Order Number 9503174

# Goal-directed qualitative reasoning with partial states

Decoste, Dennis Martin, Ph.D.

University of Illinois at Urbana-Champaign, 1994



U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

# GOAL-DIRECTED QUALITATIVE REASONING WITH PARTIAL STATES

BY

DENNIS MARTIN DECOSTE

B.S., University of Illinois at Urbana-Champaign, 1986
M.S., University of Illinois at Urbana-Champaign, 1989

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1994

Urbana, Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

FEBRUARY 1994

WE HEREBY RECOMMEND THAT THE THESIS BY

DENNIS MARTIN DECOSTE

ENTITLED GOAL-DIRECTED QUALITATIVE REASONING WITH PARTIAL STATES

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF DOCTOR OF PHILOSOPHY

Director of Thesis Research

Head of Department

Committee on Final Examination†

Chairperson

† Required for doctor's degree but not for master's.

0-517

# GOAL-DIRECTED QUALITATIVE REASONING WITH PARTIAL STATES

**Dennis Martin DeCoste, Ph.D.**
**Department of Computer Science**
**University of Illinois at Urbana-Champaign, 1994**
**Kenneth D. Forbus, Advisor**

This research explores the representational and computational complexities of qualitative reasoning about time-varying behavior. Traditional techniques employ qualitative simulation (QS) to compute envisionments (i.e. state-transition graphs) representing all possible behaviors. Unfortunately, QS exhaustively case-splits on all choices, regardless of specific task goals. It reasons with completely described states and explores every (ambiguous) future of each.

In this thesis we introduce a new representation, called *sufficient discriminatory envisionments* (SUDE's), which addresses these problems. SUDE's discriminate the possible behavior space by whether the goal is possible, impossible, or inevitable from each state in that space. Our techniques for generating SUDE's strive to reason with the smallest state descriptions which are sufficient for making these discriminations.

We present algorithms for generating SUDE's via a two-stage process. First, exhaustive regression *sketches* the space of possible paths between the initial and goal states. Second, we *qualify* these possible paths, identifying conditions under which the goal is impossible or inevitable and finding all possible transitions between these paths.

We formulate Nature's regression operators in terms of minimal chunks of causality, exploiting the causal, compositional nature of Qualitative Process Theory models. We integrate continuity-based and minimality-based theories of change to support discontinuous change due to actions and modelling simplifications.

We discuss our implementation of these techniques and our test examples in three domains, which we call ball-world, tank-world, and kitchen-world.

To Chui and Matthew ...

# Acknowledgments

Foremost, I thank my advisor, Ken Forbus, for his encouragement and for providing an exciting and productive environment for research. I will be forever grateful for his confidence and patience at critical times.

Thanks also go to Jerry DeJong, Caroline Hayes, Seth Hutchinson, and Marianne Winslett, for serving on my committee and providing useful suggestions.

I also extend sincere thanks to other past and present members of the Qualitative Reasoning Group, for many enlightening and entertaining discussions and for graciously enduring my frequent ramblings. I especially thank those who have suffered the most — John Collins and Gordon Skorstad.

I am grateful for the comradery and ideas provided by my UIUC officemates, particularly Dan Oblinger, Jon Gratch, David Page, Melinda Gervasio, Scott Bennett, and Steve Chien, and by my ILS officemates, particularly Chris Lopez and Nikitas Sgouros.

Of course, much thanks must also go to BananaJr and Marvin, the trusty old IBM RT and speedy new IBM RS/6000 on which most of this research was performed.

Thanks to NASA Langley Research Center, and Kathy Abbott in particular, for funding my thesis research.

Many thanks also go to Richard Doyle at the NASA Jet Propulsion Laboratory, for his patience during the completion of this thesis and for providing an exciting AI research group at JPL to move on to.

Deep thanks go to my parents and brothers for their love and support over the years.

Special thanks go to baby Matthew, for offering me fascinating glimpses of a new intelligent problem solver in the making.

And, finally, I offer infinite thanks to my wife, Chuigang, for her love, support, understanding — and patience — throughout this whole ordeal.

# Table of Contents

**Chapter**


**1 Introduction** . . . . . 1

1.1 Qualitative Physics . . . . . 1

1.2 The Standard Approach: Qualitative Simulation . . . . . 2

1.3 The Problem: Envisionments Are Excessive . . . . . 4

1.3.1 Cause 1: Considering All Completions . . . . . 5

1.3.2 Cause 2: Considering All Nexts . . . . . 5

1.3.3 Underlying Issues . . . . . 7

1.3.4 An Analogy With Logical Resolution . . . . . 10

1.4 Types of Qualitative Reasoning . . . . . 11

1.5 A New Perspective Is Required . . . . . 14

1.6 Our New Approach: Discriminatory Envisionments . . . . . 14

1.6.1 AE-Based Concise DE's . . . . . 15

1.6.2 Causal-Based Sufficient DE's . . . . . 16

1.7 Example SUDE's . . . . . 17

1.7.1 Example 1: Basic Ideas . . . . . 18

1.7.2 Example 2: No Explicit Goal . . . . . 24

1.7.3 Example 3: Other Issues . . . . . 25

1.8 Road Map . . . . . 28

**2 Representing Qualitative States** . . . . . 30

2.1 Qualitative Process Theory . . . . . 31

2.1.1 Quantity and Derivative Inequalities . . . . . 31

2.1.2 Functional Relations . . . . . 32
2.1.3 Influences . . . . . 33
2.1.4 Processes and Views . . . . . 33
2.1.5 Domain Theories Versus Scenario Models . . . . . 34
2.2 The Qualitative Law of Diminishing Returns . . . . . 37
2.3 Types of State Distinctions . . . . . 38
2.3.1 Process/View Conditions . . . . . 39
2.3.2 Derivative Conditions . . . . . 40
2.3.3 Rate Relations . . . . . 42
2.3.4 Duration . . . . . 49
2.4 Types of States . . . . . 49
2.5 Reasoning About Closures of Partial States . . . . . 51
2.5.1 The Sufficiency of Closures for Computing Next States . . . . . 52
2.5.2 The Adequacy of Assumption Closures . . . . . 53
2.5.3 Computing Assumption Closures Via Completions . . . . . 54
2.5.4 Approximating Closures Via Boolean Constraint Propagation . . . . . 55
2.5.5 Exploiting the Nature of QPT Models . . . . . 58

**3 Computing State Transitions** . . . . . 65
3.1 Continuity-Based Change . . . . . 66
3.1.1 Base Temporal Constraints . . . . . 67
3.1.2 Temporal Constraints Facilitate Partial State Reasoning . . . . . 68
3.1.3 Action-Augmented Envisionments . . . . . 70
3.2 Problems With Continuity-Based Approaches . . . . . 72
3.2.1 Discontinuity . . . . . 72
3.2.2 Coincidences . . . . . 75
3.3 Combining Minimality and Continuity-Based Change . . . . . 78
3.3.1 Correcting the Continuity-Based Algorithms . . . . . 78
3.3.2 Handling the Pump-Cycle Sudden-Pump Example . . . . . 85
3.4 Comparison to Minimality-Based Approaches . . . . . 87
3.4.1 Possible Worlds Versus Possible Models . . . . . 88

3.4.2 PMA Update Using DNF States . . . . . 89
3.4.3 Justification-Based Measures . . . . . 92
3.4.4 Priority-Based Measures of Minimality . . . . . 94
3.4.5 Set-Inclusion Versus Cardinality Measures of Closeness . . . . . 96

**4 Finding State Paths** . . . . . 98
4.1 Suitability of Regression for Qualitative Reasoning . . . . . 98
4.2 Types of Regression Operators . . . . . 99
4.2.1 Action Operators . . . . . 99
4.2.2 Dynamics Operators . . . . . 100
4.3 Formulating Dynamic Operators . . . . . 102
4.3.1 Sufficient Support States for Derivative Inequalities . . . . . 102
4.3.2 Computing Complete Sets of Sufficient Support States . . . . . 105
4.3.3 Using Sufficient Support States to Formulate Dynamic Operators . . . . . 106
4.3.4 Using Dynamic Operators to Assess QP Models . . . . . 112
4.4 Goal-Based Regression . . . . . 112
4.4.1 Conjunctive Subgoal Conflicts Seem Rare . . . . . 115
4.4.2 Types of Subgoals . . . . . 116
4.5 Special Issues . . . . . 117
4.5.1 Incomplete Initial State . . . . . 117
4.5.2 The Need For Simultaneous Dynamic Operators . . . . . 118
4.5.3 Allowing Instant-Instant Transitions . . . . . 119
4.5.4 Avoiding Forced Clobberings . . . . . 119

**5 Qualifying State Paths** . . . . . 122
5.1 Impossibility . . . . . 123
5.1.1 Extending REGRESS to Compute Seed Impossibility Sets . . . . . 125
5.1.2 Determining Which Unknowns to Falsify . . . . . 127
5.2 Falling Off Regression Paths . . . . . 128
5.2.1 Propagating Impossibility . . . . . 128
5.2.2 Ensuring Soundness of Impossibility Sets . . . . . 131
5.2.3 The Significance of Impossibility Sets . . . . . 132

**6 Conclusions** . . . . . 133
6.1 Summary of Contributions . . . . . 133
6.1.1 Towards Sufficiency Conditions for QR . . . . . 133
6.1.2 Turning the Weakness Of QP Constraints Into A Strength . . . . . 134
6.1.3 Integrating Theories of Minimal and Continuous Change . . . . . 134
6.1.4 Integrating QR With Planning . . . . . 134
6.2 Challenges to the QR Community . . . . . 135
6.2.1 The Suitability of Envisionments . . . . . 135
6.2.2 The Adequacy of Bottom-Up QR . . . . . 136
6.3 Summary of Limitations . . . . . 137
6.3.1 Reliance on Causal Models . . . . . 137
6.3.2 Qualitative Constraints Are Very Weak . . . . . 137
6.4 Related Work . . . . . 138
6.4.1 Temporal Reasoning . . . . . 138
6.4.2 Planning . . . . . 139
6.4.3 Nonmonotonic Reasoning . . . . . 139
6.4.4 Modal Logics . . . . . 141
6.5 Future Work . . . . . 143
6.5.1 Formal Theories of SUDE's . . . . . 143
6.5.2 Comparative Analysis and Relative Probability . . . . . 143
6.5.3 Better Control Strategies . . . . . 144

**References** . . . . . 145

**Vita** . . . . . 151

# List of Tables

2.1 "Hard" vs "soft" inequality representations . . . . . 32

3.1 Temporal constraints on the next state . . . . . 68
3.2 Temporal constraints on the previous state . . . . . 68
3.3 Temporal constraints on the current state . . . . . 69
3.4 Summary of states for pump-cycle sudden-pump example . . . . . 87

# List of Figures


1.1 Generic qualitative simulation algorithm . . . . . 3
1.2 The ball-world scenario, with four ledges and one ball . . . . . 18
1.3 SUDE1 . . . . . 19
1.4 Example qualitative dynamic operator (OP1) . . . . . 21
1.5 SUDE2 . . . . . 26

2.1 Key definitions for fluid free-flow processes . . . . . 35
2.2 Key definitions for fluid pumped-flow processes . . . . . 36
2.3 Simple pump-cycle scenario . . . . . 36
2.4 The pump-cycle system in equilibrium . . . . . 37
2.5 A four-pump, three-container system . . . . . 46
2.6 The two-pump, three-container, pump-cycle system in equilibrium . . . . . 47
2.7 Scenario `CAN-WITH-SIDE-PORT` and state S1 . . . . . 59
2.8 Some scenario model constraints for `CAN-WITH-SIDE-PORT` . . . . . 60
2.9 `Container` defEntity augmented with "causality skipping" constraints . . . . . 60
2.10 Scenario `CAN-WITH-WATER-AND-OIL` and state `S2` . . . . . 63

4.1 Example action operator . . . . . 99
4.2 Simple action operator to flip preconditions . . . . . 100
4.3 Generic divergence operator . . . . . 101
4.4 Generic convergence operator . . . . . 101
4.5 Possible derivative rules . . . . . 104
4.6 Qprop chain rules . . . . . 105
4.7 Stage 1: Instantiating the generic convergence operator for P . . . . . 109

4.8 Stage 2: Inserting sufficient support state (before closure) . . . 109
4.9 Stage 3: After conditions closure and forward temporal constraints . . . 110
4.10 Stage 4: After effects closure and new change checks . . . 111
4.11 Stage 5: Final operator for P (minimized conditions/effects) . . . 111
4.12 Procedure FIND-PATHS . . . 113
4.13 Procedure REGRESS . . . 113
4.14 Procedure REGRESS-TO-FLIP . . . 114
4.15 Procedure HANDLE-DISCONTINUOUS-OPS . . . 114
4.16 Summary of functions referenced by calls to FIND-PATHS . . . 115
4.17 Example of a Discontinuous Dynamics Operator for I and G . . . 116
4.18 Four Types of Subgoals . . . 117
4.19 Handling the multiple emptying containers example . . . 120

5.1 Impossibles for the Overflow-Can3 example . . . 124
5.2 Graphical depiction of the Overflow-Can3 example . . . 124
5.3 Impossibles for the Overflow-Can2 example . . . 125
5.4 Procedure HANDLE-UNKNOWNS . . . 126
5.5 Procedure UNKNOWNS-TO-NEG-FOR-I . . . 127
5.6 Example of selecting the proper I refinements . . . 128
5.7 Procedure GATHER-IMPOSSIBILITY . . . 129
5.8 Procedure UPDATE-IMPOSSIBLES . . . 130
5.9 Some definitions used by Procedure UPDATE-IMPOSSIBLES . . . 131

6.1 Hanks-like operator for a flow process . . . 140

# Chapter 1

# Introduction

The area of *qualitative physics* (QP) has enjoyed explosive growth over the last ten years or so [52]. However, despite this growth, relatively little attention has been given to a fundamental issue: the computational complexity of reasoning about qualitatively-ambiguous behavior over time. Instead, the theoretical emphasis to date has been on soundness and completeness issues. Although that emphasis may have been necessary for defining this relatively new field, it has lead to a somewhat ironic result, given that many of the original motivations of QP arose from the area of commonsense reasoning. Specifically, current QP algorithms typically do not make obvious qualitative conclusions about a particular system more quickly than less obvious ones. In other words, the complexity of QP algorithms tend to be far more dependent on the complexity of the particular model of the physical system than on the specific task being performed. This thesis is a direct attempt to define and address this problem.

## 1.1 Qualitative Physics

A defining characteristic of qualitative physics is the discretization of continuous parameters based on physical limit points such as boiling points, zero masses, and zero derivatives. Such discretization is useful because it partitions phase space into regions representing qualitatively different behavior. Reasoning with these regions instead of exact points in phase space (as traditional numeric simulation would), results in weaker but broader conclusions.

This broad-but-weak nature of QP can be appropriate in several common contexts. In earlier stages of design or diagnosis, QP can help systematically eliminate large sets of physical systems

which are all qualitatively equivalent in ways that make them all unsuitable candidates. When the data available for monitoring or prediction tasks are incomplete, QP still allows some useful (though weak) conclusions to be made, based on the qualitative regions covering those data. Furthermore, QP supports meaningful, causal-based explanations of why particular behaviors may occur.

For the most part, previous QP algorithms have employed static partitionings of phase space. Thus, if the comparison between a quantity and a limit point is made for some time point, it is made for all time points. There are two good reasons for taking that approach. First, it facilitates comparisons between the inferences made with QP and those made with numeric simulation (which employs the most extreme case of static partitioning: exact points). Second, it leads to relatively simple inference algorithms. Although these features have proven useful in studying soundness and completeness issues, static partitioning typically results in excessive computational and representational complexities. In order to explore this issue, we first summarize the nature of standard QP algorithms below.

## 1.2 The Standard Approach: Qualitative Simulation

Most QP systems implemented to date have relied on *qualitative simulation* (QS) to make their inferences about behavior over time. QS takes a qualitative model and a description of the initial state and produces a directed state-transition graph, called an *envisionment.* Each path through the envisionment qualitatively summarizes a set of alternative possible quantitative behaviors over time.

Figure 1.1 outlines the fundamental algorithm of all standard qualitative simulators, such as `ENVISION` [12], `GIZMO` [23], `QPE` [27], and `QSIM` [39]. It is said to generate a *total envisionment* (TE) if the initial state description $\mathcal{I}$ is empty; otherwise it is said to generate an *attainable envisionment* (AE). For a given model, there is exactly one AE for each $\mathcal{I}$. denoted AE($\mathcal{I}$), representing the subgraph of the TE which may occur during or after $\mathcal{I}$.

The *completions* of a state description correspond to the set of logical models (or interpretations) containing that description. Thus, each completion corresponds to one region in the staticly partitioned phase space and no such region subsumes another. Each completion is often called a complete (or total) state.

Procedure **QualSim**($\mathcal{I}$):

$Q \leftarrow$ **Completions**($\mathcal{I}$).
*States* $\leftarrow \emptyset$.
While $Q \neq \emptyset$ do :
  $S \leftarrow \text{Pop}(Q)$.
  *States* $\leftarrow$ *States* $\cup \{S\}$.
  *Nexts*$(S) \leftarrow$ **NextStates**$(S)$.
  $Q \leftarrow Q \cup$ *Nexts*$(S) -$ *States*.

**Figure 1.1**: Generic qualitative simulation algorithm

The *nexts* of a complete state $S$ are intended to correspond to a subset of the set of complete states, representing the qualitative regions in phase space that are temporally adjacent to $S$. More precisely, for each next state $N$ of $S$, there is at least one point in $S$'s region and one point in $N$'s region such that the phase space trajectory between those two points starts in $S$, ends in $N$, and involves no state transition other than the one from $S$ to $N$.

The heart of early work on QS was to generate sound and complete sets of nexts. From a soundness/completeness perspective, the main problem with this framework is that global soundness is not guaranteed. Consider a path of three complete states $S_1 \rightarrow S_2 \rightarrow S_3$. Let $p_1$ be a point in the phase space region represented by $S_1$, let $p_3$ be similar for $S_3$ and let $p_i$ and $p_j$ be similar for $S_2$. Let these points be further constrained so that the exact trajectory through phase space from $p_1$ to $p_i$ starts in $S_1$'s region and ends in $S_2$ region and is always in one of those two regions. Similarly constrain $p_j$ and $p_2$ in terms of the regions of $S_2$ and $S_3$. Soundness of nexts guarantees that some such pair $(p_1, p_i)$ must exist for QS to generate the transition $S_1 \rightarrow S_2$ (and similarly for pair $(p_j, p_3)$ and transition $S_2 \rightarrow S_3$). Unsoundness arises when there is in fact no trajectory through the region of $S_2$ which begins at $p_i$ and ends at $p_j$.

This potential unsoundness should not be surprising. It is an inherent property of the envisionment representation, due to its inability to represent non-local constraints. In principle, global unsoundness can always be avoided by refining the partitioning to include new limit point comparisons for which local constraints will adequately reflect the global constraints. Unfortunately, such refinement can, in general, require an infinite number of new comparisons,

essentially to subdivide critical regions (such as that of state $S_2$ above) as close to exact points as necessary.

Despite theoretical concerns, we claim that global unsoundness is typically not of prime pragmatic concern for most tasks, for two key reasons. First, it appears to be rather rare. In fact, it seems to arise as special cases, such as energy constraints [30] and feedback structures [53], that can often be handled by other means. Second, and more importantly, most tasks for which we would employ QS are more concerned with completeness than soundness. Many tasks, such as monitoring, have a conservative bias, preferring false positives to false negatives. For most other tasks, such as diagnosis and design, global unsoundness generally can be detected at later (verification) stages, thus incurring costs (i.e. backtracking) only in those rare cases where unsoundness actually arises.

The main reason any task employs QS is to be able to reason under uncertainty by reasoning with a set of alternative behaviors. The user (implicitly) knows that only one of those behaviors is the "real" one. Thus, whether the others are globally *consistent* is not the primary concern. Instead, it is that the *size* of the set of alternatives is manageable in the first place. Which brings us back to the fundamental concern of our work: the computational and representational complexity of QS.

Based on the above rationale, our work does not address the global unsoundness inherent in envisionments. Our only concern with soundness will be to avoid introducing unsoundness as we move toward weaker, more manageable representations. Thus, whereas the "gold standard" for qualitative simulation has typically been differential equations, the gold standard for our work will be envisionments themselves.

## 1.3 The Problem: Envisionments Are Excessive

One would like the complexity of performing a task to reflect the complexity of that specific task, not the complexity of the physical system per se. Unfortunately, QS tends to reflect the inherent ambiguity in the qualitative model more than task-specific constraints. For example, it has been postulated that even an attainable envisionment based on a detailed qualitative model of a nuclear power plant might consist of more states than there are atoms in the universe. For

many tasks it is not necessary, desirable, nor even possible to explicitly enumerate the entire envisionment space.

Such excessive complexity results from the two types of case-splits that QS makes, to represent uncertainty in both the current and the next states. We consider each type in turn below.

### 1.3.1 Cause 1: Considering All Completions

Requiring all completions of the initial state $\mathcal{I}$ can be a major source of complexity. A glance at the relatively simple examples explored in most of the QP literature might suggest that this might not be such a problem. However, for large real-world systems one is likely to observe only a small fraction of the initial state of that system. Thus, QS would require case-splitting on all consistent combinations of the initial statuses of all unobserved system parameters. Furthermore, when considering events which can change the connectivity of the system — like opening and closing valves — almost any state could potentially lead to any other, given enough time. For such cases, an initial state alone does not even provide much constraint on the envisionment that must be generated; the attainable envisionment will be nearly as large as the total envisionment.

