF` for G_{32}. However, the availability of the "T" value for $P4$ for G_{32} allows `DATMI` to avoid the process of hypothesizing and testing fix-hypotheses that was used in Example 1.

As should be expected, the pinterp-space of Snapshot 3-1 is more general than the pinterp-space of Snapshot 1-5. The sets of `ACTIVE` and `INACTIVE` pinterps of Snapshot 3-1 properly contain the respective sets of Snapshot 1-5. This results from the more general quantity-space conversion table for this example.

Snapshot 3-1: *Status after processing all observations*

```
                 1111111111222222222233333333334444
        12345678901234567890123456789012345678901 2     P3    P4    P7    P8   P10
  Seg1: ................#.#.................1.#.    > (F T)                     0
 Seg11: ................#.....................1.    > (T F)     -     +     0
 Seg15: ................-=-...................##1    > (T F)     -     +
 Seg18: ................-.-...................#.1    > (T F)     -     +     +
 Seg19: ................-.-...................#.1    > (T F)     -     +     +
 Seg24: ................-.-...................1.#    > (F T)     -     +     +
 Seg30: ................-.-..........#.....#.#.1       (F T)     -     +     +
 Seg32: ..............................2.....1....  (< =) (F T)     -     +     +
 Seg33: ..............................1.........     < (F T)     -     +     +
 Seg62: ............................#..1.........     < (F T)     -     + (+ 0) (+ 0)
 Seg74: ............................1..#.........     < (F T)     - (+ 0) (+ 0)
 Seg75: ....-##....##.......-2#....1#..##.........     < (F T) (- 0) (+ 0) (+ 0)
Seg111: ....-##....##.......-1#....##...#.........     < (F T) (- 0) (+ 0)     0
Seg112: ....-##....##.......-1#....##...#.........     < (F T) (- 0) (+ 0) (- 0)
Seg113: ....-##....##.......-1#....##..-#.........     < (F T) (- 0) (+ 0)
Seg115: ....-##....##.......-1#....##..-#.........     < (F T) (- 0) (+ 0) (+ 0)
Seg116: ....-##....##.......-1#....##...#.........     < (F T) (- 0)           0
Seg117: ....-##....##.......-1#....##...#.........     < (F T)       (+ 0)     0
Seg118: ....-##....##.......-1#....##...#.........     < (F T) (- 0) (+ 0)     0
Seg119: ....-##...-##.......-1#...-##...#.........     < (F T)                 0
Seg120: ....-##....##.......-1#....##..-#.........     < (F T) (- 0)       (+ 0)
Seg121: ....-##....##.......-1#....##..-#.........     < (F T)       (+ 0) (+ 0)
Seg122: ....-##...-##.......-1#...-##..-#.........     < (F T)             (+ 0)
Seg123: ....-##-##-##-##....-1#-##-##-#-##........     < (F T)
Seg124: ....-##....##.......-1#....##...#.........     < (F T) (- 0)       (- 0)
Seg125: ....-##....##.......-1#....##...#.........     < (F T)       (+ 0)     0
Seg126: ....-##...-##.......-1#...-##...#.........     < (F T)                 0
Seg129: ....-##....##.......-1#....##...#.........     < (F T) (- 0) (+ 0)     0
Seg130: ....-##...-##.......-1#...-##...#.........     < (F T)                 0
Seg137: ....-##....##.......-1#....##...#.........     < (F T) (- 0) (+ 0)     0
Seg138: ....-##...-##.......-1#...-##...#.........     < (F T)                 0
Seg147: ....-##...-##.......-1#...-##..-#.........     < (F T)             (+ 0)
Seg149: ....-##...-##.......-1#...-##...#.........     < (F T)                 0
Interpretation credibility = 3.049393E-33 (8.797574E-31 normalized)
{Normalization under-estimated due to cycles}
Best interpretation ending in state other than 22 ends in state 28:
   It's credibility is 1.00 times smaller.
   {cred = 3.0493626E-33  (normalized: 8.797485E-31)}
```

Final Summary at Snapshot 3-1:

```
============== Global Interpretation: ==============
From    0.0   <Seg 1> to    5.0   <Seg 1>:    39
From    5.0  <Seg 11> to    7.0  <Seg 11>:    41
From    7.0  <Seg 15> to   11.0  <Seg 19>:    42
From   11.0  <Seg 24> to   14.0  <Seg 24>:    40
From   14.0  <Seg 30> to   15.0  <Seg 30>:    42
From   15.0  <Seg 32> to   15.5  <Seg 32>:    38   32
From   15.5  <Seg 33> to   36.0  <Seg 62>:    32
From   36.0  <Seg 74> to   36.5  <Seg 74>:    29
From   36.5  <Seg 75> to   54.5  <Seg 75>:    28   22
From   54.5 <Seg 111> to   79.5 <Seg 149>:    22
====================================================

Number of numerical measurements:  644
System Run Time = 6.57 seconds
```

B.3 Noticing State-Duration Conflicts

B.3.1 Example 4: Drainage Taking Longer Than Expected

In this example, DATMI initially believes that water is continuously draining from a container. However, when the water flow from this container lasts longer than the maximum time required to drain a full container of water, DATMI changes its best interpretation to one where the valve is not OPEN for so long.