### 1.3.2 Cause 2: Considering All Nexts

Considering all next states of each state, each at complete levels of detail themselves is the other major source of complexity in QS. Branching can occur whenever there are multiple quantities moving toward limit points but some of their relative derivatives and relative distances from those limit points are unknown. Without those relative constraints, it is usually ambiguous which quantity will converge to its limit point first; thus, QS will often branch on all possible orderings. In fact, there are only two types of exceptions:

1. one convergence must occur before another because of transitivity constraints and
2. two must occur together due to correspondence constraints.

Consider the now classic QP two-container example: two containers of water connected at their bottoms by a horizontal pipe. Let $B_i, L_i$, and $P_i$ refer respectively to container $i$'s bottom height, water level height, and bottom pressure. Imagine a state $\mathcal{I}$ where $L_1$ is higher than $L_2$.

That causes $P_1 > P_2$, which causes a flow of water from container 1 to container 2, which causes $L_1$ to drop and $L_2$ to rise. An example of exception type 1 is that $L_1$ equalizes with $L_2$ before $L_1$ reaches $B_1$. Based on domain knowledge $L_2 \geq B_2$ and $B_2 = B_1$, the transitive relation $L_1 > L_2 > B_1$ holds in $\mathcal{I}$. An example of type 2 is that $P_1$ equalizes with $P_2$ when $L_1$ equalizes with $L_2$. Domain knowledge indicates that the relation between $P_1$ and $P_2$ corresponds to the relation between $L1$ and $L_2$.

QS has the especially undesirable property that two causally unrelated possible convergences will necessarily result in branching on which occurs first. Thus, QS does not gracefully scale-up when reasoning about the most common real-world systems: large, loosely-coupled systems (e.g. the common every-day world, automobiles, airplanes, power plants, and so on). Furthermore, even when convergences are related, it is often the case that some possible orderings will have the same result at some point in the future. Consider, for example, many tanks of water connected in series by pipes. Regardless of which two neighboring tanks equalize first, eventually all tanks will have equal water levels. This is an instance of a general problem called *occurrence branching*. It was first explored in some detail in the context of post-processing envisionments to identify occurrence branchings [41]. However, that approach fails when the envisionment cannot be computed in the first place. In fact, we claim that this is typically the case as one scales up to reason about real physical systems.

Branching is extremely common in QS because the necessary relative constraints on derivatives and distances are often either insufficient, unmanageable, or even unrepresentable. For example, consider a quantity $Q_1$ at a limit point $V_1$ and converging up toward a limit point $L_1$ and similarly consider $Q_2$ at $V_2$ and converging up toward $L_2$. Further assume that the relative distance relation $L_1 - V1 < L_2 - V_2$ always holds. If one can guarantee that relative derivative relation $D(Q_1) \geq D(Q_2)$ will hold at least until either $Q_1$ or $Q_2$ converges, then one can indeed infer that $Q_1$ will reach $L_1$ before $Q_2$ reaches $L_2$. In contrast, knowing $D(Q_1) > D(Q_2)$ and $L_1 - Q_1 > L_2 - Q_2$ (via assuming $L_1 - V_1 > L_2 - V_2$ instead) is insufficient: $Q_1$ is moving faster than $Q_2$ but also has farther to go. Thus, the use of relative relations cannot always prevent branching in QS. Order of magnitude reasoning can help; for example, one can infer that $Q_1$ converges before $Q_2$ does if $D(Q_1)$ is much larger than $D(Q_2)$ and $L_1 - V_1$ is only a little larger than $L_2 - V_2$. However, that only pushes the problem to the higher order relations, for which branching can still be a problem. In general, the level of detail necessary to always

avoid branching is simply unrepresentable as a finite number of qualitative constraints. Furthermore, relative relations are only representable if the units of the derivative and the distance are compatible for both converging quantities. For example, if a temperature and a pressure are converging to respective landmarks, relative relations cannot be used to constrain one to happen first.

Relative constraints are sufficient to infer that one of two possible convergences occurs first only if two conditions are meet:

1. the relative derivative relation and the relative distance relation are not in the same direction,

2. the relative derivative relation holds at least until one of the convergences occurs.

Unfortunately, even when the first condition happens to be satisfied, the second condition is often difficult to guarantee. Essentially, the problem is that the relative derivative relation is itself often convergencing toward the derivatives becoming equal (at least momentarily). Just as order of magnitudes can provide higher-order constraint on the first condition, higher order derivatives can provide higher-order constraint on the second condition. However, again, this only pushes the underlying problem to higher levels. We explore such underlying problems in the next section.

### 1.3.3 Underlying Issues

Many researchers have suggested (both implicitly and explicitly) that the above complexities are inherent to QP, due to the ambiguous, weak nature of qualitative constraints. Indeed, that perspective has driven much recent work on hybrid systems which merge symbolic, numeric, and qualitative constraints, such as MINIMA [55] and Q2 [40]. However, we argue that there are two other issues which have been largely overlooked in previous work, and yet which can significantly impact computational complexity. We consider those two issues in turn below. In short, we claim that the bulk of the complexity is often more related to the nature of QS than the nature of QP.

#### 1.3.3.1 Adding Qualitative Detail Can Be Harmful

We alluded to a problem of infinite regress in the previous section on branching. Tracking higher-order relations over time in order to better constrain lower-order relations can easily result in higher complexity, not less, because of newly introduced ambiguities in how the higher-order relations themselves change over time. We will explore this phenomena in great detail in later chapters, as it plays a fundamental role in the complexity of any QP approach. For now, we will simply summarize our intuitions. We claim that it is generally sufficient (for soundness), and much more efficient, to include such higher-order relations in a state description only when they are inferable from existing constraints for that state or have necessarily persisted from a previous state. This avoids the complexity that QS suffers from having statically partitioned phase space so that every state makes a commitment to the status of every relation that one might wish to track over time. Its promise is based on the intuition that the deep inherent ambiguity in qualitative constraints usually makes it foolish to try to reduce lower-order case-splits by introducing higher-order case-splits.

#### 1.3.3.2 Goal States Provide Exceptional Constraint for QP

While qualitative ambiguity appears to minimize the utility of higher-order detail, it conversely appears to maximize the utility of goal states. Because of the weakness of qualitative constraints, there is typically at least one path through an envisionment connecting some completion of a given initial state $\mathcal{I}$ with some completion of a given goal state $\mathcal{G}$, as long as there is some conceivable way that this might occur. Specifically, such paths often exist as long as there are some trees of causal directed influences over time which relate each quantity in $\mathcal{G}$ back to quantities in $\mathcal{I}$. If generating envisionments via QS is so complex due to ambiguity and yet the same goal achievability results can usually be determined via goal-directed search through influence structure, then the appropriateness of QS comes into question. After all, most real tasks are fundamentally concerned about the achievability of goals.

For example, if a ball is thrown to the right, then qualitatively that ball might eventually hit any target on the right. Try to imagine scenarios in which the qualitative constraints alone are sufficient to indicate that certain targets on the right are unreachable, and yet there is a path

of influences relating $\mathcal{G}$ to $\mathcal{I}$. In other words, consider how influence search might be unsound, relative to envisionments.

One scenario is that there is a force field between the ball and a given target which is stronger than the force due to the ball's movement. However, qualitative ambiguity would also allow that that field might not be strong enough, so any target must be considered potentially reachable. So, consider an infinite force opposing the ball, say due to a tall wall that blocks the ball from some targets. However, the absoluteness of that force makes this case also easy to handle. One need only locally check that for some set of candidate influences (for moving a quantity Q between two values A and B) there is no opposing set of influences on Q which always dominate when Q is at some value between A and B.

Another scenario is that the ball is on a short, unbreakable tether line that would abruptly stop the flying ball before it reached some target T. Realizing that target T is unreachable is trivial, for any reasonable qualitative model of that tether. Any goal specifying that the ball is at T would be inconsistent, because it would require the tether length being longer than its limit.

A more subtle scenario would be if there was some triggering device past T which would release the tether if the ball hit that device. Here, the ball being farther from its initial location than the tether length would not be universally inconsistent, since the tether could be in its released status. However, even in this case, searching influence structure would be sufficient to realize that the ball will never reach T, if the tether was initially unreleased. To influence the ball all the way from its initial location to T (via flying) would require the tether to be released. However, releasing the tether requires the ball reaching the trigger, which itself requires the tether to be released. This deadlock makes the goal unreachable.

These results may appear surprising, particularly from the perspective of general planning frameworks. Consider a planning framework where dynamical processes (like flows or flying) are modelled as STRIPS-like operators [22] available to a special agent called Nature. The long-term dynamical effects of an event (e.g. a ball hitting a trigger) will be due to a sequence of Nature operators, not simply the add and delete lists of one action or Nature operator. So, it might appear that validation (via QS) would often be required to ensure that all possible dynamical effects are considered when predicting goal achievability. However, the point of the above exercise is to suggest that qualitative constraints are typically so weak that the resulting

ambiguity makes predicting goal achievability, *to the same degree of precision as QS would obtain*, possible without such validation.

Nevertheless, QS can still be useful in showing, for example, that a tether might have to break if a ball is to reach a far away target. Indeed, one of the underlying motivations of QP is to provide such explanations, as opposed to simply making predictions. However, even then, it is often the case that the qualitative behavior which explains whether a goal can be achieved occurs relatively near to the goal occurrence itself. For example, a closed container might soon explode if pressure is increasing due to material flow into it from another container. The outcome is ambiguous because the flow might stop first (before the pressure gets too high). Note that this explanation indicates that the explosion is possible up until the very last qualitative change occurs. Generally, the critical point, after which all futures either lead to the goal or never lead to the goal, is one of the last states before the states consistent with the goal itself. Once again, this phenomena is due to qualitative ambiguity.

Although QS has been used in several task frameworks, such as diagnosis and monitoring [20, 16], existing QS algorithms do not accept goal states as inputs. As a consequence, such task frameworks are intractable when the envisionments are too complex to compute, even though in the end only a small portion of the envisionment space would actually be searched for the specific task.

### 1.3.4 An Analogy With Logical Resolution

Toward removing the excesses in QS, it is illuminating to consider an analogy between QS and logical resolution. In order to ensure completeness, resolution branches on case-splits represented by clausal disjunctions. Similarly, to ensure its completeness, QS branches on case-splits represented by completions and nexts. However, whereas efficient resolution algorithms seldom explicitly search all case-splits, QS algorithms always do exactly that.

Certainly, some resolution efficiency comes from being goal-directed; our emphasis on goal-directed qualitative reasoning promises to yield similar efficiencies. However, other efficiencies in resolution come from making simplifications in what degree of completeness is acceptable. For example, unit resolution is very efficient but is only complete for Horn clauses. Analogous simplifications on QS might similarly be useful.

Since QS and resolution do differ in what they compute, it is important to understand when those differences might justify the more exhaustive nature of QS. Resolution is used to prove satisfiability and, less typically, to generate all prime implicates. QS analogs to these must reflect the additional dimension of time. A natural QS analog to satisfiability is whether every envisionment path starting at every completion of $\mathcal{I}$ contains some completion of $\mathcal{G}$. We will call this *QS satisfiability*. Based on that analog, the QS analog to prime implicates is a collapsed envisionment structure that minimizes case-splits while still being sufficient for QS satisfiability. We call such structures *prime envisionments*.

Those analogs suggest two interesting computational issues:

1. Transforming an AE envisionment $\mathcal{E}$ into a prime envisionment $\mathcal{E}'$.

2. Directly computing $\mathcal{E}'$, without having to compute $\mathcal{E}$ first.

Understanding these issues will help us better understand how excessive QS is for QS satisfiability problems.

An obvious transformation to consider is one in which the next of each state $S$ in $\mathcal{E}'$ represents the intersection of the state descriptions of the nexts of all completions of $S$ in $\mathcal{E}$. Starting with $S$ being $\mathcal{I}$, this transformation yields a chain of successively more partial states, which eventually subsumes $\mathcal{G}$ if $\mathcal{G}$ is QS satisfiable. This represents a most extreme type of $\mathcal{E}'$, where there are no case-splits at all.

Although such $\mathcal{E}'$'s would be complete, they would typically be very unsound for QS satisfiability. In fact, due to accumulated ambiguity, states in $\mathcal{E}'$ just a few transitions away from $\mathcal{I}$ would typically be "static", containing only those facts which always persist after $\mathcal{I}$. The problem is that some distinctions among nexts in $\mathcal{E}$ represent important disjunctions that must be temporally projected to preserve the soundness of $\mathcal{E}$. Trying to solve this problem, while also striving to keep as close to the $\mathcal{E}'$ formulation as possible, is a major theme of this thesis.

## 1.4 Types of Qualitative Reasoning

At this point, it should be becoming clear that a key difficulty in assessing the complexity of QS is that it is actually not clear what the output requirements for reasoning about time-varying qualitative behavior really are. This issue has hardly been addressed since the early days of

QP research, from which the envisionment-centered mindset originated. The overwhelming majority of previous work on QP algorithms can be viewed as either generating envisionments (i.e. QS) or transforming envisionments into task-specific outputs. For tasks where QS is prohibitively expensive, a steady-state assumption is often employed, where $\mathcal{I}$ specifies that all derivatives are zero. That reduces the complexity to only that of computing all completions of $\mathcal{I}$. However, rather than resorting to limiting simplifying assumptions to make QS manageable, it is often more appropriate to use an approach which directly provides the output that is really desired.

We will use the term "qualitative reasoning" (QR) to refer to any type of reasoning about time-varying behavior based on QP. To better understand the capabilities and requirements of QR, we have found it useful to distinguish six types of QR, as enumerated below.

*Precise QR* is the qualitative analog of quantitative simulation. It partitions phase space as precisely as possible using the qualitative language of the model. It is the only type of QR for which the desired output is indeed the envisionment produced by QS using static partitioning. We claim that most tasks are actually better served by one of the other types of QR. Indeed, perhaps the only task that truly requires precise QR is demonstrating that a qualitative model is sound and complete with respect to a quantitative model. For that task, precise QR seems necessary almost by definition; at the very least it makes the task extremely straight-forward.

*Verification QR* outputs whether or not it is inevitable (or impossible) for a goal state $\mathcal{G}$ to occur sometime after an initial state $\mathcal{I}$. Unfortunately, the weak nature of QP constraints typically makes $\mathcal{G}$ neither inevitable nor impossible from $\mathcal{I}$. Nevertheless, verification QR does have the potential to be much less computationally and representationally complex that precise QR, since only binary output is required. For example, only one path from $\mathcal{I}$ to $\mathcal{G}$ need be found to conclude that $\mathcal{G}$ is not impossible after $\mathcal{I}$.

*Predictive QR* is functionally equivalent to verification QR, with $\mathcal{G}$ ranging over all of state space. A related task is temporal reasoning (TR) [14], which tells what facts hold at given time points. One can map from QR to TR representations by assigning start and end time points to every QR state. This might suggest that a reverse mapping would allow QR problems to be reduced to TR problems. Unfortunately, existing TR work does not adequately handle disjunctive time constraints. Thus, TR does address the fundamental issue of QR: what case-

splits to pursue. This means that TR-based verification QR would be potentially unsound and incomplete.

*Explanatory QR* provides causal explanations of particular behaviors. It does not simulate per se, but rather explain how a given behavioral path could arise. SIMGEN [28] is an example: it uses explanatory QR both to explain the predictions of its quantitative simulator and to update its quantitative model as influence structure changes. When a given behavior is completely specified, as in the case of SIMGEN, explanatory QR totally avoids the complexity of case-splits, since the behavior makes a commitment for every case-split.

*Differential QR* indicates how differences in system structure or initial parameter values would result in different behavior. For example, if the mass of a spring-block oscillator is heavier, the period would be longer. Weld's Differential Qualitative Analysis [51] and deKleer's Incremental Qualitative Analysis [8] are examples of differential QR. Essentially, differential QR is a form of explanatory QR in which the differences between two behavioral trees (nominal and perturbed) are explained.

Finally, *discriminatory QR* combines the later four types. It goes beyond verification and predictive QR to indicate not only whether $\mathcal{G}$ might eventually occur from $\mathcal{I}$, but what initial and intermediate conditions would determine whether $\mathcal{G}$ becomes inevitable or impossible. Furthermore, for states in which $\mathcal{G}$ is neither inevitable or impossible, differential QR can indicate whether particular perturbations would make $\mathcal{G}$ more or less likely to eventually occur.

All six types of QR could be performed by analyzing AE($\mathcal{I}$) (and a corresponding AE($\mathcal{I}$'), in the case of differential QR). For example, the output of predictive QR for each $\mathcal{G}$ could be based on which subsets of AE state descriptions are inevitable or impossible after $\mathcal{I}$. However, no type of QR besides precise QR seems to require the full complexity of QS.

We will focus on discriminatory QR for three reasons: novelty, generality, and significance. First, it has not been previously identified in QR literature, perhaps because of the emphasis on prediction over goal-directed reasoning. Second, it involves all other types of QR, so our results have general import. Third, it appears to be the most appropriate type of QR for most tasks.

Identifying discriminatory boundary states seems to be of particular importance for tasks that reason under uncertainty. For example, articulating when negative conditions become impossible and when positive conditions become inevitable allows a controller to expend minimal

efforts if such cases arise, or even strive to achieve them. Furthermore, designers/planners would like to at least ensure that their device/plan is not destined to fail if certain conditions arise. In short, identifying boundary conditions seems to be one of the most important contributions that QR can make to the overall analysis of a system or procedure.

## 1.5 A New Perspective Is Required

As we noted earlier, static partitioning has played an important role in the development of QP, when the emphasis was on studying soundness and completeness issues. However, we argue that serious attempts to study computational complexity now require a change of perspective, for which static partitioning is no longer appropriate. We call the old perspective *precision-based* because it stresses characterizing possible behaviors over time as precisely as one can in the language of the qualitative model. In that view, the key way to reduce complexity is by formulating qualitative constraints and inference mechanisms which are strong enough to limit the complete set of predictions to only a few possible trajectories through the qualitative phase space. As we have discussed, that approach does not scale up because the relevancy of specific comparisons with limit points changes over time and across goals.

In contrast, the key way to reduce complexity in our new view is by dynamically redefining the qualitative phase space partitioning over time, such that only a few possible trajectories are necessary to distinguish the complete set of ways in which goal conditions would be reached in the future from the complete set of ways in which they would not. We call this new perspective *discrimination-based* because it stresses characterizing possible behaviors over time only in enough detail to distinguish their outcomes relative to some goal.

In essence, this perspective addresses computational complexity by redefining what is computed. We argue that this new output representation suffers less from the intractabilities mentioned earlier and provides a behavioral summary which is more appropriate for most tasks.

## 1.6 Our New Approach: Discriminatory Envisionments

This thesis addresses the above problems and concerns by exploring the use of partial states and goal states to compute a new class of representations that we call *discriminatory envisionments* (DE). For a given $\mathcal{I}$ and $\mathcal{G}$, we declare any envisionment $X$ to be a DE if it is sound and

complete with respect to reachability of $\mathcal{G}$, to the same extent that AE($\mathcal{I}$) is. That is, for any complete state $S$ in AE($\mathcal{I}$) and any partial state $S_X$ in $X$ which represents a subset of $S$'s description, $S_X$ always (never) reaches $\mathcal{G}$ if and only if $S$ always (never) reaches $\mathcal{G}$. Although any AE trivially qualifies as a DE, we are interested in minimal structures which suffice to satisfy the relaxed requirements of DE's.

### 1.6.1 AE-Based Concise DE's

Let us first consider how an AE might be collapsed into alternative DE representations, denoted $E_i$. This requires a mapping from the complete states of AE($\mathcal{I}$) to the partial states of $E_i$. Consider each state $S$ in AE($\mathcal{I}$), along with each of its nexts $S_n$. If $S$ and $S_n$ differ in their reachability relation with $\mathcal{G}$ (i.e. always, possible, or never), then let them each map to a different state in $E_i$ (with a transition between them). Otherwise, let them map to the same state in $E_i$. Since we desire minimal DE representations, our definition of $E_i$ must also include some means of compressing $E_i$'s state descriptions, to allow them to each correspond to multiple AE states.

First, consider an extreme case, denoted $E_1$. Let each state $S$ in $E_1$ represent the intersection of the phase space regions represented by the AE states that map to $S$. Essentially, $E_1$ collapses AE($\mathcal{I}$) into at most three states, which are respectively intended to correspond to all states from which $\mathcal{G}$ is either inevitable, impossible, or neither.

Although $E_1$ is extremely concise, it is unfortunately not a DE. The problem is that AE states which differ in their reachability of $\mathcal{G}$ can map to the same $E_1$ state. This occurs because compressing state descriptions via intersection is too aggressive. It can easily dilute the state description of a $E_1$ state so much that it becomes a subset of the description of an AE state which does not map into it.

One way to modify the $E_1$ approach to produce DE's is to (non-deterministically) decide to not intersect state descriptions whenever that would (eventually) result in many-to-many mappings. Denote the result of this approach as $E_2$. To avoid an intersection, the AE state would be mapped to a new state in $E_2$. However, that approach has many problems as well. First, $E_2$ would not be unique, since there will be many ways of retaining detail from one state while allowing detail in another to be abstracted away via intersections. Second, the worst-case cost of searching the space of which intersections to perform can be exponentially larger than

that of QS itself, since collapsing each AE state can require consideration of the powerset of state descriptions. Thus, it could be extremely costly just to compute any such DE, let alone any maximally concise ones. Third, it is based on performing QS first; thus, we would still suffer from the computational complexities of QS even if the resulting representational complexity is improved. Fourth, the preference criterion among the admissible DE representations is based on shallow, information-theoretic grounds, as opposed to deeper causal or explanatory grounds. Thus, $E_2$ would not even be particularly useful for QR, despite the high cost they might entail to compute.

### 1.6.2 Causal-Based Sufficient DE's

We propose a DE representation which, though typically not optimal in the information-theoretic sense, retains explanatory power. It also retains the reachability soundness and completeness of AE$\mathcal{I}$ while avoiding excessive complexities due to irrelevant case-splits. Our proposal is based on the simple fact that if $\mathcal{I}$ does not always reach $\mathcal{G}$ then it must reach some state $N$ from which $\mathcal{G}$ is unreachable. Paths from $\mathcal{I}$ to $\mathcal{G}$ show how $\mathcal{G}$ could arise, while paths from $\mathcal{I}$ to $N$ show how $\mathcal{G}$ would never arise. Call these *positive* and *negative* paths, respectively. Any physical system follows at least one positive and one negative path simultaneously until some critical state is reached, after which only either $\mathcal{G}$ or some $N$ is reachable. By using partial states to formulate these paths, we can restrict ourselves to only those case-splits necessary to represent changes over time which would be sufficient to possibly reach $\mathcal{G}$ (or N). Consequentially, we call envisionments composed of such paths *sufficient discriminatory envisionments* (SUDE's). [1]

For a given $\mathcal{G}$, assume we knew of some set of states $N_1 \ldots N_n$, each known to never reach $\mathcal{G}$. Further assume that this set of states is complete, such that any state which never reaches $\mathcal{G}$ must eventually reach one of them. Instead of generating all of AE($\mathcal{I}$) explicitly, imagine that we somehow were able to directly generate all the paths between $\mathcal{I}$ and $\mathcal{G}$, as well as those between $\mathcal{I}$ and each $N_i$. The resulting subgraph of AE($\mathcal{I}$) would suffice as a DE for $\mathcal{I}$ and $\mathcal{G}$. However, it would still be far from a minimal DE, due to the inherent excessive case-splits in AE states.

[1] We prefer to pronounce "SUDE's" as "suds".

Imagine that we could instead generate sets of paths of partial states between $\mathcal{I}$ and each of $\mathcal{G}, N_1, \ldots N_n$, such that there is necessarily at least one path through AE($\mathcal{I}$) corresponding to each and that together they correspond to every path in AE($\mathcal{I}$) from $\mathcal{I}$ to each of $\mathcal{G}, N_1, \ldots N_n$. Unfortunately, such paths would not be sufficient to form a DE, because one path to $\mathcal{G}$ and one path to some $N_i$ might each contain a partial state representing (different) subsets of the state description of some $S$ in AE($\mathcal{I}$). Thus, they could conflict in claims about the reachability of $\mathcal{G}$ from $S$. However, this problem can be rectified by *interweaving* each (positive) path from $\mathcal{I}$ to $\mathcal{G}$ with each (negative) path from $\mathcal{I}$ to each $N_i$. For a given pair of paths, this interweaving would explore all consistent temporal orderings of which path's pending transition occurs first. The result would show how behavior can proceed toward both $\mathcal{G}$ and some of the $N_i$ states for awhile, and then eventually end up at either $\mathcal{G}$ or some $N_i$.

Interweaving of all positive and negative paths will generally lead to more distinctions than are required to form SUDE's. The goal of this work is to explore and characterize how much interweaving must be done to meet the requirements of SUDE's.

## 1.7 Example SUDE's

We will now summarize some examples which highlight key aspects our SUDE representation. For this purpose, we use a simple spatial/motion domain, since trajectories through this domain's phase space are especially easy to illustrate and discuss.

#### 1.7.0.1 The Ball-World Domain

Consider the scenario illustrated in Figure 1.2. By convention, the ball's height (y) being at limit point Yi means that the ball is touching the top of ledge i and y=Yi' means that the ball is touching the bottom of ledge i. For this scenario, the constraints among landmarks are as follows:

$$Y0 < Y1' < Y1 < Y2' < Y2 < Y3' < Y3,\ Y0 < Y4' < Y4, \text{ and}$$
$$X0 < X1 < X2 < X3 < X4 < X5.$$

We intentionally do not constrain the height of ledge 4 (relative to the height of the other three ledges), to illustrate that the complexity of SUDE's decreases faster (relative to AE's) as constraints are relaxed.

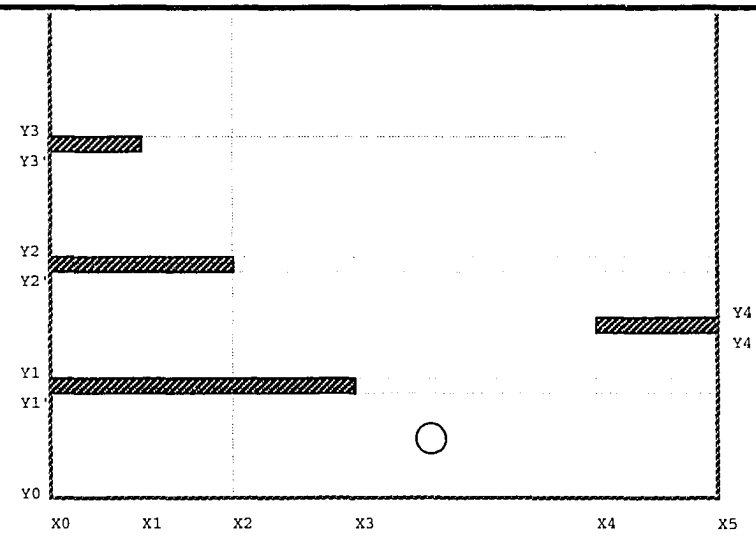


**Figure 1.2**: The ball-world scenario, with four ledges and one ball

Note: Despite graphical appearances, the height relation between ledge 4 and the other ledges is not specified.

We make two key simplifying assumptions. First, all collisions are inelastic, as if the ball is a heavy cannonball. Second, the ball never hits the side edge of any ledge, as if the ledges are infinitely thin. Our work does not alleviate the need for such modelling simplifications to manage complexity, nor do we address the issue of how to find appropriate simplifications. However, our work does help ensure that such simplifications will actually lead to tractable QR. As we shall see below, the above two simplifications lead to very concise SUDE's, whereas the corresponding AE's are still extremely complex.

### 1.7.1 Example 1: Basic Ideas

Let $\mathcal{I}$ define the ball as initially being between X3 and X4, between Y0 and Y1', and moving straight up and let $\mathcal{G}$ define the ball as being at Y0. Figure 1.3 illustrates the SUDE (SUDE1) for this example; it indicates that $\mathcal{I}$ will always reach $\mathcal{G}$ (specifically, state 1), after peaking exactly once (at state 3).

Note that SUDE1 ignores irrelevant details about which ledges the ball passes before peaking. In fact, SUDE1 does not contain even a single case-split, whereas AE($\mathcal{I}$) case-splits exten-

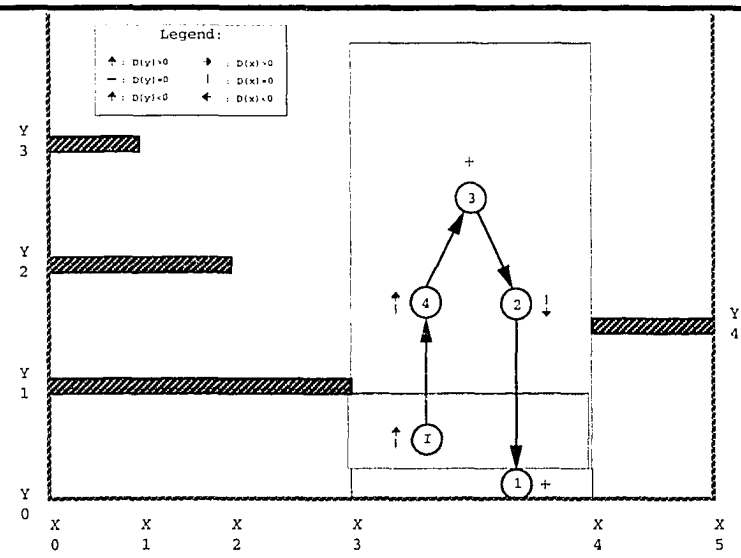


**Figure 1.3**: SUDE1

Each shaded box indicates the region in phase space specified by the corresponding state. Note that state 4 is a generalization of state I, ignoring details about whether y is below Y1' or not.

sively. For instance, AE($\mathcal{I}$) case-splits on whether the ball passes ledge 1 (i.e. y=Y1) before peaking (i.e. d(y)=0) or not. Furthermore, since Y4 is unconstrained with respect to Y1, Y2, and Y3, AE($\mathcal{I}$) almost fully case-splits on the set $y < Y4 \vee y = Y4 \vee y > Y4$. For example, for every AE state specifying $y > Y4$, there are corresponding AE states which only differ by specifying $y < Y4$ or $y = Y4$ instead. The only exceptions are states specifying $y = Y0$, which must necessarily specify $y < Y4$ (due to transitivity).

SUDE1 essentially tracks over time only the relations between y and Y0 and between d(y) and 0. All other relations are specified in states only when they are inferable by those two relations and other relations that necessarily persist from $\mathcal{I}$. This illustrates how SUDE's avoid the two causes of excessive complexity that we mentioned earlier (case-splitting on all nexts and on all completions). First, SUDE1 does not track relations between y and landmarks Y1', Y1, Y2', Y2, Y3', and Y3', even though y is known to be less than each of them in $\mathcal{I}$. This avoids case-splits on nexts of $\mathcal{I}$ that are irrelevant. Second, the relations between y and Y4' and between y and Y4, which are unspecified in $\mathcal{I}$, are never introduced in SUDE1. This avoids case-splits on completions of $\mathcal{I}$ that are irrelevant.

#### 1.7.1.1 Regressing With Qualitative Dynamic Operators

We generate paths between $\mathcal{I}$ and $\mathcal{G}$ by regressing with operators, much in the style of state-based planners such as STRIPS. However, unlike standard STRIPS-like operators, the effects of our operators do not necessarily hold once all the conditions are satisfied. Each *dynamic* operator specifies conditions under which the derivative of a particular quantity Q could have a particular sign and cause Q to change its relation with some particular landmark. Figure 1.4 illustrates one dynamic operator used in this example.

A dynamic operator's change might not actually occur, for one of two reasons. First, some of the conditions might stop holding before the derivative causes the change to occur. Second, competing operators corresponding to the same derivative but with opposite sign might also be active; this could prevent the change from occurring even if the conditions hold indefinitely. These two sources of ambiguity are unavoidable; they result from inherent limitations of QP constraints — the same limitations which make multiple nexts in QS so common. This precludes us from being able to simplify via the STRIPS assumption (i.e. that the explicit set of operator effects are complete).

Only in special circumstances can one formulate a dynamic operator using enough qualitative conditions to ensure that its change *must* occur if its conditions all hold. A theory for formulating dynamic operators should, at the very least, identify those circumstances. However, more generally, the theory should also identify sets of necessary and sufficient conditions under which it is consistent that a particular change *might* occur.

For example, $x \geq X3$ is a necessary condition of the operator OP1 of Figure 1.4; omitting it would suggest that the ball could reach Y0 by passing through ledge 1. Imagine that the model is enhanced to include the possibility of a upward wind from Y0 and downward wind from Y3. OP1 would still be correct unless the following constraints hold: the forces of the downward wind and gravity are individually less that the force of the upward wind but together are greater. In that case, OP1 must further be conditioned on the upward wind being inactive or both winds being active. That would result in multiple new operators to replace OP1.

We leave the details of our theory for automatically formulating dynamics operators from a qualitative model for later. For now, the important thing to note is that by regressing with our

conditions:
$y > Y0$
$vy < 0$
$x \geq X3$
$x \leq X4$
$(ay < 0)$
change:
$y = Y0$

**Figure 1.4**: Example qualitative dynamic operator (OP1)

This operator represents one instantiation of a negative influence on quantity y, due to gravity (ay).

dynamic operators, we are able to provide sufficient causal explanations of how any $\mathcal{G}$ can arise from any $\mathcal{I}$, without introducing the sort of irrelevant distinctions that are so prevalent in QS.

#### 1.7.1.2 Qualifying Regressed Paths to Form SUDE's

Regression paths only show the simplest ways in which $\mathcal{G}$ can be achievable from $\mathcal{I}$. They do not show how $\mathcal{G}$ becomes impossible nor how facts in $\mathcal{I}$ can change even when such changes are not necessary for $\mathcal{G}$ to occur.

In order to formulate a complete SUDE, we must consider all of the nexts of each regression state. This can invoke additional regression, to show how $\mathcal{G}$ might be reached (or is impossible) from each of these nexts. We call this *qualification* of the regression paths. Qualification explores alternative behaviors that could occur before each regression operator has a chance to cause the desired change to occur.

SUDE1 itself involves no qualifications, because no case-splits are required to characterize the behavior of this simple example.

We refer to the propositions in the state description of a state S as the propositions to which S is *committed.* Thus, committing a state to a particular proposition means adding it to that state's working description. A key role of qualification is deciding what commitments to add to the partial states articulated during regression.

#### 1.7.1.3 Key Issue 1: What Facts to Regress

In addition to having a different sort of operator, our approach differs from STRIPS-like planning in that we do not necessarily regress all facts true in each intermediate goal state. The need for selective regression arises from our desire to avoid committing states to irrelevant details that can lead to excessive branching at qualification time. This issue does not arise in classical planning, which has no analog to our qualification stage.

For example, we do not regress the relation between y and Y1 when regressing from state 1 back to state 2 during the generation of SUDE1. Regressing that relation would yield a specialization of state 2 which also specified $y < Y1$, since $y < Y1$ is derivable in state 1 (from transitivity on $y = Y0$ and $Y0 < Y1$) and is not affected by the regression operator (OP1). In fact, regressing it would make all regressed states (on the path from $\mathcal{G}$ back to $\mathcal{I}$) specify $y < Y1$, since $y < Y1$ is already established in $\mathcal{I}$. So, qualifying that regression path would introduce branching on whether y passes Y1 before the ball peaks, an irrelevant case-split that SUDE1 avoids.

Unfortunately, avoiding regression of derivable facts is not sufficient to avoid introducing all irrelevant case-splits that arise in QR. In particular, a major claim of this thesis is that derivative relations should generally not be regressed, even when they are not derivable. The intuition behind this claim is that derivatives are fundamentally different from non-derivatives in ways that warrant such special treatment. Generally, committing a particular state to a particular derivative relation does not constrain the SUDE in a useful way. For example, specifying that $d(y < 0)$ holds in state 2 does not change the fact that $y > Y0$ in state 2 could either change to $y = Y0$ (transitioning to state 1) or persist (transitioning to some other state because some other change occurs first).

SUDE1 might appear to contradict the above claims about non-regression of derivable and derivative relations. For example, SUDE1 appears to regress $d(y) = 0$ in state 1 to $d(y) < 0$ in state 2; but, in fact, that is not true. Instead, state 2 specifies $d(y) < 0$ because it is inferred from $vy < 0$, a condition of OP1 representing a downward velocity of the ball along the y-axis. Thus, it turns out that the derivative relation between d(y) and 0 is effectively regressed, but only because it is inferable from the regressed relation between vy and 0. This special

circumstance fundamentally exists because there is only one influence on y, due to movement. Thus, when the conditions of that influence hold, the derivative relation necessarily holds.

This approach effectively tracks derivative changes only when they do not introduce branching. We will argue that such cases are the only ones for which tracking derivatives over time actually leads to interesting conclusions. For SUDE1, it results in identifying that the ball necessarily rises, peaks, and then falls during its trajectory from $\mathcal{I}$ to $\mathcal{G}$.

#### 1.7.1.4 Key Issue 2: What Facts to Project

Regression tends to make early states more detailed (i.e. specifying new facts) than later states, due to the backward accumulation of facts as operator conditions are added to the working goal state. In STRIPS-like planning, it is clear how to project such facts down the regression path, to make all states at the same detail (i.e. specifying the same propositions, or their negations). Specifically, the STRIPS assumption implies that each operator's conditions persist through $\mathcal{G}$, unless deleted by that operator or by later operators in the plan.

Without the STRIPS assumption, similar projection would transform a regression path of successively less detailed states into a tree of same-detail states. This tree would represent every consistent way that facts in early states could change over time, eventually passing through each regression state. Such a tree would approach the complexity of AE($\mathcal{I}$), except that the its states would not necessarily be complete and its paths necessarily stop at states consistent with $\mathcal{G}$.

Having all states explicitly specified at the same detail might seem to be of particular importance when the STRIPS assumption does not hold. The underlying intuition is the same one behind the motivation for case-splitting on all completions of $\mathcal{I}$ to produce AE's. Namely, that a path of partial states might indicate a transition between two states (i.e. S1 and S2) for which there are no completions of S1 and S2 (ie. S1' and S2') such that S1' transitions to S2' in AE($\mathcal{I}$). Of course, this issue does not arise under the STRIPS assumption, where operator conditions are sufficient to guarantee transition to their effects, by definition.

However, we claim that our formulation of dynamic operators precludes the need for such complete projection, even though these operators do not guarantee their changes will occur.

Nevertheless, projecting certain types of facts from $\mathcal{I}$ can significantly reduce the complexity of regression. In particular, it is useful to project facts from $\mathcal{I}$ as long as branching is not required. It turns out that there are two such cases: when $\mathcal{I}$ has only one next state at the

same detail as itself and when facts persist forever from $\mathcal{I}$. For the first case, $\mathcal{I}$ is updated to be its necessary next state, before regressing. For the second case, forever-persisting facts are added to each state during regression.

For example, $d(x) = 0$ necessarily persists forever from $\mathcal{I}$ in SUDE1 because the model mentions no possible influences on x. Since $X3 < x < X4$ holds in $\mathcal{I}$, that means $d(x) = 0$, $x > X3$, and $x < X4$ hold for every state after $\mathcal{I}$. These constraints ensure that regression does not explore possibilities such as the ball falling to $\mathcal{G}$ from being under ledge 4 or ledge 1. We are not claiming that such application of projection for more efficient regression is especially novel. Instead, we are noting that it appears to be the only worthwhile use of projection, since other types of projection appear to introduce branching without particular benefits.

#### 1.7.1.5 SUDE's Are Non-Optimal

Note that SUDE1 does not avoid all details which turn out to be irrelevant to distinguishing behaviors in which $\mathcal{G}$ occurs from those in which it does not. In particular, SUDE1 does not collapse states S2, S3, and S4 into one, even though the result would still be a SUDE and would still show that the ball will eventually fall to Y0 due to gravity. In fact, the minimal DE for example 1 requires just two states:

$$(y > Y0, X3 < x < X4, d(x) = 0) \rightarrow (y = Y0, X3 < x < X4, d(x) = 0).$$

However, the relation between d(y) and 0 has causal relevance, as reflected by the conditions of operators such as OP1. There appears to be no general way to correctly anticipate whether such distinctions will actually turn out to be necessary to partition phase space into regions which are on the trajectory from $\mathcal{I}$ to $\mathcal{G}$ and those which are not. So, we accept such relatively rare cases of unnecessary case-splitting. The most important thing to note is that SUDE's tend to avoid the case-splits which are the biggest source of excessive complexity in AE's, such as relations between y and the ledge heights in example 1.

### 1.7.2 Example 2: No Explicit Goal

The most extreme type of SUDE is one based on an empty $\mathcal{G}$. An empty $\mathcal{G}$ suggests that all behaviors from $\mathcal{I}$ are potentially interesting. However, even in this case there are better SUDE's than AE($\mathcal{I}$) itself. From our discrimination-based perspective, an empty $\mathcal{G}$ implies that what is

most interesting is which states are inevitable or impossible from $\mathcal{I}$, as well as what behaviors might occur to make some merely possible states become inevitable or impossible. Thus, a SUDE for an empty $\mathcal{G}$ need only describe behaviors in terms of those states, not all states in AE($\mathcal{I}$).

The key insight for formulating such SUDE's is that every system behavior eventually enters some minimal subgraph of AE($\mathcal{I}$), within which it remains forever. The two most common examples of such subgraphs are equilibrium states and oscillation cycles. By identifying the states of such subgraphs and treating each as a $\mathcal{G}$ for goal-directed SUDE generation, we can develop a composite SUDE which indicates all causally minimal ways in which the system can reach one of these subgraphs. States which are inconsistent with every state in the resulting SUDE are impossible and states which are consistent with some state in every path through that SUDE are inevitable. The main point is that we let the specific dynamical interactions of the system (which define the minimal terminal subgraphs) guide the SUDE-generation process, unlike QS (which is not goal-directed).

There are five states in which quantity $y$ is in equilibrium, each indicating that the ball is at one of the five landmarks $Y0 \ldots Y4$. Let Ei denote the equilibrium state where $y = Yi$. For the same $\mathcal{I}$ as example 1, only E0 is consistent with the facts that always persist from $\mathcal{I}$ (i.e. $x > X3$ and $x < X4$). Thus, using E0 as $\mathcal{G}$, the resulting SUDE is identical to SUDE1.

### 1.7.3 Example 3: Other Issues

Now, let $\mathcal{I}$ specify only that y is initially below Y1' and let $\mathcal{G}$ specify that the ball eventually rises above Y2. Figure 1.5 illustrates the SUDE (SUDE2) for this example. SUDE2 makes several case-splits: whether d(x) is initially negative, zero, or positive, whether d(y) becomes 0 before the ball reaches Y2, and whether the ball falls under ledges 1 or 4 or between ledges 2 and 1. Despite involving many more case-splits than SUDE1, SUDE2 still case-splits much less than AE($\mathcal{I}$), and in fact each case-split is relevant to whether $\mathcal{G}$ occurs.

We briefly note below some additional issues that this example illustrates.