Domain-specific state-duration information indicates that a state cannot persist for more than 125 seconds when flow occurs through the pipe but no pumping occurs. Similarly, a state can persist for at most 65 seconds when pumping occurs but no flow through the pipe occurs. These upper bound state-durations are based on knowledge about how long it takes for a full container of water to drain or to be pumped dry.

At Snapshot 4-1, it is consistent that the water has been draining for the last 50 seconds in state S_{40}. This is because $D_U(S_{40}) = 125$, according to the above domain-specific state-duration information, and $125 > 50$.

Snapshot 4-1: *Through time = 50 seconds*

```
                1111111111222222222233333333334444
       12345678901234567890123456789012345678901 2     P4    P5    P7    P8
  Seg1: ..................#.....................1..      F     T     -     +
  Seg4: --..------.....-=..------.......##..#1..       F
Interpretation credibility = 0.0060430258 (0.28767118 normalized)
Best interpretations ending in states other than state 40:
   End in one of states: (39 36).
   All have credibility 1.00 times smaller.
   {cred = 0.006042965  (normalized: 0.28766832)}
```

At Snapshot 4-2, state S_{40} may still be spanning for the entire 100 seconds since $D_U(S_{40})$ $= 125 > 100$.

Snapshot 4-2: *Through time = 100 seconds*

```
                1111111111222222222233333333334444
       12345678901234567890123456789012345678901 2     P4    P5    P7    P8
  Seg1: ..................#.....................1..      F     T     -     +
  Seg4: --..------.....-=..------.......##..#1..       F
Interpretation credibility = 0.0060430258 (0.28767118 normalized)
Best interpretations ending in states other than state 40:
   End in one of states: (39 36).
   All have credibility 1.00 times smaller.
   {cred = 0.006042965  (normalized: 0.28766832)}
```

However, at Snapshot 4-3, the pump-cycle cannot span with S_{40} for the entire 140 seconds since $D_U(S_{40}) = 125 < 140$. Thus, $P(G_{16}, S_{40})$ cannot b-depend on $P(G_4, S_{40})$. The best global interpretations becomes the ones ending at $P(G_{16}, S_{36})$ and $P(G_{16}, S_{39})$. These interpretations still start at $P(G_1, S_{40})$, but they do not span all the segments with S_{40}. The best global interpretation ending at $P(G_{16}, S_{36})$ passes through $P(G_4, S_{36})$ because it is more plausible to

stay in the same state for both G_4 and G_{16}. Similarly, the best global interpretation ending at $P(G_{16}, S_{39})$ passes through $P(G_4, S_{39})$.

Snapshot 4-3: *Through time = 140 seconds*

```
                 11111111112222222222333333333344
       12345678901234567890123456789012345678901234     P4    P5    P7    P8
  Seg1: ..................#.....................1..      F     T     -     +
  Seg4: --..------.....-=..------.......#1..##..      F
 Seg16: --..------.....--..------.......#1..##..      F
Interpretation credibility = 0.001208593 (0.10616848 normalized)
{Normalization under-estimated due to cycles}
Best interpretation ending in state other than 36 ends in state 39:
   This alternative interpretation has the same credibility.
```

At Snapshot 4-4, the best global interpretation goes through $P(G_{16}, S_{40})$ instead of $P(G_{16}, S_{36})$, as it did for Snapshot 4-3. This is because the observations for G_{21} make $P(G_{16}, S_{36})$ INACTIVE because there is no transition consistency relation between $P(G_{16}, S_{36})$ and G_{21}. S_{36} is a state where the two water levels are equal and the pump is OFF. So, it is not possible for the water levels to change after being in state S_{36} unless the pump is turned ON. Since the pump is OFF in G_{21}, $P(G_{16}, S_{36})$ must be INACTIVE.