#### 1.7.3.1 Overlapping Regression Paths

During regression, $x < X3$ in state 4 is achieved by committing state 5 to $d(x) < 0$ and $x = X3$. Eventually, $d(x) < 0$ regresses all the way back from state 5 to $\mathcal{I}$, creating a refinement of $\mathcal{I}$

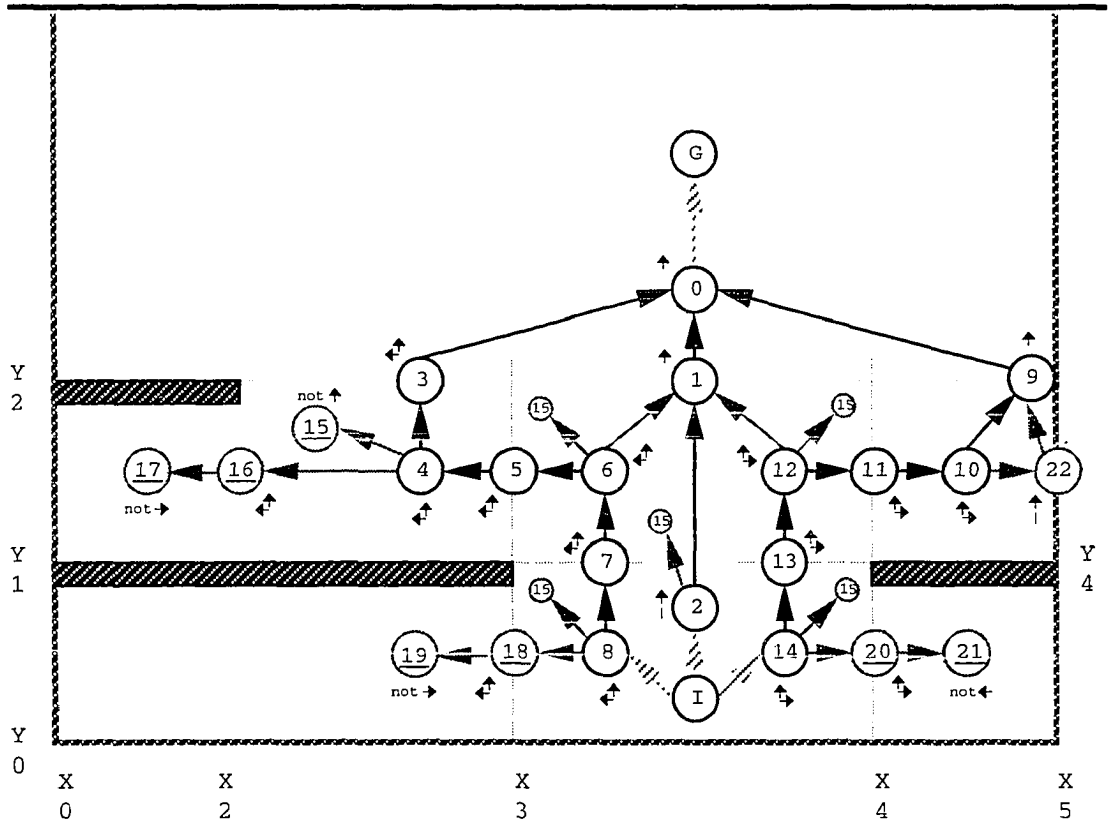


**Figure 1.5**: SUDE2

States, transitions, and state refinements due to qualification are shown faded, relative to the portion of the SUDE due to regression.

(state 8). Symmetrically, $d(x) > 0$ is regressed from state 11 to create other refinements of $\mathcal{I}$ (states 12, 13, and 14). However, regression does not lead to a refinement of $\mathcal{I}$ for the remaining case (i.e. $d(x) = 0$); regressing from state 1 to state 2 does not require a commitment to $d(x) = 0$.

It turns out that committing state 2 to $d(x) = 0$ is appropriate. However, we are only able to make this realization at qualification time. At that time, we note that states 6 and 12 may transition to state 1, which means that the only case in which state 2 is unique is when it commits to $d(x) = 0$; other cases are already represented by states 6,7,8 and 12,13,14.

This illustrates a general issue for the SUDE representation: some regression paths can overlap others. Essentially, these overlaps occurs when different partial versions of the same AE state are represented. In general, this could result is a SUDE having more states than a corresponding AE. However we have yet to notice such a situation and we suspect that such a situation would be extremely exceptional.

#### 1.7.3.2 Falling Off Regression Paths

Unlike SUDE1, SUDE2 is indeed affected by qualification. For example, state 6 has three nexts at its specified level of detail, each representing one of the following changes: $x = X3$, $y = Y2$, or $d(y) = 0$. Only the first change is represented in the regression paths (as a transition between states 6 and 5). The second change can be handled by recording a new transition between states 6 and 1. However, the third change must transition to a new state. The corresponding next state (call it N3) is inconsistent with all existing states (1-14).

We refer to this third case as "falling off" the regressed paths. Qualification attempts to generate regression paths between N3 and $\mathcal{G}$. Since there are none, $\mathcal{G}$ is impossible from N3.

One could record a transition from state 6 to N3 and label N3 as a negative state. However, that would lead to excessive case-splitting because, in fact, any change to $d(y) \leq 0$ in SUDE2 makes $\mathcal{G}$ impossible. As part of qualification, we isolate a portion of N3's specification which makes $\mathcal{G}$ impossible. In this case, that portion is $d(y) \leq 0\, y < Y2$, represented as state 15. This allows states 4, 6, 12, 10, 8, 2, and 14 to all transition to state 15, instead of their own specific next representing a change to $d(y) = 0$.

#### 1.7.3.3 Tradeoff: Precision Versus Simplicity

Our ball-world model indicates that $d(x) < 0$ persists through any next of state 6. Similarly, $d(x) = 0$ persists through any next of state 2 and $d(x) > 0$ persists through any next of state 12. Nevertheless, SUDE2 does not case-split state 1 by the relation between d(x) and 0; this allows state 1 to simultaneously represent a next of each of states 6, 2, and 12. In contrast, state 4 does commit to the $d(x) < 0$ that persists from state 6. Without that commitment, state 4's nexts would (unsoundly) include one in which $x = X3$ occurs.

The key intuition is that we are willing to tradeoff precision for reduced complexity as long as path soundness is maintained. For example, despite the imprecision of state 1, every next of state 1 is reachable from every previous state of state 1. We will argue that this tradeoff can be performed locally, without the need to case-split first and then collapse unnecessary case-splits.

## 1.8 Road Map

This thesis re-examines the very nature of envisionments from our new discrimination-based perspective. Thus, it seems appropriate to organize this thesis into chapters which successively focus on each of the four fundamental representations employed: states, transitions, paths, and graphs. Highlights of each of these four chapters are as follows.

Chapter 2 presents a detailed re-examination of the types of qualitative state distinctions that one can make, with emphasis on when their inclusion in state descriptions is worth the cost of introducing higher-order ambiguities. As a consequence of systematically re-exploring these issues, oblivious to the complexities that static partitioning might bring, we have discovered some useful new classes of distinctions as well. As a result, we have found that envisionments can be globally unsound in more ways than previous QP work has suggested. In any case, this chapter lays the theoretical groundwork for one of the key underlying contributions of this thesis: exploring the utility and validity of reasoning with partial states, as opposed to complete states.

Chapter 3 re-explores the issue of how to define the set of nexts for a given state. We argue that previous approaches are both unsound and incomplete. For example, most work in QP only considers nexts that correspond to continuous changes, which makes them incomplete unless tedious details are modelled and simulated. We show that when generalizing to account for

discontinuous changes, such as those due to sudden actions, previous approaches are unsound as well. Furthermore, we show that such unsoundness extends to the related areas of model update and formal theories of change. We present a new approach to minimality-based change which avoids these problems. We argue that, using this new approach, QP provides appropriate representations for basing formal theories of change.

Chapter 4 explores the issue of generating paths between two states S1 and S2. Naturally, this issue is related to planning. However, whereas classical plans *prove* the achievability of goals via *discrete* actions, these paths indicate sufficient conditions under which S2 *could* arise from S1 due to *continuous* and *ambiguous* dynamics. Thus, the emphasis in this chapter is not on goal-directed search per se but on the special issues that arise due to reasoning about derivatives and under ambiguity. A key contribution of this chapter is the automatic formulation of our dynamic operators from a qualitative model.

Chapter 5 builds on the previous three chapters to show how paths between $\mathcal{I}$ and $\mathcal{G}$ can be qualified to form SUDE's. The major concern in this chapter is how to ensure soundness and completeness without excessive case-splits.

Finally, Chapter 6 concludes with a summary and a discussion of future and related work.

# Chapter 2

# Representing Qualitative States

The issue of how to qualitatively summarize the possible states of the world is central to all work in qualitative physics. However, this issue has traditionally been explored from the perspective of qualitative simulation of states at a relatively uniform level of detail. Since the complexity of such simulation critically depends on the level of detail, such work has necessarily emphasized *what* distinctions are typically worth making across *all* states.

In this chapter, we re-explore this fundamental issue from a relaxed perspective in which states are allowed to vary in detail over time. From this new perspective, the key issues become:

- *when* particular classes of distinctions are useful and
- *how* to maintain an appropriate level of detail over time.

In order to understand these issues, the first half of this chapter explores the nature of qualitative constraints. The main result from this analysis is the realization that different classes of state distinctions can and should be treated differently. For example, it turns out that derivative conditions need not be explicitly tracked over time. The same degree of soundness can be achieved by tracking the more fundamental conditions that constrain them. Thus, the *chattering* derivatives problem [41, 29] disappears because we do not branch on ambiguous changes in derivative conditions.

Such insights suggest local *sufficiency* conditions for state descriptions. These conditions are discussed in the second half of this chapter. Later chapters will build upon these concepts, showing how further refinements of these states should be made to reflect global constraints due to state transitions and paths of transitions.

## 2.1 Qualitative Process Theory

We assume that the physical system is modelled using Forbus' Qualitative Process Theory (QPT) [24]. We will argue throughout this thesis that QPT offers particularly useful leverage for goal-directed reasoning with partial states. This section highlights features of QPT which are fundamental to this thesis. We presume that this thesis might also have general utility to other ontologies which share these key features.

### 2.1.1 Quantity and Derivative Inequalities

Inequalities are the main primitives with which QPT qualitatively partitions phase space. Each inequality refers to an assertion of a specific ordinal relation between two numbers. We refer to inequality between the values of two quantities as a *quantity inequality* and the inequality between the values of the derivatives of two quantities as a *derivative inequality.*

We will generally refer to quantity values as $Q_i$, derivative values as $D(Q_i)$, and numbers (which are either) as $N_i$. [1] For uniformity, we assume that the form of all inequalities is $N_1$ *vs* $N_2$, with $D(Q_1)$ *vs* $D(Q_2)$ referring to the derivative inequality of the quantity inequality $Q_1$ *vs* $Q_2$, and vice versa. We reformulate exceptions (e.g. $Q_1 > Q_2 + Q_3$) by introducing equivalent internal numbers (e.g. $Q_1 > Q_{i1}$ and $Q_{i1} := Q_2 + Q_3$). Furthermore, we assume that both sides of a derivative inequality are derivatives; again, reformulating as needed to enforce this.

To facilitate partial state representations, we allow some disjunction in inequality relations. Since it rare to know only "$Q_1 > Q_2$ or $Q_1 < Q_2$", without $Q_1 = Q_2$ also being possible, it turns out to be most useful to represent all inequalities in terms of the relation $\geq$, as shown in Table 2.1. Despite their generality, soft inequalities even lead to more concise encodings, using only two propositions per relation instead of three.

[1] We deviate slightly from standard QPT notion, where the value (amount) of a quantity $Q$ is referred to as $A[Q]$. The rationale behind the $A[]$ qualifier is to distinguish equality of values from equality of quantities (which implies equality of all their higher-order derivatives). To avoid that problem, we use $Q_1 := Q_2$ to refer to definitional equality and $Q_1 = Q_2$ to refer to value comparisons.

$N_i > N_j \Leftrightarrow N_i \geq N_j \wedge N_j \not\geq N_i$
$N_i = N_j \Leftrightarrow N_i \geq N_j \wedge N_j \geq N_i$
$N_i < N_j \Leftrightarrow N_i \not\geq N_j \wedge N_j \geq N_i$

$N_i \geq N_j \vee N_j \geq N_i$

$N_i \not\geq N_j \Leftrightarrow N_i < N_j$
$N_j \not\geq N_i \Leftrightarrow N_i > N_j$

**Table 2.1**: "Hard" vs "soft" inequality representations

$N_k$ signifies a number (either a quantity or its derivative).

### 2.1.2 Functional Relations

QPT employs *qualitative proportionalities* (qprops) and *correspondences* to represent partial information about functional relations between quantities.

We denote each qprop with a Q+ or a Q-, depending on the direction of proportionality. For example, the two qprop relations $x1 \xrightarrow{Q+} y$ and $x2 \xrightarrow{Q-} y$ qualitatively represent functional relations of the form $y = f(x1, x2, \ldots)$, such as $y := x1 - c \cdot x2 + d$. Note that qprops commit to a particular causal ordering, in this case from x1 and x2 to y.

A correspondence relation for a given set of qprops indicates a set of quantity/value pairings representing one point in a function underlying those qprops. For instance, the correspondence $\texttt{corr}((y,0),(x1,0),(x2,0))$ implies $f(0,0) = 0$. Furthermore, if equality relations hold for all but two of the pairs, the ordinal relation of each other pair is inferable by the other. For instance, if the above correspondence holds then $y = 0 \Rightarrow ((x1 < 0 \Leftrightarrow x2 < 0) \wedge (x1 = 0 \Leftrightarrow x2 = 0) \wedge (x1 > 0 \Leftrightarrow x2 > 0))$ holds as well.

We give special attention in this thesis to correspondences of exactly two pairs, which we call *same-relations*. Same-relations are defined as follows:

$$\texttt{Same-Rel}((Q1, L1), (Q2, L2)) \equiv$$
$$(Q1 < L1 \wedge Q2 < L2) \vee (Q1 = L1 \wedge Q2 = L2) \vee (Q1 > L1 \wedge Q2 > L2).$$

There are three good reasons to highlight same-relations. First, our QPT modelling experience suggests that correspondences of more than two pairs are much rarer. Second, all correspon-

dences can be reformulated more concisely in terms of same-relations; our previous example becomes:

$$y = 0 \Rightarrow \texttt{Same-Rel}((x1, 0), (x2, 0)),$$

$$x1 = 0 \Rightarrow \texttt{Same-Rel}((x1, 0), (y, 0)),$$

$$x2 = 0 \Rightarrow \texttt{Same-Rel}((x1, 0), (y, 0)),$$

$$x1 > 0 \wedge x2 > 0 \Rightarrow y > 0,$$

$$x1 < 0 \wedge x2 < 0 \Rightarrow y < 0.$$

Third, explicitly using same-relations in state definitions allows enforcing the (three-way) case-split without necessarily committing to a specific case.

### 2.1.3 Influences

Saying that quantity Q1 *influences* quantity Q2 is equivalent to saying that D(Q2) is functionally related to Q1. So, one could qualitatively represent an influence from Q1 to Q2 as a chain of qprops connecting Q1 with D(Q2), such as:

$$Q1 \overset{Q+}{\rightarrow} Q3 \overset{Q-}{\rightarrow} D(Q4) \overset{Q+}{\rightarrow} D(Q2).$$

However, QPT's distinction between *direct* and *indirect* influences avoids the need to represent qprops both in terms of quantities and their derivatives. For example, QPT represents the above influence chain as follows:

$$Q1 \overset{Q+}{\rightarrow} Q3 \overset{I-}{\rightarrow} Q4 \overset{Q+}{\rightarrow} Q2,$$

where I+ and I- denote direct influence relations. Whereas the influence from Q3 to Q4 is direct, the influence from Q3 to Q2 is indirect.

Direct influences essentially represent integration, giving an explicit modelling commitment of where causal feedback cycles should be broken.

### 2.1.4 Processes and Views

QPT organizes the conditions under which functional relations hold into *views* and *processes*. Figures 2.1 and 2.2 illustrate some of the processes and views used in our examples. A process or view instantiation is called *active* if all of its conditions hold; otherwise it is called *inactive*.

Processes differ from views in that only processes specify direct influences; thus, processes represent the primary agents of dynamic change.

By allowing functional relations for a given quantity to be partially specified across multiple processes and views, QPT supports (and, in fact, encourages) composible models. Each qprop or direct influence relation from quantity $Q_i$ to quantity $Q_j$ contributes to the composed functional definition of $Q_j$. More precisely,

$$Q_i \xrightarrow{Q+} Q_j \equiv Q_j = f(\ldots, Q_i, \ldots) \wedge \frac{\partial f}{\partial Q_i} > 0,$$

$$Q_i \xrightarrow{Q-} Q_j \equiv Q_j = f(\ldots, Q_i, \ldots) \wedge \frac{\partial f}{\partial Q_i} < 0,$$

$$Q_i \xrightarrow{I+} Q_j \equiv D(Q_j) = f(\ldots, Q_i, \ldots) \wedge \frac{\partial f}{\partial Q_i} > 0,$$

$$Q_i \xrightarrow{I-} Q_j \equiv D(Q_j) = f(\ldots, Q_i, \ldots) \wedge \frac{\partial f}{\partial Q_i} < 0.$$

Whereas previous work has stressed how QPT's organizational and composibility properties support the task of modelling, our work demonstrates that these same properties can be exploited during QR itself. First, the process/view (PV) conditions indicate the primary conditions to regress during goal-directed search. Second, QPT's commitment to causal ordering provides an explicit direction for regression. Third, the decompositional nature of QPT models allows us to focus on sufficient subsets of the model that make goal behaviors qualitatively possible. Whereas QS hides many of these issues, we bring them out by reasoning with partial states and goal states.

### 2.1.5 Domain Theories Versus Scenario Models

Throughout this thesis, we will assume that our model of the physical system is propositional; this is called the *scenario model.* It is obtained by instantiating a first-order *domain theory* of general physical constraints (such as process definitions) by a propositional *scenario* specific to the physical system and overall task.

The scenario provides three major types of constraints:

1. statics (e.g. configuration of containers and pipes),
2. limits on dynamics (e.g. pressure $P$ will always be below limit point $L$),
3. modelling assumptions (e.g. ignore friction, consider gases).

```
(defProcess (Flow ?qty-type ?ob1 ?ob2 ?rate)
  Instance-of ((Flow-Rel ?qty-type ?ob1 ?ob2 ?rate))
  Influences  ((I+ (?qty-type ?ob2) ?rate)
               (I- (?qty-type ?ob1) ?rate)))

(defView (Fluid-Flow ?path ?rate ?src-port ?dst-port ?src-cs)
  Instance-of ((Primed-Fluid-Flow ?path ?rate ?src-port ?dst-port ?src-cs ?dst-cs))
  Conditions  ((Aligned ?path))
  Relations   ((= ?rate (virtual ?rate))
               (Flow-rel MASS ?src-cs ?dst-cs ?rate)))

(defView (Primed-Fluid-Flow ?path ?rate ?src-port ?dst-port ?src-cs ?dst-cs)
  Instance-of ((Supports-Fluid-Flow ?path ?rate ?src-port ?dst-port ?src-cs ?dst-cs))
  Individuals ((?path :type Pipe))
  Conditions  ((> (pressure ?src-port :ABSOLUTE) (pressure ?dst-port :ABSOLUTE)))
  Relations   ((Qprop0+ (virtual ?rate) (pressure ?src-port ?dst-port))))

(defView (Supports-Fluid-Flow ?path ?rate ?src-port ?dst-port ?src-cs ?dst-cs)
  Instance-Of ((Exposed-to ?src-port ?src-cs))
  Individuals ((?path :conditions (Fluid-Connect ?path ?src-port ?dst-port))
               (?rate :bind (flow-rate ?path ?src-cs))
               (?src-cs :form (C-S ?sub ?st ?src-can))
               (?dst-port :form (?dst-pn ?dst-can))
               (?dst-cs :bind (C-S ?sub ?st ?dst-can))))

(defView (Exposed-to ?pt ?cl)
  Individuals ((?cl :type Contained-Liquid :form (C-S ?sub LIQUID ?can))
               (?pt :type Portal :form (?pn ?can)))
  Conditions  ((> (height (liquid-in ?can)) (height ?pt))
               (Positive (mass ?cl))))
```

**Figure 2.1**: Key definitions for fluid free-flow processes

```
(defView (Pumped-Flow ?path ?rate ?src-port ?dst-port)
  Instance-of ((Primed-Pumped-Flow ?path ?rate ?src-port ?dst-port ?src-cs ?dst-cs))
  Conditions  ((On ?path))
  Relations   ((= ?rate (virtual ?rate))
               (Flow-rel MASS ?src-cs ?dst-cs ?rate)))

(defView (Primed-Pumped-Flow ?path ?rate ?src-port ?dst-port ?src-cs ?dst-cs)
  Instance-of ((Supports-Fluid-Flow ?path ?rate ?src-port ?dst-port ?src-cs ?dst-cs))
  Individuals ((?path :conditions (Pump-Connection ?path ?src-port ?dst-port)))
  Relations   ((:= (virtual ?rate) (spec-flow-rate ?path))))

(defEntity (Fluid-Pump ?pump)
  (Positive (spec-flow-rate ?pump)))
```

**Figure 2.2**: Key definitions for fluid pumped-flow processes

```
(defScenario PUMP-CYCLE
  (State LIQUID)
  (Substance WATER)
  (Fluid-Path-Connection (bottom CAN1) (bottom CAN2))
  (Fluid-Pump-Connection (bottom CAN2) (bottom CAN1))
  (= (height (bottom CAN1)) (height (bottom CAN2))))
```

**Figure 2.3**: Simple pump-cycle scenario

Figure 2.3 gives a scenario description for a simple pump-cycle system that we will often use to illustrate key points; Figure 2.4 illustrates this system for a particular state.

The scenario model includes closed-world assumptions that the possible existence of individuals is limited to those explicitly mentioned in the scenario model. The main consequence of these assumptions is that the sets of possible influences on quantities are closed.