Snapshot 4-4: *Status after processing all observations*

```
                 11111111112222222222333333333344
       12345678901234567890123456789012345678901234     P4    P5    P7    P8
  Seg1: ..................#.....................1..      F     T     -     +
  Seg4: --..------.....-=..------.......--..1#..      F
 Seg16: --..------.....--..------.......--..#1..      F
 Seg21: ..................-.....................1..      F     T     -     +
Interpretation credibility = 1.726544E-4 (0.081958115 normalized)
{Normalization under-estimated due to cycles}
No interpretation ends in a different state!
```

Final Summary at Snapshot 4-4:

```
============== Global Interpretation: ==============
From    0.0   <Seg 1> to    7.0   <Seg 1>:    40
From    7.0   <Seg 4> to  120.0   <Seg 4>:    39
From  120.0  <Seg 16> to  200.0  <Seg 21>:    40
====================================================

Number of property assertions:    41
System Run Time = 5.17 seconds
```

APPENDIX C

THE FROB SINGLE-WELL TOTAL ENVISIONMENT

This appendix describes the single-well model used for the DATMI examples show in Appendix D. As shown in Figure C.1, this single-well system consists of nine outer wall segments (labelled S19, S18, S17, S16, S15, S14, S78, S79, S12). This system includes a narrow well formed by the walls S17, S16, and S15. The behavior of this system is represented by the movement of a ball within these boundaries. This system was modelled with FROB (Forbus, 1981) to provide a total envisionment of all such behaviors.

Describing the properties for all 360 states of the total envisionment for this single-well system would take too much space here. Therefore, descriptions of particular states will be provided as needed in the examples of Appendix D. Each location in the single-well system is either a SREGION or SEGMENT. A SREGION, abbreviated as SR in Figure C.1, describes a non-empty area of space. Alternatively, a SEGMENT, abbreviated as S, indicates either a wall surface or an imaginary border between two SREGION's. Qualitative values (nil, up, down, left, and right) for both the horizontal and vertical directions indicate the motion of the ball during each state. During a particular state, the ball can either *collide* with a wall, *fly* from a SEGMENT or through a SREGION, *pass* through an imaginary border, *stop* at a wall, or *fall* outside of the system. Transitions among all the states of the total envisionment are indicated by Figure C.2. As can be seen in this figure, this is a rather large envisionment. It has 484 state transitions among 360 states.

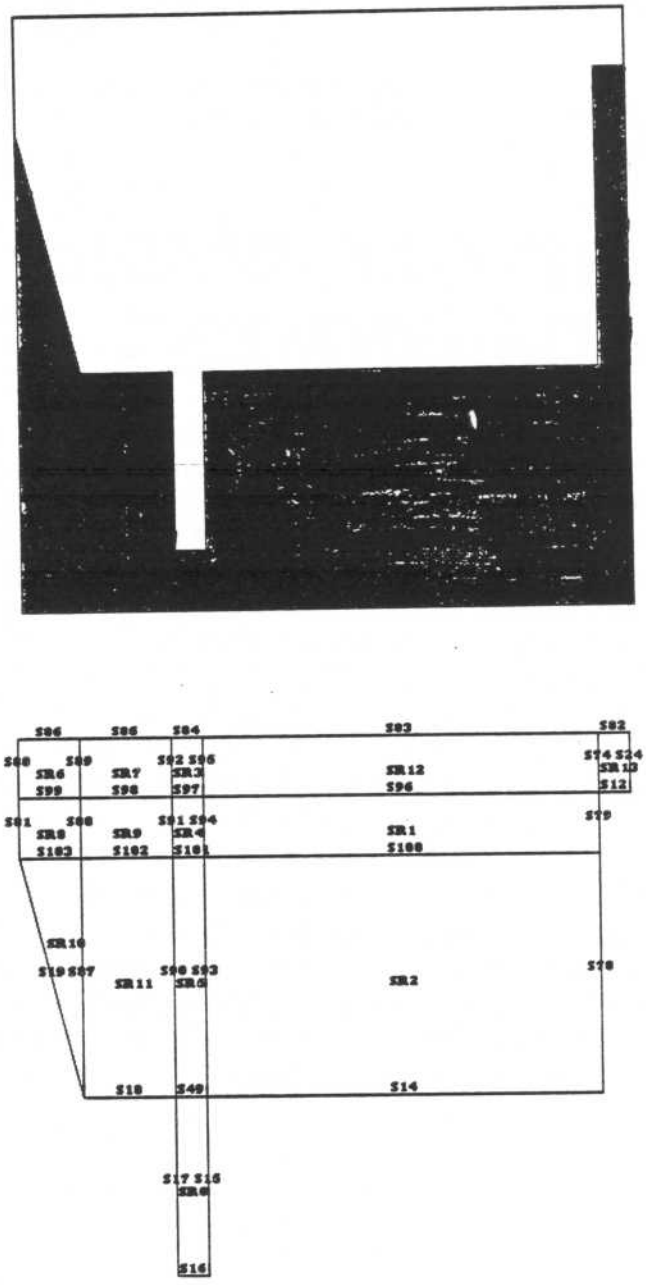


Figure C.1: The single-well scenario

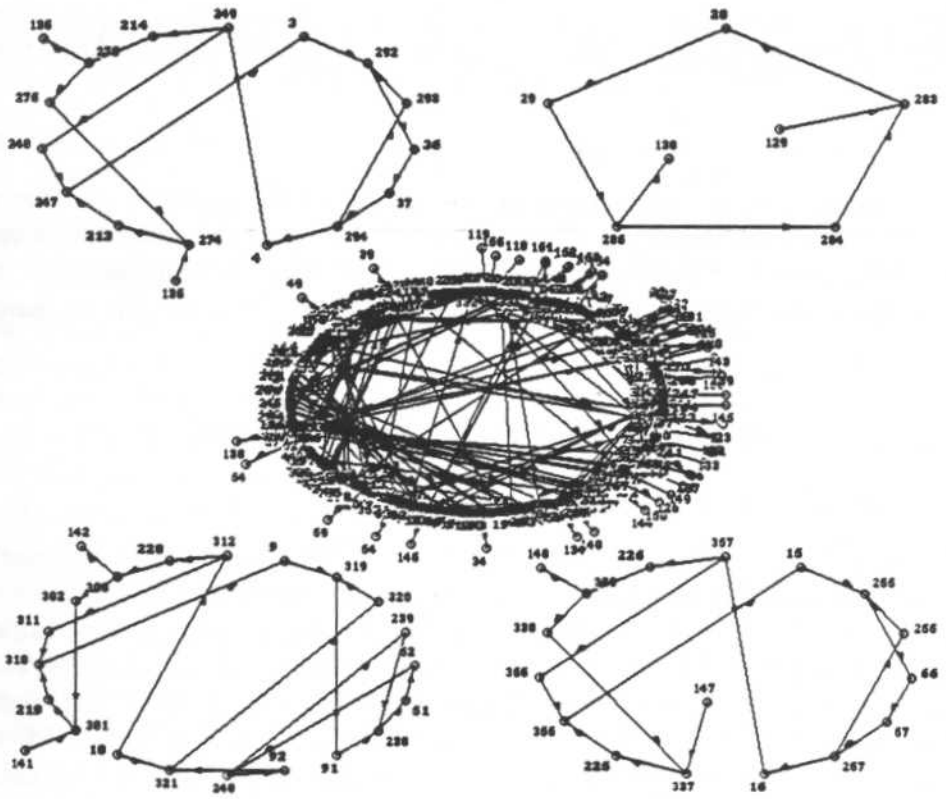


Figure C.2: State-transition diagram for the single-well system
(Instantaneous states are denoted with boldface labels.)

APPENDIX D

DATMI EXAMPLES FOR THE FROB SINGLE-WELL SYSTEM

For these examples, DATMI used the total envisionment described in Appendix C. For each example, this appendix describes the status of the pinterp-space using the snapshot representations introduced in Appendix B. However, because the single-well envisionment is so large, the snapshots of this appendix mention only those states which are COMPATIBLE in at least one segment.

D.1 Maintaining the Most-Probable Interpretation

D.1.1 Example 5: Three Collisions

In this example, DATMI observes three successive collisions of the FROB ball. After just observing the first collision, DATMI suspects that the ball hit the largest wall since that is the *a priori* most likely wall to be hit. After observing the second collision, the working interpretation is adjusted so that the ball hits that largest wall at the second collision. This is necessary since the second collision could not be explained if the ball hit the largest wall first. Alternative interpretations where the ball hits a wall far from the wall that was first hit involve long sequences of states indicating movement through many imaginary borders. Since there are many ways for the ball to move through the space between far walls, the probability of each such sequence is relatively small. Thus, smaller sequences tend to be the best working interpretation, as is the case here. Finally, the third observed collision results in a working interpretation where the largest wall is not hit at all.

Snapshot 5-1 shows that DATMI initially believes that the ball probably hit S14. Property $P2$ is the horizontal (x) direction (Right or Left) of the ball, $P3$ is the vertical (y) direction (Up or Down), and $P4$ is the action of the ball (Fly, Collide, Pass, etc.). Descriptions of the key states for a pinterp-space are shown at the end of the snapshot. The first section of these state descriptions summarize the best working interpretation. The second section describes some interesting states of alternative interpretations.