We denote the constraints of the scenario model as $\mathcal{M}$. Let $\mathcal{P}$ denote the set of all propositions mentioned in $\mathcal{M}$, plus their negations. $\mathcal{P}$ includes the special proposition $\perp$, indicating logical inconsistencies (e.g. $A \wedge \overline{A} \Rightarrow \perp$).

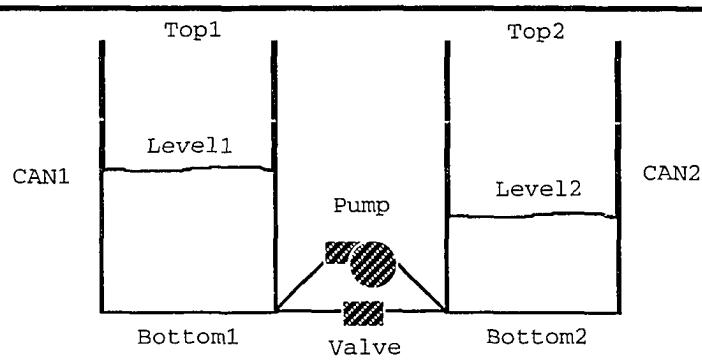


**Figure 2.4**: The pump-cycle system in equilibrium

## 2.2 The Qualitative Law of Diminishing Returns

Since the complexity of any search problem is proportional to the level of state detail, the choice of the state description language can be critical. Indeed, the worst-case cost of a search problem can typically be transformed from exponential to polynomial by using an appropriate abstract language [38]. In most AI search problems this issue is not always so critical, since the worst-case complexity can often be avoided using best-first search. For example, planning in a static world typically will not require exploring the entire search space unless there is no valid plan.

Unfortunately, qualitative simulation must always explore its entire search space, since the world is in control of the dynamics. To completely cover the range of possible qualitative behavior, one must branch on whether each ambiguous change might occur. [2] For this reason, qualitative simulation is generally restricted to a finite set of states — otherwise it would not even be decidable.

What makes this problem especially difficult is that it is generally a losing proposition to try to reduce ambiguity by adding more qualitative detail. This *law of diminishing returns* arises for two reasons. First, there is a *locality of utility* for most distinctions. For example, in the pump-cycle scenario shown in Figure 2.4, case-splitting on whether the pump rate is

[2]Chapter 4 presents a restrictive form of qualitative simulation analogous to planning, which assumes that dynamics act cooperatively with respective to some specific goal state. However, such reasoning does not obliterate the need for exhaustive search. Indeed, as explained in Chapter 5, a qualification stage is still required, to consider ways in which dynamics may defeat such fortuitous behavior.

equal, stronger, or weaker than the flow rate tells whether the liquid level in CAN1 is steady, rising or dropping. However, if the valve of some other pipe into CAN1 is later opened, then the distinction between those two rates is no longer sufficient. In fact, without distinctions comparing those rates with the rate of the new flow, that distinction simply results in the number of future states being excessive by a factor of three.

Whereas the problem of locality could be addressed by changing the level of detail appropriately over time, the second problem is more fundamental to the nature of qualitative constraints. Consider partially resolving an ambiguity in whether a pressure might rise to a critical point by case-splitting on whether that pressure is currently rising or dropping. The problem is that one has reduced that ambiguity but introduced new ambiguity — in this case, ambiguity in how the derivative of the pressure is changing. We call this the problem of *ambiguity regress.*

We suspect that the old precision-based perspective prevented earlier work in QR from identifying this problem of diminishing returns. In that perspective, every distinction always has some value, since it helps bring qualitative state descriptions closer to the precision of quantitative states. However, in our discrimination-based perspective, a state distinction's value diminishes whenever it becomes irrelevant to whether goal state $\mathcal{G}$ can arise. Thus, any distinction which causes the number of envisionment states to increase, without improving envisionment soundness with respect to goal reachability, should be avoided. In the following section we begin to address this question of when particular types of distinctions are relevant.

## 2.3 Types of State Distinctions

In general, any proposition in $\mathcal{P}$ could serve as a state distinction. In fact, in many approaches, including QSIM, all propositions are treated equally as state distinctions. However, this section shows that QPT actually imposes some strong constraints which make it useful to treat classes of propositions differently. The following subsections discuss the classes which define a subset $\mathcal{P}'$ of $\mathcal{P}$ that is sufficient for defining QPT states. Since this formulation facilitates efficient reasoning using assumption-based truth maintenance systems [9] [5], we shall also refer to propositions in $\mathcal{P}'$ as state *assumptions.*

### 2.3.1 Process/View Conditions

We use the term *influence structure* to refer to a directed graph of quantity nodes connected by qprop and direct influence edges. The most important state distinctions are PV conditions, since a particular global influence structure holds only as long as a particular set of PV conditions holds. Thus, at a minimum, $\mathcal{P}'$ must include all PV conditions and their negations.

QPT classifies PV conditions into two major types. First, a *quantity condition* (QC) is an inequality between a quantity value and a limit point, such as a temperature being greater than a boiling point. Second, a *precondition* (PC) is a discrete boolean condition, such as a pump being on. Generally, dynamics determine when QC's change while external agents (such as operators) determine when PC's change.

#### 2.3.1.1 User Conditions

Often models include views which identify conditions of general interest to the task user, but which have no causal influence on whether $\mathcal{G}$ occurs. Such conditions are never relevant distinctions for any states of a SUDE for $\mathcal{G}$. For example, imagine that our ball-world model includes a user view which is active when the ball's height is at the vertical limit point L which is half way between the heights of ledges 1 and 2. Assume that the ball is on the ground (Y0) in $\mathcal{G}$ and just above the ground and rising in $\mathcal{I}$. In that case, it is irrelevant to $\mathcal{G}$'s reachability from $\mathcal{I}$ whether the ball ever reaches L.

#### 2.3.1.2 Probabilistically-Relevant Conditions

One might argue that some user conditions might provide discriminatory power by partitioning phase space where conditional probabilities of $\mathcal{G}$ occurring change. For example, consider a model of the pump-cycle which includes a view that is active if container `CAN1` is half full. Assume that `CAN1` is empty in $\mathcal{I}$ and overflowing in $\mathcal{G}$ and `CAN1` becoming half full is represented as state $S_H$. $S_H$ occurring does not causally impact whether $\mathcal{G}$ occurs; $\mathcal{G}$ will eventually occur unless the flow from `CAN1` to `CAN2` ends up opposing the pumped flow with an equal rate before the water level in `CAN1` reaches the top of `CAN1`. However, the probability that $\mathcal{G}$ occurs from $S_H$ is higher than from $\mathcal{I}$. That result hinges on the facts that all paths from $\mathcal{I}$ reaching $\mathcal{G}$ must first pass through some refinement of $S_H$ and that not all paths from $\mathcal{I}$ reach $\mathcal{G}$.

Such cases seem rare in QR because it is seldom determinable how conditional probabilities change over qualitatively partitioned phase space. Even assuming that each behavior path is equally likely seldom helps. That is because one typically does not know how many quantitative behaviors map into each qualitative path; often there are an infinite number.

We call a condition *probabilistically relevant* if it always occurs at some point in every path from $\mathcal{I}$ to $\mathcal{G}$ and yet does not always occur after $\mathcal{I}$. We have been able to identify only one type of probabilistic relevance, which we define as follows:

**Definition 2.1 (Transitivity-induced probabilistic-relevance)** *Assume $q = a$ holds in $\mathcal{I}$ and $q = b$ holds in $\mathcal{G}$. For every landmark Li for which $a < Li < b$ necessarily holds, $q = Li$ is either probabilistically-relevant condition for $\mathcal{G}$ or persists forever from $\mathcal{I}$.*

It can be useful to update one's belief that $\mathcal{G}$ will occur once $S_H$ is observed to occur; for example, a controller wishing to prevent $\mathcal{G}$ should become more anxious if $S_H$ occurs. Nevertheless, there are two good reasons why SUDE's should not explicit represent occurrences of $S_H$. First, transitivity-induced probabilistic-relevance is a simple consequence of the Mean-Value Theorem. Thus, it can be reasoned about locally for the particular quantity, external to SUDE representations. Second, not only is `CAN1` being half full not causally significant, it is not especially probabilistically significant either. If we had defined multiple limit points between the bottom and top of `CAN1`, then the water level reaching each one of them would be probabilistically-relevant. Making a commitment to distinguish states by such conditions increases complexity without any clear benefit, simply to be consistent to that obligation.

### 2.3.2 Derivative Conditions

Derivative conditions are of fundamental importance to QR, since without constraining derivatives every QC is always free to change. There are multiple ways in which such constraint can be represented as state distinctions. In this section we explore these alternatives and determine under what contexts each is most relevant.

#### 2.3.2.1 Derivative Inequalities

Derivative inequalities are the most general type of derivative conditions. The constraints imposed by derivative inequalities on the convergence and divergence (i.e. change) of quantity

inequalities are as strong as possible using derivative conditions alone. Specifically, the quantity inequality $x > y$ cannot (continuously) *converge* to $x = y$ if the derivative inequality $d(x) \geq d(y)$ holds. Furthermore, $x = y$ will necessarily *diverge* (instantaneously) to $x > y$ if $d(x) > d(y)$ holds, or to $x < y$ if $d(x) < d(y)$ holds.

Clearly, one never needs to consider a particular $D(Q_1)$ *vs* $D(Q_2)$ if $\mathcal{M}$ never refers to $Q_1$ *vs* $Q_2$. However, we claim that it is actually sufficient (for soundness in computing valid possible changes) to include in $\mathcal{P}'$ only those $D(Q_1)$ *vs* $D(Q_2)$ for which $Q_1$ *vs* $Q_2$ is a QC. This claim is based on our later claim that representing states by their assumption closures is sufficient to capture all local constraints.

#### 2.3.2.2 Ds Values

QPT defines a *Ds* value as the sign (i.e. -1,0,+1) of the derivative of a particular quantity. Since the derivative of any constant is 0, the Ds value for a quantity Q is equivalent to a derivative inequality of the form $D(Q)$ *vs* $D(0)$. For uniformity, we avoid explicit Ds values in favor of their derivative inequality counterparts. This avoids the need for special Ds-based qualitative algebra rules. For example, the constraint Q3 := Q1 + Q2 implies (via qualitative algebra of Ds values) that Ds(Q3)=+1 whenever Ds(Q1)=+1 and Ds(Q1)=+1. In terms of derivative inequalities, that is equivalent to $D(Q1) > D(0) \wedge D(Q2) > D(0) \Rightarrow D(Q3) > D(0)$, which follows from general rules enforcing inequality transitivity over sums.

#### 2.3.2.3 Net-Influences and Higher-Order Derivatives

Despite our avoidance of Ds values, we do adopt the standard QPT convention of introducing *net-influence* terms for each quantity $Q$ which is directly influenced. This involves introducing the quantity inequality `Net-infl`(Q) *vs* 0 to $\mathcal{P}'$ and adding `Net-infl`(Q) := $D(Q)$ to $\mathcal{M}$. The most important consequence of doing so is that the derivative inequality $D$(`Net-infl`(Q)) *vs* $D(0)$ will also be introduced, allowing the consideration of the second-order derivatives of $Q$. For example, these second-order derivative inequalities are the key to avoiding the problem of stutter in the three-container example of [25].

The introduction of quantities such as net-influences which are equivalent to derivatives is how we represent higher-order derivatives in general. For example, the relation between distance, velocity, acceleration, and their derivatives is formulated naturally in QPT in this

way, without the explicit need to refer to terms such as D(D(distance)). The key point is that relevant higher-order derivatives are naturally introduced by QPT models as required. Thus, we need not postulate higher-order derivatives which the QPT model does not introduce; they will not be required for soundness.

#### 2.3.2.4 When To Distinguish States By Derivative Relations

There appear to be only two cases in which distinguishing a state $S$ by a derivative relation $R$ is useful:

1. $R$ is inferable in $S$ or
2. for a transition from $S$ to a possible next state $S_n$, $R$ must necessarily hold at the start of that transition *and* $R$ necessarily persists across that transition and throughout $S_n$.

Case one simply reflects the general rule that knowing the deductive closure of a state is always useful. Case two suggests that it is most useful to introduce and track a derivative inequality across time when it is required to support a change and it then persists unambiguously. We claim that these cases retain all the constraining benefits of derivative relations without the case-split complexities that QS would suffer, such as chattering.

### 2.3.3 Rate Relations

QPT compositionally defines the derivative of a quantity $Q$ as a function of partial derivatives which each correspond to one active influence on $Q$. The process of determining the signs of derivatives is called *resolving influences*. Resolution is trivial when all influences on $Q$ are of the same sign. For mixed influences, resolution requires knowledge about the relative strengths of the negative and positive influences.

For indirect influences, characterizing those relative strengths is difficult. Consider the function $Q := Q1 - 2 \cdot Q2 + 5$, whose qualitative representation as qprops is:

$$\mathrm{Q1} \xrightarrow{Q+} \mathrm{Q}$$
$$\mathrm{Q2} \xrightarrow{Q-} \mathrm{Q}$$

The relation $D(Q1) > D(Q2)$ is insufficient to infer $D(Q) > 0$; $D(Q1) > 2 \cdot D(Q2)$ is required for this case. QPT models typically specify only the qprops for Q, not the exact underlying

function, because it is either indeterminate from available data or even non-existent (when reasoning across multiple similar systems). Thus, we typically cannot appeal to derivative inequalities to resolve mixed indirect influences on Q.

Resolving indirect influences is not as critical as resolving direct influences, for two reasons. First, all quantity changes are ultimately caused by direct influences, making ambiguity at that fundamental level more serious. Second, direct influences tend to reflect specific system structure, whereas indirect influences tend to reflect general physical laws. Thus, whereas physics often happens to define unambiguous indirect influences, structure can be arbitrarily adjusted to force ambiguous direct influences. For example, connecting some new pipes to a container (or opening valves) can cause the direct influences on the mass of water in that container to become ambiguous. In contrast, the indirect influence from that mass to the container's water level remains the same, due to underlying physical relations between mass, density, and volume and between volume, area, and height.

Fortunately, the compositional functions resulting from direct influences are always simple sums, unlike those of indirect influences. QPT calls the partial derivative corresponding to a direct influence a *rate*. Let $\mathtt{D}^+(Q)$ be the sum of the rates of all positive direct influences on quantity $Q$ and $\mathtt{D}^-(Q)$ be the sum of the rates of all negative direct influences on $Q$. The derivative of $Q$ is simply the difference of those two rates:

$$D(Q) := \mathtt{D}^+(Q) - \mathtt{D}^-(Q).$$

For example, if the only influences on Q are:

$$\mathtt{rate}_1 \xrightarrow{I+} \mathrm{Q}$$
$$\mathtt{rate}_2 \xrightarrow{I+} \mathrm{Q}$$
$$\mathtt{rate}_3 \xrightarrow{I-} \mathrm{Q}$$
$$\mathtt{rate}_4 \xrightarrow{I-} \mathrm{Q}$$

then the derivative of Q is defined as follows:

$$D(Q) := (\mathtt{rate}_1 + \mathtt{rate}_2) - (\mathtt{rate}_3 + \mathtt{rate}_4).$$

Knowledge that each rate of $\mathtt{D}^-(Q)$ ($\mathtt{D}^+(Q)$) is dominated by a unique rate of $\mathtt{D}^+(Q)$ ($\mathtt{D}^-(Q)$) is sufficient to resolve the influences on $Q$. Thus, either of the following conditions is sufficient to infer $D(Q) > 0$:

1. $(\texttt{rate}_1 > \texttt{rate}_3) \wedge (\texttt{rate}_2 > \texttt{rate}_4)$,

2. $(\texttt{rate}_1 > \texttt{rate}_4) \wedge (\texttt{rate}_2 > \texttt{rate}_3)$.

We call inequalities such as $\texttt{rate}_1 > \texttt{rate}_3$ *rate relations.* Since mixed direct influences are quite common and rate relations are the only qualitative constraints which can resolve them, it is important to be able to take full advantage of them whenever possible. A rate relation is useful for influence resolution only if it compares rates of the same sign (for opposing direct influences on the same quantity). For that reason, we assume that the QPT model ensures that each rate is positive when its corresponding direct influence holds. [3]

This modelling convention is reasonable because it usually seems natural to distinguish between positively and negatively influencing processes anyway. For example, the distinction between backward and forward flow seems very natural and useful (particularly for qualitative explanations), even though one could model their consequences using one bi-directional process. Sometimes this convention leads to some modelling ackwardness. For example, it is natural to consider any influence on velocity to be due to a single acceleration rate (based on a net force). However, to ensure a positive rate for the negative influence on velocity, we must introduce a local rate with the same magnitude but oppositive sign of acceleration. Interestingly, it seems that avoiding such ackwardness by allowing negative rates in such cases may be acceptable, without loss of ability to resolve influences via rate relations. For instance, there can be no influence on velocity which opposes the influence of acceleration, due to acceleration being the derivative of velocity.

Unfortunately, a particular set of rate relations is seldom useful for resolving the ambiguous influences of a particular state. Thus, distinguishing every state by all rate relations typically has a high cost to benefit ratio, making rate relations ill-suited for QS. For example, QPE supports the optional use of some types of rate relations, but users seldom use that option in practice.

[3]Thus, whereas QPT allows a negative direct influence to be formulated either as an I- with a positive rate or as an I+ with negative rate, we insist on the former.

#### 2.3.3.1 When To Distinguish States By Rate Relations

We advocate a limited use of rate relations in which a rate relation $R$ appears in the description for state $S$ only if one of the following two conditions holds:

1. $R$ is inferable in $S$ or
2. $S_p$ is a possible previous state of $S$, $R$ is inferable in $S_p$, and $R$ necessarily persists for the transition from $S_p$ to $S$.

The main consequence of this approach is that rare relations are only tracked across time when their behavior is unambiguous. We claim that this allows us to retain the constraining benefits of rate relations without the case-split complexities that QS would suffer.

One might argue that introducing case-splits in rate relations at some point might be necessary for global soundness in state paths following that point. Consider an extreme example, where all initial rate relations would persist no matter which case-split occurs. Figure 2.5 illustrates such an example. If $\mathcal{G}$ specified CAN2 overflowing, it might seem useful to refine $\mathcal{I}$ by each of the following alternative conditions:

1. $(\texttt{rate}_{A+} > \texttt{rate}_{A-}) \wedge (\texttt{rate}_{C+} > \texttt{rate}_{C-})$,
2. $(\texttt{rate}_{A+} > \texttt{rate}_{A-}) \wedge (\texttt{rate}_{C+} = \texttt{rate}_{C-})$,
3. $(\texttt{rate}_{A+} = \texttt{rate}_{A-}) \wedge (\texttt{rate}_{C+} > \texttt{rate}_{C-})$,
4. $(\texttt{rate}_{A+} > \texttt{rate}_{C-}) \wedge (\texttt{rate}_{C+} > \texttt{rate}_{A-})$,
5. $(\texttt{rate}_{A+} > \texttt{rate}_{C-}) \wedge (\texttt{rate}_{C+} = \texttt{rate}_{A-})$,
6. $(\texttt{rate}_{A+} = \texttt{rate}_{C-}) \wedge (\texttt{rate}_{C+} > \texttt{rate}_{A-})$,
7. $(\texttt{rate}_{A+} \leq \texttt{rate}_{A-}) \wedge (\texttt{rate}_{C+} \leq \texttt{rate}_{C-})$
   $(\texttt{rate}_{A+} \leq \texttt{rate}_{C-}) \wedge (\texttt{rate}_{C+} \leq \texttt{rate}_{A-})$,
8. $(\texttt{rate}_{A+} > \texttt{rate}_{A-}) \wedge (\texttt{rate}_{C+} < \texttt{rate}_{C-})$,
9. $(\texttt{rate}_{A+} < \texttt{rate}_{A-}) \wedge (\texttt{rate}_{C+} > \texttt{rate}_{C-})$,
10. $(\texttt{rate}_{A+} > \texttt{rate}_{C-}) \wedge (\texttt{rate}_{C+} < \texttt{rate}_{A-})$,

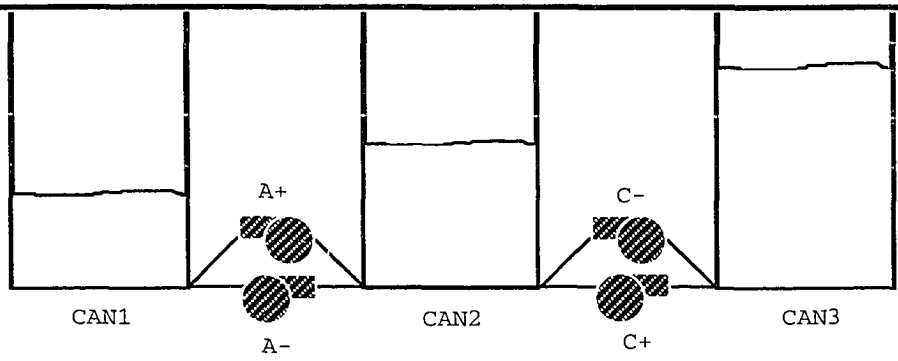


**Figure 2.5**: A four-pump, three-container system

Each pump is always on, pumping at its own constant rate. The water levels of containers CAN1 and CAN3 are assumed to never empty.

11. $(\texttt{rate}_{A+} < \texttt{rate}_{C-}) \wedge (\texttt{rate}_{C+} > \texttt{rate}_{A-})$.

Under any of conditions 1-6, $\mathcal{G}$ is inevitable and under condition 7, $\mathcal{G}$ is impossible. Conditions 8-11 cover the remaining cases, in which $\mathcal{G}$'s reachability from $\mathcal{I}$ is ambiguous. However, in fact, case-splitting $\mathcal{I}$ by $D(\texttt{level}_2) > 0$ and its negation is sufficient to show that $\mathcal{G}$ is either inevitable or impossible from $\mathcal{I}$, depending on whether that one derivative inequality is true or false in $\mathcal{I}$.