Domain-specific rules compute higher probability for states where the ball is at S14 than for states where the ball is at some other wall. These rules basically assign *a priori* state probabilities proportional to the area of the ball location specified by the states. However, they also take into consideration the relative difficulty of reaching each state. Thus, states where

the ball is in the narrow well are not as likely as one would expect based solely on the rather long lengths of the well walls.

Snapshot 5-1: *Through time = 2 seconds*

```
                  11111111122222223333333
          33456677011233445455678901234 45
          08530889528417395109876543210 98      P2    P3    P4
  Seg1: ....#..#..--#----####-#1-------         R     D     F
  SEG2: #1##.##.##.....................         R     D     C
Interpretation credibility = 6.452681775320579d-5
Best interpretation ending in state other than 38 ends in state 105:
   It's credibility (3.970881022884232d-5) is 1.63 times smaller.
-------------------------------------------------------------------
State 295: PLACE = SREGION2,  VX = RIGHT, VY = DOWN, ACTION = FLY
State  38: PLACE = SEGMENT14, VX = RIGHT, VY = DOWN, ACTION = COLLIDE
-------------------------------------------------------------------
State 105: PLACE = SEGMENT78, VX = RIGHT, VY = DOWN, ACTION = COLLIDE
```

Hitting S78 was not considered as likely as hitting S14 at Snapshot 5-1, due to the relative lengths of those two walls. However, at Snapshot 5-2, hitting S14 for the first collision is not possible any more because the ball could not then be moving left and down at the second collision, as required by segment G_4. Thus, the best working interpretation at this point is hitting S78 first, and then hitting S14.

After observing all three collisions at Snapshot 5-3, DATMI realizes that the ball must hit either S19 or S17 for the third collision. Since S17 is in the well, the ball will more likely hit S19. Although hitting S14 at the second collision is possible (since $P(G_4, S_{33})$ is ACTIVE), the domain knowledge suggests that it is a little more likely to go from S78 to S19 directly.

The best interpretation contains a hidden-transition of eight states across segment G_3. The movement of the ball from S78 to S19 is largely unconstrained by the observations for G_3. This situation illustrates the need for efficient hidden-transition search.

Some alternative interpretations suggest that the ball may have been in the well for part of the time. It may have bounced around in the well for all three collisions or it may have just fallen into the well for the last two collisions. In either case, the probability of falling into the narrow well is very low. As Snapshot 5-3 shows, interpretations where the ball fell into the well are at least 83.58 times less likely than the best interpretation. Thus, interpretations where the ball is in the well are considered highly unlikely, although they are still considered consistent.

Snapshot 5-2: *Through time = 6 seconds*

```
                                  1111111111111111111111111222222222
          112333444556667778890001122333344455566788990112234 45
        173950381583603818939515828471379359157395173951739517395140     P2     P3     P4
  Seg1: ..............#....-........--.-.-.-.-.-..............#.#       R      D      F
  SEG2: .....-.-.#.-...-.-.....1.#.............................          R      D      C
  Seg3: ####....#..........-#-1.#...-.-.-.-.-.-.##-##-##-----#.#.      L      D (F P)
  SEG4: ....-.1...#.#.#.#.....................................          L      D      C
        22222222223333333333333
        55667788899001122344455
        39281706958473625140392 8
  Seg1: .-.-.-.-.1.-.-.-.-.-.-.-
  SEG2: ........................
  Seg3: #.#.-.-.2.-.#.#.-.-.#.#.
  SEG4: ........................
Interpretation credibility = 4.941192348422297d-9
Best interpretation ending in state other than 33 ends in state 63:
   It's credibility (9.170142555638645d-15) is 538834.85 times smaller.
-----------------------------------------------------------------
State 295: PLACE = SREGION2,  VX = RIGHT, VY = DOWN, ACTION = FLY
State 105: PLACE = SEGMENT78, VX = RIGHT, VY = DOWN, ACTION = COLLIDE
State 101: PLACE = SEGMENT78, VX = LEFT,  VY = DOWN, ACTION = FLY
State 289: PLACE = SREGION2,  VX = LEFT,  VY = DOWN, ACTION = FLY
State  33: PLACE = SEGMENT14, VX = LEFT,  VY = DOWN, ACTION = COLLIDE
-----------------------------------------------------------------
State  63: PLACE = SEGMENT18, VX = LEFT,  VY = DOWN, ACTION = COLLIDE
```

Snapshot 5-3: *Status after processing all observations*

```
                                                  111111111111111111
                 1112223333444455556666777778889990000111223333444455
          12783490570358135803680358134893590571358028471379359 15     P2    P3    P4
  Seg1: ......................#.......-.............--.-.-.-.-.-.-      R     D     F
  SEG2: ..........-..-..#..-.....-...-.........1..#.............      R     D     C
  Seg3: #.#.#.#.......#................-.#.-.1..#....-.-.-.-.-.      L     D (F P)
  SEG4: .........-..#.....#..-..#..1.............................      L     D     C
  Seg5: .-.-.-.-.-.-..#..-..#.....#...1...-.#.-.-..-.............      L     U (F P)
  SEG6: ......................#.....1.............................      L     U     C
        11111111111111122222222222222222222222222222222233333333333333333333
        55666777888899900011112223334445555666777888899900001112223334444555
        79359157137935915712783490571460359248137026915804793682571460359248
  Seg1: ............................#..#..-..-..-..-..1..-..-..-..-..-..-..-..-
  SEG2: ..........................................................................
  Seg3: 7.#.-.5.#.-.3.#.-.-.-.-.-.-.#..#..8..6..-..-..2..-..#..4..-..-..#..#..
  SEG4: ..........................................................................
  Seg5: .#.-.-.#.-.-.#.-.-.-.-.-.-.#..-..2..#..-..-..#..-..-..#..-..-..-..-.
  SEG6: ..........................................................................
Interpretation credibility = 1.6238139670461854d-20
Best interpretation ending in state other than 73 ends in state 58:
   It's credibility (1.9428988031530713d-22) is 83.58 times smaller.
------------------------------------------------------------------------
State 295: PLACE = SREGION2,  VX = RIGHT, VY = DOWN, ACTION = FLY
State 105: PLACE = SEGMENT78, VX = RIGHT, VY = DOWN, ACTION = COLLIDE
State 101: PLACE = SEGMENT78, VX = LEFT,  VY = DOWN, ACTION = FLY
State 289: PLACE = SREGION2,  VX = LEFT,  VY = DOWN, ACTION = FLY
State 193: PLACE = SEGMENT93, VX = LEFT,  VY = DOWN, ACTION = PASS
State 316: PLACE = SREGION5,  VX = LEFT,  VY = DOWN, ACTION = FLY
State 175: PLACE = SEGMENT90, VX = LEFT,  VY = DOWN, ACTION = PASS
State 262: PLACE = SREGION11, VX = LEFT,  VY = DOWN, ACTION = FLY
State 157: PLACE = SEGMENT87, VX = LEFT,  VY = DOWN, ACTION = PASS
State 253: PLACE = SREGION10, VX = LEFT,  VY = DOWN, ACTION = FLY
State  71: PLACE = SEGMENT19, VX = LEFT,  VY = DOWN, ACTION = COLLIDE
State  74: PLACE = SEGMENT19, VX = LEFT,  VY = UP,   ACTION = FLY
State 255: PLACE = SREGION10, VX = LEFT,  VY = UP,   ACTION = FLY
State  73: PLACE = SEGMENT19, VX = LEFT,  VY = UP,   ACTION = COLLIDE
------------------------------------------------------------------------
State  58: PLACE = SEGMENT17, VX = LEFT,  VY = UP,   ACTION = COLLIDE
```

Final Summary at Snapshot 5-3:

```
============== Global Interpretation: ==============
From    0.0   <Seg 1> to    2.0   <Seg 1>:   295
From    2.0   <Seg 2> to    2.0   <Seg 2>:   105
From    2.0   <Seg 3> to    6.0   <Seg 3>:   101 289 193 316 175 262 157 253
From    6.0   <Seg 4> to    6.0   <Seg 4>:    71
From    6.0   <Seg 5> to    8.0   <Seg 5>:    74 255
From    8.0   <Seg 6> to    8.0   <Seg 6>:    73
=====================================================
Number of property assertions:    18
System Run Time = 30.81 seconds
```

D.2 Noticing Reach-Duration Conflicts

D.2.1 Example 6: Three Collisions – With Duration Constraints

The same behavior is observed here as for Example 5, except that duration constraints are enforced. For the first two collisions, the best working interpretations are identical to those in Example 5. However, after the third collision the best interpretation consistent with the duration estimates is that all three collisions occur inside the well. The interpretation suggested for Example 5 contradicts the domain-specific duration estimates indicating that the ball cannot move from S14 or S78 to S19 in the short period of time between collisions. As this example shows, recovery from a garden-path interpretation, which would typically require much backtracking, can be handled relatively efficiently by dynamically adjusting the pinterp-space.