So, even in this extreme case, case-splitting on rate relations would be lead to excessive complexity. This example illustrates an underlying observation: it is sufficient to case-split on a single derivative inequality rather than on the set of underlying rate relations from which that derivative inequality could be inferred, in those rare cases where case-splitting on either would be useful in discriminating $\mathcal{G}$'s reachability. Note, however, that the case-split on the status of $D(\texttt{level}_2) > 0$ is useful in this example only because it turns out that it persists forever from $\mathcal{I}$ (satisfying case two of Section 2.3.3.1). For relaxations of this extremely constrained example, say where CAN1 or CAN3 could become empty, reachability of $\mathcal{G}$ would not be discriminated by case-splitting on that derivative inequality.

To complete our discussion of rate relations, we identify some special types of rate relations below. We advocate the same limited use mentioned above for all of them. However, extending our definition of rate relations to include these additional types is necessary to ensure soundness.

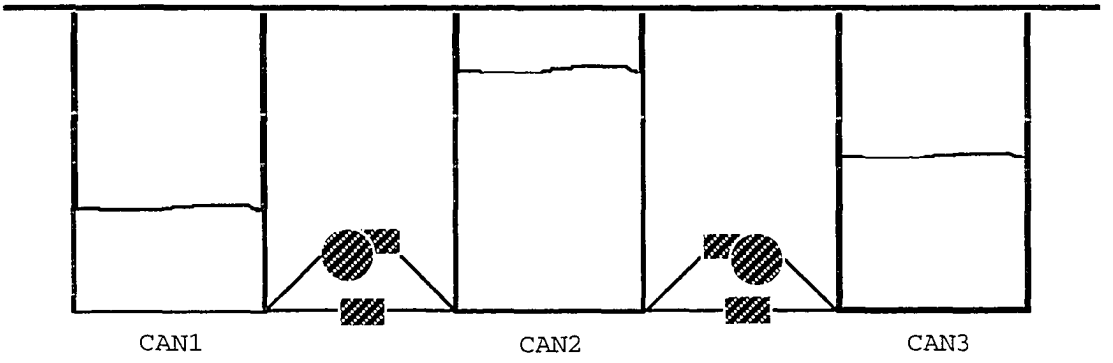


**Figure 2.6**: The two-pump, three-container, pump-cycle system in equilibrium

Furthermore, it is illuminating to explore the rich variety of rate relations that are unmanageable under earlier static-partitioning perspectives.

#### 2.3.3.2 Conjunctive-Rate Relations

In general, resolving the direct influences on a quantity Q may be possible by comparing the sum of an arbitrary subset of $\mathtt{D}^+(Q)$ against the sum of an arbitrary subset of $\mathtt{D}^-(Q)$. For example, consider an extension of our pump-cycle system, as shown in Figure 2.6. Let the goal $\mathcal{G}$ be that CAN2 overflows. Assume that we do not know the initial statuses of the valves (i.e. open or closed) and the pumps (i.e. on or off). With no initial rate relations, $\mathcal{G}$ is possible as long as either of the pumps is on. With initial relations $\mathtt{rate}_{\mathtt{flow-2to3}} > \mathtt{rate}_{\mathtt{pump-1to2}}$ and $\mathtt{rate}_{\mathtt{flow-2to3}} > \mathtt{rate}_{\mathtt{pump-3to2}}$, both pumps must be on for $\mathcal{G}$ to be possible. With initial relation $(\mathtt{rate}_{\mathtt{pump-1to2}} + \mathtt{rate}_{\mathtt{pump-3to2}}) < \mathtt{rate}_{\mathtt{flow-2to3}}$, $\mathcal{G}$ is impossible. We call $(\mathtt{rate}_{\mathtt{pump-1to2}} + \mathtt{rate}_{\mathtt{pump-3to2}})$ a *conjunctive rate.*

#### 2.3.3.3 Virtual-Rate Relations

We introduce the concept of a *virtual-rate* to represent the value that a rate would have if all of the preconditions of its process held. A virtual-rate is defined in a view which is active exactly when all the quantity conditions necessary to define the actual rate hold. In Figure 2.1, view `Primed-Fluid-Flow` gives an example definition of a virtual-rate and view `Fluid-Flow` gives an example usage. We assume that the domain model explicitly identifies virtual-rates by using

the predicate Virtual with a unary argument representing the actual rate. All rate relations involving the actual rate are duplicated with the virtual-rate replacing that rate.

Consider the initial state $\mathcal{I}$ where $\texttt{rate}_{\texttt{pipe}} > \texttt{rate}_{\texttt{pump}} > 0$, which implies $\texttt{level}_1 > \texttt{level}_2$, the valve is open, and the pump is on. If the valve is then closed and the re-opened, $\texttt{rate}_{\texttt{pipe}} > \texttt{rate}_{\texttt{pump}}$ should still hold. Otherwise, it would appear possible that CAN1 could overflow after the valve re-opens, when in fact it cannot.

Virtual rates are particularly useful when they are constants, such as the spec-rates of constant-flow pumps. That is because inferred relations between constant virtual-rates persist to all possible futures, since a relation between two constants never changes. Imagine two such pumps, one pumping into a container and one pumping out. If one observes the level dropping when both pumps are on, one should realize that, in all futures, if the output pump is on then the level will be dropping. Note, however, that things are not so simple if the level was observed to be rising. If the input pump later empties its source container, the level will not be able to rise. [4]

#### 2.3.3.4 QC Rate Relations

Consider any particular QC $Q_1$ *vs* $Q_2$. Corresponding to the generalization from Ds values of $D(Q_1)$ and $D(Q_2)$ to derivative inequality $D(Q_1)$ *vs* $D(Q_2)$, there is a generalization from the relations between the rates influencing $Q_1$ and $Q_2$ individually to the relations between such rate pairs. In example, for the simple influence structure:

$$\texttt{rate}_1 \xrightarrow{I+} Q_1, \texttt{rate}_2 \xrightarrow{I+} Q_2,$$

the rate relation $\texttt{rate}_1 > \texttt{rate}_2$ indicates that $Q_1 > Q_2$ cannot be converging toward $Q_1 = Q_2$. We call such rate relations *QC rate relations.* Whereas other rate relations can be useful to resolve the Ds value of a particular quantity, QC rate relations might be useful to resolve how an entire quantity inequality is changing. Such resolution occurs when the derivative inequality of that quantity inequality can be inferred from the available QC rate relations.

For simple examples such as the one above, there is no point in including the QC rate relation in $\mathcal{P}'$, since its status is logically equivalent to that of the corresponding derivative inequality.

[4] In reality, the pump rate would drop below the spec-rate before it reaches zero, at some point before the source empties. Modelling this behavior requires two modes for the pump, one where the spec-rate is constant and one where is it a linear function of the level in the source (when the level is low).

Unfortunately, useful QC rate relations appear to be extremely rare, for two reasons. First, one quantity in many QC quantity pairs is a constant limit point, which has no influences. Second, QC quantities are often not directly influenced. For example, in our pump-cycle example, flow rates directly influence water mass, which indirectly influences the pressure differences that define the flow process QC's. In fact, we have not been able to formulate an example within our test domains in which QC rate relations are useful; we mention them simply for thoroughness.

### 2.3.4 Duration

Finally, states can be distinguished by qualitative measures of duration. The simplest, and most common, distinction is instantaneous (`Instant`) versus non-instantaneous (`Interval`) states. `Interval` states can be further distinguished as either infinite or finite in duration, depending on whether the state could possibly last forever. Some QR frameworks introduce explicit propositions to distinguish finite and infinite durations. However, we have found it sufficient, and representationally elegant, to indicate that a state $S$ could last forever via an explicit transition from $S$ to $S$. We call these *self-transitions.*

Traditionally, many QR frameworks characterize each possible behavior over time as a strict alternation of `Instant` and `Interval` states. However, since we do not adopt the traditional precision-based perspective, we are merely interested in distinguishing by duration when that has practical import in discriminating goal-reachability. A major motivation in traditional QR for distinguishing between `Instant` and `Interval` durations is to disallow (interval) changes *to* equality with limit points simultaneous with (instant) changes *from* equality with limit points. However, we argue in the next chapter that coincidental changes can be safely ignored in discrimination-based QR. Thus, our framework views simultaneous instant and interval changes as coincidental, eliminating the need for partitioning phase space by duration distinctions. In our framework, the most important duration distinction is to identify possible self-transitions.

## 2.4 Types of States

There are several useful classifications of states that we will employ throughout this thesis. They are defined as follows:

**Definition 2.2 (Exact States)** *An* exact *state is a point in phase space. It represents a conjunctive set of exact quantitative values for every system parameter of the model. Thus, there are an infinite number of exact states. For a deterministic model, an exact state always has exactly one next state and one previous state.*

**Definition 2.3 (Qualitative States)** *A* qualitative state *is a region in phase space representing a conjunctive set of qualitative propositions of $\mathcal{P}$. We assume that the useful space of qualitative states is defined by the powerset of $\mathcal{P}'$. In this work we restrict this space to be finite by assuming that $\mathcal{P}'$ is finite.*

Unless otherwise noted, we use the term "state" to refer to a qualitative state. When we refer to a state, we are generally referring to the set of propositions logically inferable in it. Thus, for example, when we say that one state is a subset of another, we are saying that its propositions are a subset of the others. [5]

**Definition 2.4 (Consistent State)** *A state S is called* consistent *if it is logically consistent; otherwise it is called* inconsistent.

**Definition 2.5 (Complete and Partial States)** *A state is called* complete *if it contains either the negative or positive version of every proposition in $\mathcal{P}'$. Otherwise, it is called* partial. *Essentially, each consistent complete state represents a valid "model" or "interpretation" of classical propositional logic (for the set of literals $\mathcal{P}'$).*

Typically, each qualitative state corresponds to an infinite set of exact states. Even a complete state can correspond to an exact state only if it implies that each quantity is at a named limit point; this typically will be true only for some equilibrium states.

**Definition 2.6 (State Abstraction, Refinement, and Completion)** *An* abstraction *of a state S is any state which is a subset of S. A* refinement *of S is any state which is a superset of S and does not contain the negation of any proposition in S. These definitions include S itself (unless prefaced by "proper"). By definition, a refinement of S also cannot contain the negation of any proposition in S. A* completion *of S is a refinement which is complete.*

[5] Note that there is a reasonable alternative interpretation of state subset: one state's phase space is a spatial subset of another's. That corresponds to a superset relation between state propositions.

**Definition 2.7 (State Closures)** *The* closure *of a state $S$ is the logically complete set of propositions implied by $S \cup \mathcal{M}$. We denote the closure of $S$ as $\mathcal{C}_P(S)$. Whereas we restrict the domain of $S$ to $\mathcal{P}'$, the domain of $\mathcal{C}_P(S)$ is all of $\mathcal{P}$. $S$ is inconsistent exactly if $\mathcal{C}_P(S)$ contains $\perp$.*

**Definition 2.8 (Equivalent States)** *Two states are* equivalent *exactly if both have the same set of consistent refinements. That is, two states are equivalent exactly if their closures are identical.*

**Definition 2.9 (Compatible States)** *Two states are* compatible *exactly if their union is a consistent state. Equivalently, two states are compatible exactly if both are abstractions of some consistent complete state.*

**Definition 2.10 (Base State)** *The* base state *represents the set of all propositions that are true in the scenario model, as well as any closed-world assumptions. We denote the base state as $\mathcal{B}$. By default, a given state is assumed to be a refinement of $\mathcal{B}$.*

## 2.5 Reasoning About Closures of Partial States

A major claim of this thesis is that generating all consistent completions of a partial state is generally excessive for QR tasks. However, generating the closure of a partial state is particularly useful. Closures are sufficient for the following key operations on a given state $S$:

1. detecting whether $S$ is inconsistent;
2. detecting whether $S$ is equivalent to other known states;
3. maximizing local explanatory precision for state $S$.

Detecting inconsistent and equivalent states improves search efficiency because inconsistent states do not have to be considered and only one representative of a set of equivalent states needs to be considered. $S$'s closure provides the maximal explanation of what propositions must hold if $S$ occurs.

Unfortunately, propositional satisfiability is NP-complete; thus, so is the general problem of computing closures (i.e. via refutation proofs). In this section we explore this issue and

argue that reasonable ontological constraints on QPT models can be exploited to address this problem.

### 2.5.1 The Sufficiency of Closures for Computing Next States

Perhaps the most important use of closures is for defining what propositions must hold after a given transition between two partial states. For example, let state $S$ be $\{A > B, B = C, D(B) < D(C)\}$; its closure contains $A > C$, via transitivity. Standard QR temporal constraints suggest that both $A > B$ and $A > C$ should persist to the next state of $S$, since $D(B) < D(C)$ indicates that $B = C$ is instantaneously moving away from equality to $B < C$. However, $A > C$ is not in the closure of $\{A > B, B < C, D(B) < D(C)\}$; thus it will not be realized to persist unless the temporal constraints are applied to the closure of $S$.

Thus, in lieu of reasoning with completions of $S$, we must at least reason with the closure of $S$ in order to soundly capture the temporal constraints of QR over time. To analyze the sufficiency of closures, let us first define a few useful types of next state sets.

**Definition 2.11 (Possible complete nexts)** *Let the* possible complete nexts *of a state $S$ be the set of all next states of all consistent completions of $S$. These next states are all consistent and complete.*

**Definition 2.12 (Transition complete nexts)** *Let the* transition complete nexts *for a transition $S_i \rightarrow S_j$ be the states in $S_i$'s set of possible complete nexts that are refinements of $S_j$. The other possible complete nexts represent other transitions from $S_i$. Let the* transition complete successor *be the intersection of all of the transition complete nexts.*

**Definition 2.13 (Possible closure nexts)** *Let the* possible closure nexts *of a state $S$ be the set of the closures of the next states of the closure of $S$.*

**Definition 2.14 (Transition closure nexts)** *Let the* transition closure nexts *for a transition from $S_i$ to $S_j$ be the states in $S_i$'s possible closure nexts that are compatible with $S_j$. The other possible closure nexts represent other transitions from $S_i$. Let the* transition closure successor *be the closure of the union of $S_j$ with the intersection of all of the transition closure nexts.*

Note that the transition closure successor, unlike the transition complete successor, must be unioned and closured with $S_j$. This is because the transition closure nexts are not necessarily complete — so, they may not already contain all propositions in $S_j$.

Now, consider what it would mean for closures to be insufficient for maximally defining the propositions which necessarily hold after a transition from a state $S_i$ to a state $S_j$. Closures would be insufficient exactly if the transition complete successor contains some propositions which the transition closure successor does not.

For the closure nexts to conceivably be insufficient, there must be some disjunction which is logically implied by $S_i$ but none of its disjuncts are in the closure of $S_i$. Let us consider the simplest case, where the disjunction is $X \vee Y$. Let $N$, $N_X$ and $N_Y$ be the transition closure successors for transitions $S_i \rightarrow S_j$, $S_i \cup \{X\} \rightarrow S_j$, and $S_i \cup \{Y\} \rightarrow S_j$, respectively. Clearly, $N$ can contain propositions that the closure of $S_j$ does not — such as $A > C$ in the previous example. The real issue is whether $N_X \cap N_Y$ can contain propositions that $N$ does not. If so, then closure nexts are insufficient.

We claim that $N_X \cap N_Y$ cannot contain propositions that $N$ does not. It appears that counter examples would have to be of the following sort. Let $X$ be $D(A) > 0$ and let $Y$ be $D(A) < 0$; thus, $X \vee Y$ represents $D(A) \neq 0$. Let $S_i$ contain $A = 0$ and enough other propositions to somehow imply $X \vee Y$. $A \neq 0$ should be true immediately after $A = 0$ and $D(A) \neq 0$. Yet, $N$ will not contain a proposition for $A \neq 0$. However, neither will $N_X \cap N_Y$, because $A \neq 0$ is itself a disjunction in our soft inequality representation scheme. This is not simply an oversight; we argue that $\neq$ derivative relations are not appropriate for defining states exactly because they lead to these sorts of disjunctions. Instead, we insist that states at least commit themselves to $\geq$ or $\leq$ derivative relations when $\neq$ derivative relations would hold.

### 2.5.2 The Adequacy of Assumption Closures

A particularly useful subset of state closures is:

**Definition 2.15 (Assumption Closures)** *The* assumption closure *of a state $S$, denoted as $\mathcal{C}(S)$, is the subset of its state closure $C_P(S)$ defined by $C_P(S) \cap \mathcal{P}'$. Each complete state is trivially equivalent to its assumption closure.*

We explicitly associate the assumption closure with each state. For efficiency, we approximate state equivalency and consistency in terms of assumption closures, instead of explicitly referring to state closures. Since we define $\mathcal{P}'$ so that it contains all potentially relevant state distinctions, assumption closures are sufficient for our purposes. The key issue is under what conditions we can tractably compute logically complete assumption closures.

### 2.5.3 Computing Assumption Closures Via Completions

A simple way to compute the assumption closures of a state $S$ is to first compute all consistent completions of $S$ and then take the intersection of all of them. [6] We call this the *completion-intersection* approach. There are four (overlapping) cases when this approach can be efficient:

1. $\mathcal{P}'$ is small;
2. $S$ is almost complete already;
3. the total number of consistent completions of $S$ is small;
4. all consistent complete states have been precomputed.

Implicitly, this is how assumption closures are available from AE's — via analysis of brute-force, exhaustive case-splits.

When the closure of $S$ is much smaller than any completion, as is typically the case, it can be much more efficient to compute the closure based on refutation proofs. Since we determine consistency by finding some consistent completion, we call this the *completion-refutation* approach:

1. Let $X = S$ (i.e. the set of assumptions initially defining state S).
2. Let $InferChoices = \mathcal{P}'$.
3. Let $CheckChoices = \mathcal{P}'$.
4. If $\exists P \in InferChoices$ such that there is no consistent completion $C$ (over $CheckChoices$) of $X + \{\overline{P}\}$, then add $P$ to $X$ and repeat Step 4.

[6] The process of generating all consistent completions is equivalent to what is called *interpretation construction* in the ATMS literature.

5. $X$ is the assumption closure of $S$.

Determining whether a state is consistent typically requires much less work than generating all completions. First, only one consistent completion is required to show that a state is consistent. Second, if a state is inconsistent, most inconsistencies will be detected during the completion process well before each (inconsistent) completion has been fully generated.

Although we originally explored incorporating such techniques into our ATMS implementation to obtain assumption closures, we discovered over time that all observed instances in which they helped appeared to actually be special cases that could be characterized. After briefly discussing the logical completeness of ATMS's, we discuss these characterizations below. Whereas the above methods simply mirror the effects of logical resolution (in an indirect way), our attempt below is to understand when we can avoid the general complexities of logical resolution completely for our particular purposes.

### 2.5.4 Approximating Closures Via Boolean Constraint Propagation

Boolean constraint propagation (BCP) [11] is an efficient means of deduction which is commonly used in truth maintenance systems, including ATMS's. It is logically equivalent to finding unit clauses via unit resolution. Thus, BCP is sound and is refutation complete for positive literals when all clauses are Horn.

Unfortunately, BCP is not necessarily refutation complete for an arbitrary collection of non-Horn clauses. This incompleteness is due to BCP's failure to consider case-splits. To converge towards logical completeness using BCP, one must augment a given collection of clauses with some of those that would have resulted from the full resolution rule. In order to better understand this issue, let us now analyze the nature of BCP more formally.

We adopt a view of BCP based on primitives called *justifications*, which allows for directed reasoning:

**Definition 2.16 (Justifications)** *A* justification *defines a conjunctive set of antecedent propositions and a single consequent proposition which is logically implied by that set. Each proposition in a justification may represent a negative or a positive literal.*

For implemented rules, we represent justifications as

(==> (and *antecedent-1 ... antecedent-n*) *consequent*),

or simply

(==> *ante consequent*)

when there is only one antecedent.

Computing the closure of a state $S$ via BCP can be simply characterized as follows. Start with a set of propositions $C = S$ and grow $C$ by the consequent of any justification whose antecedents are all currently in $C$; iterate until no further growth in $C$ is possible. [7]

This justification-based BCP algorithm can easily be extended to general clauses by interpreting each clause as multiple justifications:

**Definition 2.17 (Clauses and Clausal Justifications)** *We use the term* clause *to refer to a clause of classical propositional logic. A* clausal justification of a clause *is a justification in which one of the clause propositions is viewed as the consequent and the others are viewed as negated antecedents. For example, one clausal justification of clause* $A \vee \overline{B} \vee C$ *is* $\overline{A} \wedge B \Rightarrow C$. *BCP effectively interprets a clause of N propositions as all N of its clausal justifications. We further define the* clausal justifications of a justification *to be all of the clausal justifications of the clause corresponding to that justification.*

For efficiency, most BCP implementations (including ours) treat clauses as one unit, not as multiple justifications. [8] In fact, one could instead adopt a clause-based view of BCP and interpret justifications as their corresponding clauses. However, we wish to retain the directedness of justifications. In any case, a justification-based view of BCP is sufficient for discussing the logical properties of BCP, and so we adopt that here.

**Definition 2.18 (BCP-completeness)** *The full level of completeness offered by BCP can be achieved by interpreting each justification and clause as their clausal justifications. We call this* BCP-completeness. *This is equivalent to the degree of completeness provided by unit resolution.*

[7] This process is made extremely efficient by associating a counter with each justification, representing the number of antecedents not yet in $C$. By decrementing the counter whenever one of the antecedents gets added to $C$, the consequent is deduced as soon as the counter goes to zero. Thus, the worst-case complexity of this process is linear in the total number of propositions in all of the justifications.

[8] The clause counter starts out being the size of the clause and gets decremented whenever the negation of a literal in that clause gets added to $C$. When the counter gets to one, the literal in that clause whose negation is not in $C$ is the one that is inferred.

**Definition 2.19 (BCP-closure)** *The* BCP-closure *of a set of propositions is the closure of that set that is deduced by BCP for a given set of justifications.*

**Definition 2.20 (BCP-consistency)** *A set of propositions is* BCP-inconsistent *if* $\perp$ *is in its BCP-closure. Otherwise, that set is* BCP-consistent.

**Definition 2.21 (Interpretation-completeness)** *An assumption closure of a state* $S$ *is* interpretation-complete *exactly when it contains the same assumptions as the intersection of all BCP-consistent completions of* $S$.

**Definition 2.22 (Approximate State Closures)** *The* approximate closure *of a state* $S$ *is the set of assumptions which are in the BCP-closure of* $S$. *We denote the approximate closure of* $S$ *as* $C'(S)$. *In general, we desire such closures to be interpretation-complete.*

One way to improve the completeness of BCP is to add a *nogood clause* [10] whenever a state is recognized as inconsistent. For example, the nogood clause due to a state $\{A, \overline{B}, C\}$ whose BCP-closure is found to include $\overline{C}$ would be $\overline{A} \vee B \vee \overline{C}$. The main utility of nogood clauses is for caching information deduced via proof-by-refutation during interpretation constructions, as case-splits are explored. This technique can be particularly useful if it results in only a few, small nogood clauses. However, it is typically not suitable for our task of generating the assumption closure of a state $S$. Guaranteeing that the computed closure of $S$ is interpretation-complete would require explicitly generating all completions of $S$ anyway. So, caching nogood clauses when computing closures for other states would simply help reduce the amount of backtracking that would occur during interpretation construction on top of $S$.

BCP can be made complete by using all *prime implicates*. The prime implicates are identical to the clauses produced by full resolution, after discarding all subsumed clauses [11]. Thus, BCP-completeness is equivalent to logical completeness when the clauses represent all prime implicates.

Although we have explored the use of the above two techniques, we have found it more beneficial to try to characterize the types of logical incompleteness that can result from applying BCP to QPT models. This is the topic of the next section.

### 2.5.5 Exploiting the Nature of QPT Models

Let $\mathcal{J}_0$ denote the base justifications specified by the QPT scenario model $\mathcal{M}$, where any clauses in $\mathcal{M}$ are specified as their clausal justifications. For comparison, let $\mathcal{C}_0$ denote the clauses corresponding to $\mathcal{J}_0$. Let $\mathcal{J}$ denote a superset of $\mathcal{J}_0$ which is sufficient for BCP to generate the interpretation-complete assumption closure of any partial state $S$ specified from the set $\mathcal{P}'$.

In general, $\mathcal{J}$ might require the clausal justifications of an arbitrary number of resolvents of $\mathcal{C}_0$, to account for case-splits. However, we claim that for QPT models one can do much better. In fact, we claim that through careful analysis of the local nature of QPT models, we can encode QPT models such that $\mathcal{J}$ is typically identical to $\mathcal{J}_0$. For cases where this does not hold, we argue that principled, local case-split search appears to be suffice. Although we have not been able to prove the sufficiency of this approach, we will justify our intuitions, based on both empirical and analytical evidence.

The key observation behind our argument is that most qualitative constraints do not appear to yield disjunctions which could lead to larger computed closures if case-splits were searched. Many disjunctions which can be produced via resolution are not useful for implying new propositions. For example, consider transitivity inferences for a model in which interesting quantity comparisons include: A vs B, B vs C, C vs D, and A vs D. For a state containing $A > B$ and $B > C$, the constraint $(B > D) \Leftrightarrow (A > D)$ also holds. However, we have never come across, nor been able to conceive of, any physical model for which resolving such a constraint could lead to a larger closure.

In the simplest case, case-splitting is required to infer $C$ given constraints like:

$$A \vee B, A \Rightarrow C, B \Rightarrow C.$$

Binary qualitative constraints such as same-relations fit this pattern. For example, consider `Same-Rel`$((volume, 0), (level, bottom))$. One clause implied by that same-relation is $(volume > 0) \vee (level \leq bottom)$, which could theoretically play the role of $A \vee B$ above. The issue is under what conditions there would then exist analogs to both $A \Rightarrow C$ and $B \Rightarrow C$ in a QPT model. Although we can easily make up an artificial domain in which such rules exist, we have not been able to conceive of any such C for any real domain true to the semantics of *volume* and *level*.

```
(defScenario CAN-WITH-SIDE-PORT
  (State LIQUID)
  (Substance WATER)
  (Container CAN1)
  (Portal (PORT1 CAN1)))

(defState S1
  (> (height (liquid-in CAN1)) (height (PORT1 CAN1))))
```

**Figure 2.7**: Scenario `CAN-WITH-SIDE-PORT` and state S1

State S1 should have `(Positive (mass (C-S WATER LIQUID CAN1)))` in its closure because there is only one contained-liquid in `CAN1` (for substance `WATER`).

The intuition is that this phenomena is not coincidental to those particular quantities. Fundamentally, "magical", non-obvious local inferences arising only due the case-splitting provided by resolution do not appear likely, in large part because of the strong commitment that QPT makes to specific causal orderings. QPT's organizational structure reflects a natural flow of constraints from QC's and PC's to derivatives, through influence structure. Indeed, it seems reasonable to argue that a QPT model for which BCP is insufficient to compute a state closure can probably be reformulated to make BCP sufficient, while at the same time making the model more natural as well. We will explore this intuition via the case studies presented below, towards developing a principled basis for such model formulations.

#### 2.5.5.1 Case Study 1: Inferring Positive Mass from Positive Level

Consider state S1 for a simple scenario, as defined in Figure 2.7. Figure 2.8 highlights some key constraints in the resulting scenario model. In particular, note that for this example we are using the defEntity of Figure 2.9 for containers. It infers whether the volume of liquid in a container is positive or zero based on whether there is any contained-liquid of positive mass in the container.

Due to justifications (5), (2) and (4), BCP deduces:

`(> (height (liquid-in CAN1)) (height (bottom CAN1)))` from
`(> (mass (C-S WATER LIQUID CAN1)) 0)`.

Similarly, due to justifications (6), (1), and (4), BCP deduces:

```
from (Container CAN1) defEntity (of Figure 2.9):
(1)  (==> (not (Wet CAN1)) (<= (volume (liquid-in CAN1)) 0))
(2)  (==> (Wet CAN1)       (>  (volume (liquid-in CAN1)) 0))

from (Heighted-Container CAN1) defEntity:
(3)  (Qprop+ (height (liquid-in CAN1)) (volume (liquid-in CAN1)))
(4)  (Same-Relation ((height (liquid-in CAN1)) (height (bottom CAN1)))
                    ((volume (liquid-in CAN1)) 0))

from (Wet CAN1) defView:
(5)  (==> (>  (mass (C-S WATER LIQUID CAN1)) 0) (Wet CAN1))
(6)  (==> (<= (mass (C-S WATER LIQUID CAN1)) 0) (not (Wet CAN1)))

from (Contained-Liquid (C-S WATER LIQUID CAN1)) defEntity:
(7)  (Qprop+ (volume (C-S WATER LIQUID CAN1)) (mass (C-S WATER LIQUID CAN1)))
(8)  (Same-Relation ((volume (C-S WATER LIQUID CAN1)) 0)
                    ((mass (C-S WATER LIQUID CAN1)) 0))
(9)  (Qprop+ (volume (liquid-in CAN1)) (volume (C-S WATER LIQUID CAN1)))

from (Side-Portal (PORT1 CAN1)) defView:
(10) (> (height (PORT1 CAN1)) (height (bottom CAN1)))
```

**Figure 2.8**: Some scenario model constraints for CAN-WITH-SIDE-PORT

These constraints follow from applying scenario CAN-WITH-SIDE-PORT of Figure 2.7 to our domain theory for the tank-world. Note that constraint (6) comes from closing the (Wet CAN1) proposition, since there is only one contained-liquid for CAN1.

```
(defEntity Container
  ...
  ;;; additional "causality skipping" relations:
  (Fact-to-Close! (Wet ?self))  ;; in case no contained-liquid instantiated
  (when (not (Wet ?self)) (Zero (volume (liquid-in ?self))))
  (when (Wet ?self)       (Positive (volume (liquid-in ?self)))))

(defView (Wet ?can)
  Individuals ((?cl :type Contained-Liquid :form (C-S ?sub LIQUID ?can)))
  Conditions  ((Positive (mass ?cl))))
```

**Figure 2.9**: Container defEntity augmented with "causality skipping" constraints

(= (height (liquid-in CAN1)) (height (bottom CAN1))) from

(= (mass (C-S WATER LIQUID CAN1)) 0).

However, BCP will not be able to make the reverse deductions based on those justifications.

Being able to infer that the mass is positive, given that the level of liquid is positive, is particularly important for our thermodynamics test domain. Without knowing that the mass is positive, portal PORT1 will not be known to be Exposed-To the contained-liquid of water in CAN1. That Exposed-To is a prerequisite for the Fluid-Flow view which enables the process of water flowing out through PORT1.

If justifications (1), (2), (5), and (6) are interpreted as clausal justifications, then BCP is able to make those reverse deductions. Since the number of clausal justifications corresponding to all clauses and justifications is at most the total number of propositions in those clauses and justification, this may appear to be a relatively harmless way to achieve BCP-completeness. However, the complexity of an ATMS does not grow polynomially in the number of justifications. Yet, we wish to use an ATMS to efficiently cache our inferences about state closures across all partial states simultaneously. Therefore, we carefully consider which justifications must actually be interpreted as clausal justifications in order to achieve BCP-completeness for QPT models. Furthermore, this analysis helps us understand the general case, where using clausal justifications is still not sufficient.

It should be noted that this issue typically does not arise when reasoning with complete states. Justifications (1), (2), (5), and (6) are sufficient to indicate that having positive level but non-positive mass is inconsistent. Thus, any consistent complete state which contains (> (height (liquid-in CAN1)) (height (bottom CAN1))) will also contain (> (mass (C-S WATER LIQUID CAN1)) 0). The issue of which justifications to interpret as clausal justifications is still important for complete states, since BCP-completeness helps the state completion process avoid exploring branches that would eventually lead to inconsistent complete states. However, in that case the issue is one of efficiency, not logical completeness.

Constraints (7), (9), and (3) of Figure 2.8 define the following causal chain:

(mass (C-S WATER LIQUID CAN1)) $\stackrel{Q+}{\rightarrow}$
(volume (C-S WATER LIQUID CAN1)) $\stackrel{Q+}{\rightarrow}$
(volume (liquid-in CAN1)) $\stackrel{Q+}{\rightarrow}$
(height (liquid-in CAN1)).

Note that Figure 2.9's use of the status of the (Wet CAN1) proposition to define the relation between (Volume (liquid-in CAN1)) and 0 effectively skips some links in this causal chain. It jumps from (mass (C-S WATER LIQUID CAN1)) to (volume (liquid-in CAN1)) directly, skipping over (volume (C-S WATER LIQUID CAN1)).

Since such non-causal tricks are sufficient for BCP-completeness for complete states, they have been commonly used as modelling conveniences. In fact, the model fragment of Figure 2.9 was originally used in models we had developed for QPE. Redefining (Wet CAN1) so that its quantity condition is (Positive (volume (C-S WATER LIQUID CAN1))) instead of instead of (Positive (mass (C-S WATER LIQUID CAN1))) would properly reflect causality. However, that would introduce new state assumptions that are not required in any other part of the model. Since the complexity of the ATMS is proportional to the number of state assumptions, modellers strive to avoid introducing quantity conditions which are not of general use.

A much better solution is to explicitly model the volume of liquid in a container as the sum of all of the volumes of the contained-liquids. Quantitatively, this assumes that the contained-liquids do not partially dissolve into each other. However, qualitatively, this sum relation is appropriate as long as no increment of a contained-liquid would fully dissolve into the existing contained-liquids.

For our example above, closing on sums leads to the following constraint:

(11) (Same-Relation ((Volume (liquid-in CAN1)) 0) ((Volume (C-S WATER LIQUID CAN1)) 0)).

Together, constraints (8), (11), and (4) bidirectionally relate any two quantities in the causal chain described above, with (11) bridging the gap between (8) and (4).

Adding constraint (11) allows us to handle this example without the "causality skipping" constraints (1) and (2) and without interpreting any justifications as causal justifications.

```
(defScenario CAN-WITH-WATER-AND-OIL
  (State LIQUID)
  (Substance WATER)
  (Substance OIL)
  (Container CAN1))

(defState S2
  (> (height (liquid-in CAN1)) (height (top CAN1))))
```

**Figure 2.10**: Scenario CAN-WITH-WATER-AND-OIL and state S2

State S2 should have (Increasing (height (liquid-in OUTSIDE))) in its closure, because *some* contained-liquid in CAN1 is above the top of CAN1 and overflowing.

#### 2.5.5.2 Case Study 2: Case-Splits Needed For Multiple Substances

Unfortunately, avoiding causality skipping constraints (by using closed sums) is not sufficient for ensuring that assumption closures are interpretation-complete. Consider a modification to the previous example in which there are two substances, water and oil, as defined in Figure 2.10. Since some liquid is overflowing in state S2, the liquid level of OUTSIDE will be rising, even though we do not know whether it is oil or water (or both) which is overflowing. Without deducing that the level outside is rising, we will not realize that any next state of S2 will necessarily have a positive level of liquid in OUTSIDE.

Note that we cannot simply infer that the "mass" of liquid in CANA is positive, analogous to our solution for case study 1. First, there is no concept of mass of liquid in CANA, only for particular liquid substances. Second, and more importantly, the level of liquid outside is only influenced by the level of liquid in CAN1 when one of the influence structure's alternative conditions for that influence holds, yet no one such condition holds in S1.

Making this deduction fundamentally requires a case-split on at least one of the contained-liquids being Exposed-to the top of CAN1. Committing to either Exposed-to view being active is sufficient to know that there is a positive resolved influence on the level of liquid outside. [9] Furthermore, boths views being inactive is inconsistent with S2's assertion that the liquid level

[9]Our test domain model states that the outside level of liquid is always below any named non-bottom level (such as tops), reflecting an assumption that the cross-sectional area of OUTSIDE is effectively infinite.

is above the top. Thus, the issue is what justifies us to focus on this case-split, over all of the other case-splits that resolution would consider.

The underlying justification comes from the way in which inequalities referencing composed quantities are fundamentally different from other propositions. Composed quantities are quantities which are defined across multiple views/processes, such as derivatives. Determining whether a composed inequality relation holds requires considering the disjunction of the closed set of alternative conditions defining its negation. Although composed qualitative constraints are usually especially ambiguous, demonstrating that no such alternative condition holds is sufficient to infer the inequality relation.

Thus, we infer `(> D(height(liquid-in OUTSIDE)) 0)` (call it relation R) in S1 via proof by refutation as follows. First, we identify the closed set of alternative QC/PC conditions in which that relation is possible. [10] Second, we check if there is any set consisting of one conjunct from each alternative such that the conjunction of their negations is consistent with S1. Since there is not, we infer R in S1, because it is impossible for its negation to hold in S1.

[10]Section 4.3.1 describes how this is done. Essentially, it involves back-chaining through the justification structure defined by Figures 4.5 and 4.6, starting at (`Possibly` R) and grounding out at QC's and PC's.

# Chapter 3

# Computing State Transitions

Computing possible state transitions is the heart of qualitative simulation. In fact, it is fundamental to any sort of qualitative reasoning about time. To properly reflect the ambiguity inherent in qualitative models and states, it is important that one be able to generate a complete set of possible transitions out of a given state. At the same time, it is important to be able to keep unsound transitions out of that set — both for precision of inference and for keeping the branching factor more manageable. Thus, the QR community has expended much effort in attempts to develop sound and complete techniques for generating transitions.

Unfortunately, existing techniques generate transition sets which are neither sound nor complete, with respect to the gold standard of quantitative simulation. In particular, they are unsound because they predict that two independent changes may coincidentally occur at the same instant. Such coincidences do not follow as mathematical consequences of the idealized qualitative model (since they have probability of zero in the limit) nor will they ever be observed in the real world (without observational uncertainties which would also suggest that one change could have occurred first). [1] Furthermore, existing techniques are incomplete because they do not predict all of the possible effects of a discontinuous change, such as when a valve is instantly opened. Few approaches even allow discontinuous changes; those that do (such as in [26]) will sometimes improperly prefer continuous changes over discontinuous changes.

[1] Our claim that coincidental changes are unsound with respect to prediction does not mean that conceiving of them is useless. We discuss this point in Section 3.2.2.2

We begin this chapter by presenting a generalization of the standard, continuity-based means of computing transitions. After explaining the incompleteness and unsoundness of that approach, we explore approaches based on minimal change which overcome those problems. Finally, we contrast our approach with those of the minimal change community, arguing that our QR perspective adds to the general understanding of the proper roles of continuity and minimality in formal theories of change.

## 3.1 Continuity-Based Change

There are many algorithms for computing the complete set of next states which could result from continuous changes from a given state $S$, including those described in [12, 23, 27, 39]. Despite their algorithmic and ontological differences, they all compute essentially the same set of next states as the following generic algorithm does:

1. Let $S_b$ be the set of base propositions that any next state of $S$ must be consistent with.

2. For each proposition $p$ in $S$ whose negation is consistent with $S_b$, consider each consistent completion of $S_b \cup \{\overline{p}\}$ to be a next state of $S$.

In short, step 1 defines the base constraints on the next states and step 2 considers what next states could occur if at least one proposition in $S$ changes status (i.e *flips*).

For specific techniques, the completions computed in step 2 are often slightly different from classical propositional logic interpretations. New propositions may be created to serve as historical markers, due to landmark introductions [39]. Also, known propositions may become irrelevant (or relevant), due to existence changes [27], and thus should not be (or should be) included in the completions. [2]

Nevertheless, such details do not change the basic nature of this class of algorithms: the base set of next propositions $S_b$ is independent of any particular transition, being based solely on constraints such as continuity which are local to individual quantities and their derivatives.

[2] Furthermore, for algorithms such as QPE which explicitly compute all complete states up-front, a state $S$ can be completed simply by gathering all known complete states which are supersets of $S$.

### 3.1.1 Base Temporal Constraints

Table 3.1 concisely generalizes the standard continuity-based rules which define the base constraints on the next state. For example, these seven rules subsume all twenty cases in Kuiper's table of legal transitions without landmark introductions (i.e. Table 2 of [39]). Using these rules, the base propositions of the next state are defined by:

$$S_b \equiv \{p \mid \texttt{Next}(p) \in S\} \cup \mathcal{B},$$

where $\mathcal{B}$ is the base state of fixed background knowledge. Thus, the unary argument of any proposition of predicate `Next` denotes a fact which necessarily holds in any continuous next state of S.

Only continuous changes due to dynamics are considered in this approach. Thus, $\mathcal{B}$ must define the status of each proposition which is not a relation between quantities or their derivatives nor always inferable from such relations and $\mathcal{B}$. For example, this typically means that the status of all QPT preconditions must be defined in $\mathcal{B}$.

Rules F1 and F2 integrate two constraints: continuity and the standard QR constraint that no change to equality can occur at the same time as a change from equality. Note that further constraints follow from these rules due to the relations between soft and hard inequalities. For example, the following fundamental continuity rule follows from rule F2:

$$\texttt{Interval} \wedge N_i > N_j \Rightarrow \texttt{Next}(N_i \not< N_j).$$

Rules F3 and F4 encode derivative consistency — the constraints that derivatives have on their quantities. Finally, rules F5, F6, and F7 enforce classical continuity. This requires a strict alternation of `Instant` and `Interval` states, allowing what is called *unstable equilibrium*. Disallowing unstable equilibrium requires omitting rule F5 (to allow an `Instant` state to be followed by another `Instant` state) and replacing F7 with f7.

Analogous constraints on the previous state can also be obtained, by reversing each relation between derivatives, as shown in Table 3.2. [3] Using such constraints to perform backward qualitative simulation (i.e. postdiction) does not seem to have been explored in the QR literature.

[3]Note that the analog to rule f7 (i.e. b7) is generally not used in qualitative reasoning. It would disallow technically asymptotic changes, such as two water levels equalizing due to a flow between two containers, from occurring.

| | | |
|---|---|---|
| F1: Instant $\wedge$ $N_i > N_j$ | $\Rightarrow$ | Next($N_i > N_j$) |
| F2: Interval $\wedge$ $N_i \geq N_j$ | $\Rightarrow$ | Next($N_i \geq N_j$) |
| F3: Instant $\wedge$ $D(Q_i) > D(Q_j) \wedge Q_i \geq Q_j$ | $\Rightarrow$ | Next($Q_i > Q_j$) |
| F4: Interval $\wedge$ $D(Q_i) \geq D(Q_j) \wedge Q_i > Qj$ | $\Rightarrow$ | Next($Q_i > Q_j$) |
| F5: Instant | $\Rightarrow$ | Next(Interval) |
| F6: Interval | $\Rightarrow$ | Next(Instant) |
| F7: Instant $\wedge$ $Q_i = Q_j$ | $\Rightarrow$ | Next(Same-Rel($(Q_i, Q_j), (D(Q_i), D(Q_j))$)) |
| f7: Instant $\wedge$ $D(Q_i) = D(Q_j) \wedge Q_i = Q_j$ | $\Rightarrow$ | Next($Q_i = Q_j$) |

**Table 3.1**: Temporal constraints on the next state

$N_k$ signifies a number (either a quantity or its derivative) and $Q_k$ signifies a quantity.
Recall that proposition Instant denotes an instantaneous state and Interval is equivalent to the negation of Instant.

| | | |
|---|---|---|
| B1: Instant $\wedge$ $N_i > N_j$ | $\Rightarrow$ | Prev($N_i > N_j$) |
| B2: Interval $\wedge$ $N_i \geq N_j$ | $\Rightarrow$ | Prev($N_i \geq N_j$) |
| B3: Instant $\wedge$ $D(Q_j) > D(Q_i) \wedge Q_i \geq Q_j$ | $\Rightarrow$ | Prev($Q_i > Q_j$) |
| B4: Interval $\wedge$ $D(Q_j) \geq D(Q_i) \wedge Q_i > Qj$ | $\Rightarrow$ | Prev($Q_i > Q_j$) |
| B5: Instant | $\Rightarrow$ | Prev(Interval) |
| B6: Interval | $\Rightarrow$ | Prev(Instant) |
| B7: Instant $\wedge$ $Q_i = Q_j$ | $\Rightarrow$ | Prev(Same-Rel($(Q_i, Q_j), (D(Q_j), D(Q_i))$)) |
| b7: Instant $\wedge$ $D(Q_j) = D(Q_i) \wedge Q_i = Q_j$ | $\Rightarrow$ | Prev($Q_i = Q_j$) |

**Table 3.2**: Temporal constraints on the previous state

Such oversight may indeed be appropriate; as will be explained in the following two chapters, explicitly branching backward on all consistent pasts does not seem to be particularly useful for goal-directed qualitative reasoning.