The first collision is interpreted just as in Example 5, leading to Snapshot 6-1.

Snapshot 6-1: *Through time = 2 seconds*

```
                  11111111122222223333333
          33456677011233445455678901234 45
          08530889528417395109876543210 98      P2    P3    P4
  Seg1: ....#..#..--#----####-#1-------         R     D     F
  SEG2: #1##.##.##.....................         R     D     C
Interpretation credibility = 6.452681775320579d-5
Best interpretation ending in state other than 38 ends in state 105:
   It's credibility (3.970881022884232d-5) is 1.63 times smaller.
-------------------------------------------------------------------------
State 295: PLACE = SREGION2,  VX = RIGHT, VY = DOWN, ACTION = FLY
State  38: PLACE = SEGMENT14, VX = RIGHT, VY = DOWN, ACTION = COLLIDE
-------------------------------------------------------------------------
State 105: PLACE = SEGMENT78, VX = RIGHT, VY = DOWN, ACTION = COLLIDE
```

After the second collision, the pinterp-space is still quite similar to that of Example 5. However, the duration constraints cause many pinterps that are ACTIVE in Snapshot 5-2 to become INACTIVE in Snapshot 6-2. Nevertheless, the best working interpretation is identical for both Snapshot 6-2 and Snapshot 5-2.

Finally, when the third collision is observed, only interpretations where the ball bounces inside the well are consistent with the duration estimates. For example, the domain-specific knowledge indicates that moving from a state where the ball is at S14 to a state where the ball is at S19 requires at least 4 seconds. This bound is derived from the geometry of the system along with a particular maximum energy bound for the ball. Since DURATION(G_5) is only 2 seconds, such movement cannot occur over G_5. Similarly, DURATION(G_4) is only 4 seconds but movements from S78 to S19 directly are known to take at least 13 seconds. Thus, the hidden-transition paths for G_3 or G_5 in Snapshot 5-3 allowing interpretations where the ball is outside of the well are all inconsistent with the duration estimates.

Note that the only alternative to the best interpretation for Snapshot 6-3 is the interpretation starting at S17 instead of inside SRO. However, the domain-specific rules indicate that it is *a priori* more likely for the ball to be at SRO at any given time than at S17.

Snapshot 6-2: *Through time = 6 seconds*

```
                                1111111111111111111111111222222222
          1123334445566677788900011223333444555667889901122344 5
         1739503815836038189395158284713793591573951739517395140     P2     P3     P4
  Seg1: ..............#....-.......--.-.-.-.-.-.-.............#.-      R      D      F
  SEG2: .....-.-.#.-...-.-.....1.-.............................        R      D      C
  Seg3: ----....#..........---1.-...-.-.-.-.-.-.------------#.-.      L      D (F P)
  SEG4: ....-.1...#.#.-.-.....................................        L      D      C
        22222222223333333333333
        55667788899001122344455
        39281706958473625140392 8
  Seg1: .-.-.-.-.-1.-.-.-.-.-.-.-.-
  SEG2: ........................
  Seg3: -.-.-.-.-2.-.-.-.-.-.-.-.-.
  SEG4: ........................
Interpretation credibility = 4.941192348422297d-9
Best interpretation ending in state other than 33 ends in state 56:
   It's credibility (2.8408644649021312d-18) is 1739327028.61 times smaller.
-----------------------------------------------------------------
State 295: PLACE = SREGION2,  VX = RIGHT, VY = DOWN, ACTION = FLY
State 105: PLACE = SEGMENT78, VX = RIGHT, VY = DOWN, ACTION = COLLIDE
State 101: PLACE = SEGMENT78, VX = LEFT,  VY = DOWN, ACTION = FLY
State 289: PLACE = SREGION2,  VX = LEFT,  VY = DOWN, ACTION = FLY
State  33: PLACE = SEGMENT14, VX = LEFT,  VY = DOWN, ACTION = COLLIDE
-----------------------------------------------------------------
State  56: PLACE = SEGMENT17, VX = LEFT,  VY = DOWN, ACTION = COLLIDE
```

Snapshot 6-3: *Status after processing all observations*

```
                                                    111111111111111111
                  1112223333444455556666777778889990000111223333444455
        12783490570358135803680358134893590571358028471379359l5      P2    P3    P4
  Seg1: ......................#.......-.............--.-.-.-.-.-.-     R     D     F
  SEG2: ..........-..-..1..-.....-...-.........-..-.............     R     D     C
  Seg3: -.-.-.-.-.......1.................-.-.-.-..-....-.-.-.-.-.     L     D  (F P)
  SEG4: ........-..-.....1..-..-..-..............................     L     D     C
  Seg5: .-.-.-.-.-.-..-..-..1.....-...-...-.-.-.-..-.............     L     U  (F P)
  SEG6: ......................1.....-...............................     L     U     C
        11111111111111122222222222222222222222222222222233333333333333333333
        55666777888899900011112223334445555666777888899900001112223334444555
        79359157137935915712783490571460359248137026915804793682571460359248
  Seg1: ............................1..-..-..-..-..-..-..-..-..-..-..-..-..-..-
  SEG2: ..........................................................................
  Seg3: -.-.-.-.-.-.-.-.-.-.-.-.-.-.2..-..-..-..-..-..-..-..-..-..-..-..-..-..
  SEG4: ..........................................................................
  Seg5: .-.-.-.-.-.-.-.-.-.-.-.-.-.-.2..-..-..-..-..-..-..-..-..-..-..-..-..-.
  SEG6: ..........................................................................
Interpretation credibility = 1.9428988031530713d-22
No interpretation ends in a different state!
-------------------------------------------------------------------
State 241: PLACE = SREGIONO,  VX = RIGHT, VY = DOWN, ACTION = FLY
State  45: PLACE = SEGMENT15, VX = RIGHT, VY = DOWN, ACTION = COLLIDE
State  41: PLACE = SEGMENT15, VX = LEFT,  VY = DOWN, ACTION = FLY
State 235: PLACE = SREGIONO,  VX = LEFT,  VY = DOWN, ACTION = FLY
State  48: PLACE = SEGMENT16, VX = LEFT,  VY = DOWN, ACTION = COLLIDE
State  50: PLACE = SEGMENT16, VX = LEFT,  VY = UP,   ACTION = FLY
State 237: PLACE = SREGIONO,  VX = LEFT,  VY = UP,   ACTION = FLY
State  58: PLACE = SEGMENT17, VX = LEFT,  VY = UP,   ACTION = COLLIDE
-------------------------------------------------------------------
State  60: PLACE = SEGMENT17, VX = RIGHT, VY = DOWN, ACTION = FLY
```

Final Summary at Snapshot 6-3:

```
=============== Global Interpretation: ===============
From    0.0   <Seg 1> to    2.0   <Seg 1>:   241
From    2.0   <Seg 2> to    2.0   <Seg 2>:    45
From    2.0   <Seg 3> to    6.0   <Seg 3>:    41 235
From    6.0   <Seg 4> to    6.0   <Seg 4>:    48
From    6.0   <Seg 5> to    8.0   <Seg 5>:    50 237
From    8.0   <Seg 6> to    8.0   <Seg 6>:    58
======================================================
Number of property assertions:    18
System Run Time = 30.84 seconds
```

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-90-1572	2.	3. Recipient's Accession No.

4. Title and Subtitle

DYNAMIC ACROSS-TIME MEASUREMENT INTERPRETATION: MAINTAINING QUALITATIVE UNDERSTANDINGS OF PHYSICAL SYSTEM BEHAVIOR

5. Report Date: February 1990

6.

7. Author(s)

Dennis Martin DeCoste

8. Performing Organization Rept. No. R-90-1572

9. Performing Organization Name and Address

Dept. of Computer Science
University of Illinois
1304 W. Springfield Ave.
Urbana, IL 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.

N00014 85 K 0225

12. Sponsoring Organization Name and Address

Office of Naval Research

13. Type of Report & Period Covered

Technical

14.

15. Supplementary Notes

16. Abstracts

Incrementally maintaining a qualitative understanding of physical system behavior based on observations is crucial to real-time process monitoring, control, and diagnosis. This paper describes the DATMI theory for dynamically maintaining a *pinterp-space*, a concise representation of local and global interpretations consistent with the observations over time. Each interpretation signifies alternative paths of states in a qualitative envisionment. Representing a space of interpretations, instead of just a "current best" one, avoids the need for extensive backtracking to handle incomplete or faulty data. Domain-specific knowledge about state and transition probabilities can be used to maintain the best working interpretation as well. Domain-specific knowledge about durations of states and paths of states can also be used to further constrain the interpretation space. When all these constraints lead to inconsistencies, faulty-data hypotheses are generated and then tested by adjusting the pinterp-space. The time and space complexity of maintaining the pinterp-space is polynomial in the number of measurements and envisionment states.

17. Key Words and Document Analysis. 17a. Descriptors

artificial intelligence
qualitative reasoning
measurement interpretation
explanation
monitoring

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
unlimited	UNCLASSIFIED	130
	20. Security Class (This Page) UNCLASSIFIED	22. Price