Table 3.3 shows the temporal constraints on $S$ itself. These are necessary to ensure that the forward and backward temporal rules are applied when needed. For example, rule N1 detects when a change from equality is occurring, insuring that rule F1 applies.

### 3.1.2 Temporal Constraints Facilitate Partial State Reasoning

The above rules are represented in the most general form possible to support reasoning with partial states. This is facilitated by the use of soft inequality representations. For example,

| | | |
|---|---|---|
| N1: $D(Q_i) > D(Q_j) \wedge Q_i = Q_j$ | $\Rightarrow$ | Instant |
| N2: Interval $\wedge$ $D(Q_i) \neq D(Q_j) \wedge Q_i \geq Q_j$ | $\Rightarrow$ | $Q_i > Q_2$ |
| N3: Interval $\wedge$ $Q_i = Q_j$ | $\Rightarrow$ | $D(Q_i) = D(Q_j)$ |

**Table 3.3**: Temporal constraints on the current state

consider the pump-cycle system of Figure 2.4, with the pump off and the valve open. $\texttt{level}_1 \geq \texttt{bottom}_1$ is always true for this system. So, for any interval state in which $\texttt{level}_1$ is not steady, rule N2 implies that $\texttt{level}_1 > \texttt{bottom}_1$. Whereas, for any state which is instantaneous and $\texttt{level}_1$ is rising, rule F3 implies that $\texttt{level}_1 > \texttt{bottom}_1$ holds in the next state. Explicit case splits on whether $\texttt{level}_1 > \texttt{bottom}_1$ or $\texttt{level}_1 = \texttt{bottom}_1$ are not required to make these inferences.

To illustrate the potential computational benefits, consider a large liquid system of many containers and pipes, where the bottoms of all containers are the same height and all pipes are connected at the bottoms. Given an initial state in which the liquid level of one container ($X$) is specified to be above the bottom and all valves are specified as open, we can deduce that all of the levels will be above the bottom in the next state following this initial state. The temporal rules allow this deduction without requiring us to complete the initial state. Thus, we avoid the cross-product of the case splits on whether each level is above the bottom or not and which levels are higher than others, which would be required by standard qualitative simulation. We summarize below how this is done.

Without loss of generality, we always consider the initial state to be instantaneous by default. This assumption is sound so long as we are willing to consider a special transition from the initial state which involves only a change in duration (in case there is no causal reason why the state must last for only an instant). So, for the above example, we consider the initial state instantaneous. Rule F5 tells us that the next state is an interval. Rule F1 tells us that the liquid level of $X$ is still above the bottom in the next state. Rule N2 tells us that the liquid levels of containers connected to $X$ also must be above the bottom. Being at the bottom would make their derivatives positive (due to a flow from $X$), which contradicts the constraints of rule N2. Similar proofs by negation hold for containers connected to them, as so on, recursively. These inferences are propagated using our procedure for computing deductive closures , whose worst-case complexity is quadratic in the final size of the next state. In contrast, the worst-case

complexity of completing the initial state would be exponential in the size of $\mathcal{P}'$. This example typifies the many cases in which standard qualitative simulation techniques are truly overkill for a particular QR task.

### 3.1.3 Action-Augmented Envisionments

In [26], Forbus extends the continuity-based approach to consider changes due to actions, emphasizing the need to allow discontinuities. He called this the action-augmented envisionment approach (AAE). Although AAE was originally formulated in terms of total envisionments, here we generalize it in terms of the continuity-based algorithm given above.

Assume a STRIPS representation [22] for actions, where the add and delete lists of an operator instance $A_i$ are defined as propositional sets $A_i^a$ and $A_i^d$, respectively. Modelling the effects of such actions requires removing from $\mathcal{B}$ those background propositions which are in the add or delete list of any operator instance or can be indirectly changed by them. Call such propositions *manipulable*. Let $P_S$ be the set of manipulable propositions which are in a given state S.

Transitions from $S$ due to dynamics are computed as before, with a slight redefinition of $S_b$ to ensure that all propositions in $P_S$ are persisted:

$$S_b \equiv \{p \mid \texttt{Next}(p) \in S\} \cup \mathcal{B} \cup P_S.$$

For simplicity, AAE makes two assumptions. First, at most one action can occur at a time. This *single action assumption* is sufficient because compound action operators can model the effects of simultaneous actions. Second, actions are forbidden from occurring in instantaneous states (from which some dynamical change from equality is occurring). Actions are easiest to integrate with dynamics if viewed as (instantaneous) trigger events which eventually will result in their stated effects. Since it would be a coincidence for such a trigger event to occur at the same time that an instantaneous dynamic change is occurring, it is sound to ignore such possibilities. When an action is not appropriately represented as a trigger event with effects that are certain, it should instead be reformulated as an action which triggers some continuous dynamical changes which might eventually result in the intended effects.

Transitions from state $S$ which might occur due to an applicable action $A$ are computed as follows:

1. Let $S_b = \{p \mid \texttt{Next}(p) \in S\} \cup \mathcal{B}$ (as before).

2. Let $P'_S = (P_S - A^d) \cup A^a$.

3. Let $C$ = the set of consistent completions of $P'_S \cup S_b$.

4. If $C = \emptyset$ then Let $C$ = the set of consistent completions of $P'_S \cup \mathcal{B}$.

5. Consider each state in $\{S_i \in C \mid \nexists S_j \in C$ such that $(|S_i \cap S| > |S_j \cap S|)\}$ to be a next state of $S$.

Essentially, this algorithm insists that all of the direct effects of the action occur while only allowing minimal changes in the other propositions. As shown in Step 5, it employs a cardinality-based measure of minimality. Specifically, since QPT states can be sufficiently represented by their state assumptions, the measure of minimality is literally the number of assumptions which persist over the transition.

Forbus argues in [26] that it is important that this cardinality measure is based only on the state assumptions and not on all state propositions. Essentially, he correctly notes that since the non-assumption state propositions are consequences of the assumptions, intuitive notions of causality can be violated if the minimality measure includes such consequences. Recently, this point has also been argued by the minimal-change community [42, 3]. However, he fails to account for those rarer cases where it also matters that some state assumptions are themselves consequences of other assumptions.

Consider his example of a pan of water resting on two burners, one on and one off and an action of turning off the first burner. Intuitively, that action should result in no heat flows to the water. Minimizing changes in all state propositions would suggest that possibility but would also suggest that the heat flow persists while the second burner (magically) turns on. In the first case, the changes are: heat-flow1 is inactive, burner1 is off, and temperature of water is dropping (due to a cooling process that remains active). In the second case, the changes are: heat-flow1 is inactive, burner1 is off, heat-flow2 is active, and burner2 is on.

However, minimizing changes in state assumptions also suggests those same two possibilities. In the first case, the two assumption changes are: burner1 is off and the water temperature is dropping. In the second case, the two assumption changes are: burner1 is off and burner2 is on. The problem is that the assumption that the derivative in the temperature of the water is

negative is inferrable when both heat flow processes become inactive. Thus, there is really only one primitive assumption change in the first case, and so case one actually represents minimal change (versus case two). Yet, AAE would not prefer either over the other. [4] Furthermore, if the water temperature is originally rising (due to burner1 being on), AAE would actually prune case one (where the temperature immediately drops due to cooling with no burners on), since it violates continuity. Since case two does not violate continuity, that means AAE would actually prefer the magical case over the reasonable one.

## 3.2 Problems With Continuity-Based Approaches

As mentioned in the introduction to this chapter, continuity-based approaches have two key limitations: failing to predict all valid discontinuous changes and failing to avoid predicting coincidental changes. We discuss each of these issues in turn below.

### 3.2.1 Discontinuity

Although the real world may be fully continuous, it is doubtful that practical models of real world systems can be. Always modelling discontinuities as continuous qualitative changes requires details that may lead to excessively complex explanations and/or intractable reasoning. Thus, a strict continuity-based approach to change does not appear to be feasible or desirable.

Unfortunately, the extensions provided by AAE do not always correctly handle discontinuous change. AAE is incomplete for three reasons:

1. it can prune a valid change whose cardinality measure of minimality is inflated due to consequential state assumptions;
2. it can prefer a continuous change over a discontinuous change even when both are valid;
3. it does not account for discontinuous changes due to dynamics.

We explore these issues in detail below.

[4] If burner2 being on is a manipulable proposition (i.e. a member of $P_S$), then AAE would actually prefer case two. However, that is simply an example of getting the right result for the wrong reason. If the burner2 being on was instead determined by dynamics (i.e. not a member of $P_S$) then AAE would fail as we have described above.

#### 3.2.1.1 Four Motivating Examples

Discontinuities seem to be more common for changes due to actions than those due to dynamics. This is not surprising given that action operators can specify arbitrary effects, whereas dynamics result in continuous changes for at least as long as the process structure remains the same. In any case, the general problem of determining when to allow discontinuous changes, and what other changes should accompany them, can be subtle, as the following examples illustrate.

First, consider a one-dimensional world with at least four named points: $X_1 > X_2 > X_3 > X_4$. Assume the move operator for object $B$ when $B$ is between positions $X_1$ and $X_2$ has the effect of $B$ being between positions $X_3$ and $X_4$. Thus, the move action results in a discontinuity in the position quantity. Imagine there are some dynamics which would be triggered by $B$ being somewhere between $X_2$ and $X_3$, such as a photoelectric-cell detector sounding an alarm. Without explicitly reasoning about $B$ being between $X_2$ and $X_3$ "during" the discontinuity in position of the action, the alarm will not be predicted.

Now, consider an explosion due to dynamics, such as a spark plug causing a discontinuous change in gas pressure. Here it is more difficult to imagine any intermediate limit points of interest that the pressure could go through before reaching the pressure that results from the quick explosion process. So, unlike the previous example, it does not appear that one would miss any important intermediate behavior during the discontinuity. This dichotomy seems to be due to the fact that the arbitrary effects of actions can model forced discrete jumps through time that qualitative models generally do not and cannot, due to qualitative ambiguity. For example, movement of $B$ due to an explosion would directly result from higher pressures, not the explosion itself. So, the alarm would be predicted if $B$ moved due to an explosion.

In contrast, consider the action of turning on the pump in the pump-cycle system when $\texttt{level}_1 > \texttt{level}_2$ and the valve is open. We will call this the *pump-cycle sudden-pump* example. Before the action, $\texttt{level}_1$ is dropping due to liquid flow from CAN1 to CAN2. This action could conceivably result in three possibilities: $\texttt{level}_1$ could continue to drop (if the flow is stronger than the pump); $\texttt{level}_1$ could suddenly rise (if the pump is stronger); or $\texttt{level}_1$ could suddenly stop (if the pump is exactly as strong as the opposing flow). Note that AAE incorrectly prunes the second case because it involves more change than the first case, due to the discontinuity in

the derivative of $\texttt{level}_1$. Ignoring the second case would ignore the possibility that CAN1 might overflow due to this action — which could be disastrous, say, if the liquid was a dangerous acid.

Discontinuities in higher-order derivatives can also occur. For example, a ball rolling off a table would result in a discontinuous change in the derivative of the ball's vertical velocity as soon as the ball goes past the edge of the table. Essentially, the process providing an opposing force to gravity becomes inactive when the table no longer supports the ball. Thus, the vertical acceleration of the ball discontinuously becomes solely that of the gravity process at the moment the ball passes the edge of the table. AAE handles such a case correctly only because there is no continuous possibility to prefer over that discontinuous behavior.

#### 3.2.1.2 Claim: Most Discontinuities Occur in Derivatives

Most discontinuities that arise, whether due to dynamics or to actions, seem to occur in derivatives. We claim that this is because most discontinuous changes arise from sudden changes in process structure, which defines the influences which determine the derivatives. The latter two examples above are typical of such changes.

Exceptions are due to instantaneous movements through quantity space, such as the first two examples of the previous section. As noted, only changes due to actions seem to cause predictive problems for such cases. Reformulating such actions as triggers of continuous behavior would avoid such problems. For example, modelling the move action as an instantaneous push action followed by sliding due to dynamics might be the more precise definition of the actual move operator and would allow qualitative simulation to predict the alarm detector being tripped. However, if we are sure that such pushes will always be strong (and weak) enough to make $B$ rest between $X_3$ and $X_4$ when it was between $X_1$ and $X_2$, then the earlier formulation of the move operator provides some useful predictive constraints that would be lost. There appears to be no general way to resolve this tradeoff between encapsulating domain constraints in discrete action effects and using continuous processes to robustly detect intermediate possibilities.

#### 3.2.1.3 Reasoning About Discontinuities in Derivatives

Since discontinuities in derivatives seem so common, it is particularly important that we handle them correctly. As the pump-cycle sudden-pump example above illustrates, minimal change is not a sufficient criterion. Instead, we must explicitly reason about the influence structure that

defines the derivatives. The essence of our solution is based on a simple observation: adding a positive influence to an already increasing quantity will not cause that quantity to turn around (discontinuously), whereas adding a negative influence might (and vice versa).

Later, we will present the details of how this analysis of influence change is performed and present our full solution to the pump-cycle sudden-pump example. However, first we must discuss coincidences, since it turns out that the levels stopping as soon as the pump is turned on happens to be a coincidence.

### 3.2.2 Coincidences

We claim that there are no valid reasons for predicting coincidental changes, such as two independent pots of water both starting to boil at the same time. First, the probability of coincidences is effectively zero. Although at the qualitative level a coincidence may appear the same as any temporal ordering between the two events, at the quantitative level the number of ways that the coincidence might occur is infintestimal compared to the number of ways that it would not occur. Second, a coincidence generally has no causal significance over its non-simultaneous counterparts. It is difficult to even imagine of an example where two dynamic events occurring at the same exact time leads to dynamic behavior that could otherwise not also happen if one event occurred a little bit before the other. Furthermore, consider the classic example of a coin landing on its edge. Although we can conceive of such an occurrence, we would never seriously consider it in the course of trying to generate a winning strategy for a toss-a-coin-in-the-bottle carnival game.

While there appear to be no valid reasons to predict coincidences, there certainly are valid reasons to avoid predicting them. Without coincidences, the worst-case number of next states is linear in the size of $S$. Allowing coincidences makes the worst-case exponential in the size of $S$.

#### 3.2.2.1 Non-Coincidental Changes

The problem, of course, is that not all non-singleton changes are coincidences. From a phase-space point of view, we want to predict only those changes which (must) occur from at least one exact state consistent with $S$. This criterion covers two types of changes at the qualitative level:

1. inferred necessary changes;

2. justified possible changes.

Examples of the first case arise from QPT correspondences, such as a container's level of liquid necessarily being above its bottom when the mass of some liquid in it becomes positive. An example of the second case is the derivative of $\texttt{level}_1$ possibly reversing when the pump is turned on in the pump-cycle sudden-pump example.

The essence of our handling of coincidences is to use minimal change to account for the first type of non-coincidental changes while using the results of our influence change analysis to avoid including possible discontinuities in derivatives in our measures of minimality. We discuss this approach in detail in Section 3.3.1.

#### 3.2.2.2 Conceiving of Coincidences Versus Predicting Them

Although there appear to be no reasons to predict coincidences, it may be desirable to verify a user query as to whether such changes are conceivable. For example, the user may wish to determine whether two changes from a state could occur arbitrarily close to one another, by seeing whether they could theoretically happen at the same time. The fact that people seem to be fascinated by the possibility of coincidences suggests that reasoning about such limits is potentially valuable.

Such queries are easily answered within our framework by simply including all changes of interest in the base next state $S_b$. Essentially this focuses the search on whether all such changes can occur together from $S$.

Although such queries may be of local interest, our main point is that coincidences are not of global interest. As discussed earlier, it is difficult to imagine how coincidences could have causal significance. Thus, whether they occur during intermediate behavior does not affect which final result states are possible from a given state.

#### 3.2.2.3 Using Coincidences to Avoid Ordering Committments

One might argue that a valid reason to allow coincidental changes is to avoid the need to branch on which independent change occurs first. For example, consider two independent heated vats of oil whose temperatures are both approaching their boiling points. It might make sense to

consider only that the oil in both vats starts to boiling at the same time, even of each one boiling first is actually possible. Although this could significantly reduce branching factors, there are two important points to realize.

First, there is a difference between *forcing* coincidental changes and *allowing* them. Allowing coincidental changes along with all other orderings simply makes the branching factor worse. Thus, we still insist that it never makes sense to mix both coincidental and non-coincidental changes for the same quantities.

Second, if two subsystems are not globally independent, forcing coincidental changes for them can introduce incompleteness — beyond the local incompleteness due to pruning orderings. Determining such global independence is not trivial, which is why we have not pursued the use of coincidences to reduce branching. For instance, one change occurring before another might be the only way that a third change could occur later. At the very least, one could identify subsystems which are always independent of each other, regardless of context. But for such extreme cases, it would be even better to simply reason about them separately in general, not just for computing transitions.

#### 3.2.2.4 Implicit Closed-World Assumption On Correspondences

Perhaps the strongest argument against avoiding coincidences is the following. A key motivation for using qualitative models is to relax the need for precise constraints, which may be unavailable or expensive to use. Now, if a correspondence constraint exists for the physical system but is missing from the qualitative model, then we would fail to predict any changes which appear to be coincidences but are actually required by that missing correspondence. Although our predictions would be complete with respect to that model, they would be incomplete with respect to the real world. Thus, our predictions will suffer incompleteness due to relaxing constraints, opposing the guiding principle behind QR.

Implicitly, we are making a closed-world assumption that the model contains all of the relevant correspondences. QR accepts the need for other CWA's, such as those on influences, why should correspondences be treated differently? In fact, if we do not make all such CWA's, then the use of minimal change to filter discontinuous possibilities is incompatible with QR's attempt to preserve predictive completeness. Lack of CWA's prevents inconsistencies (due to negations by failure) from being detected, which can result in minimal change favoring

outcomes which are in reality inconsistent. In fact, lack of CWA's is exactly what prevents nonmonotonic approaches based on minimal change from achieving the predictive completeness of envisionments.

We can also appeal to likelihood. We argue that we can safely assume that it is unlikely that correspondences will exist and yet the model will not contain them. Relatively few correspondences are causally fundamental to the general physics (like one implying that fluid mass is positive when volume is positive); such correspondences will be included in any reasonable model. Other possible correspondences simply reflect global quantitative constraints of specific systems. They are mathematically very unlikely to hold for any specific system for which we apply the general qualitative model.

## 3.3 Combining Minimality and Continuity-Based Change

In this section, we explore the use of minimal change to correct the above problems with the continuity-based approaches. In moving away from a continuity-based approach and toward a minimality-based approach, the challenge is to not discard valid continuity constraints. We will argue that this requires a more careful consideration of the justification structure of constraints than is typical of other work based on minimal change.

### 3.3.1 Correcting the Continuity-Based Algorithms

Now we are prepared to use minimal change to find all possible transitions out of a given state $S$ which do not involve coincidental changes but which do reflect all valid (possibly discontinuous) changes.

#### 3.3.1.1 The Single Base Change Assumption

We claim that it is both useful and sufficient to generate non-coincidental dynamic transitions by focusing on one proposition as the base change. This is analogous to AAE's focus on one action as the base change. However, whereas an action may have several base effects, a non-coincidental dynamical change has only one base effect — a change in some quantity or derivative. This can be clearly seen by viewing dynamic changes as due to operators of nature, such as those of Figures 4.3 and 4.3. For generality, we will refer to each action or base dynamic

change as an *operator* and its set of direct effects as its *effects*. The effects of an action consists of its add list propositions and the negations of its delete list propositions.

#### 3.3.1.2 Approach 1: Almost Sufficient for QPT Models

One could attempt to compute the next states of $S$ due to operator $O$ as follows:

1. Let $S_b = \{p \mid \texttt{Next}(p) \in S\} \cup \mathcal{B}$.
2. Let $E = \mathit{Effects}(O)$.
3. Let $C_1 = \textbf{MinimalChange}(\textbf{NonDerivs}(S), E + \textbf{NonDerivs}(S_b))$.
4. For each $S'$ in $C_1$ do:
   (a) Let $D = \textbf{PossiblyDiscontinuous}(S, S')$.
   (b) Let $D' = \{d_i | \overline{d_i} \in D\}$.
   (c) Let $C_2 = \textbf{MinimalChange}(S - D, S' + S_b - D')$.
   (d) For each $S'$ in $C_2$ do: $\mathit{Nexts}(S) \leftarrow (\mathit{Nexts}(S) \cup \textbf{Completions}(S', S))$.

All of these state operations work with state assumption closures.

**NonDerivs**($S$) returns the state assumption closure minus any derivative conditions. Recall that these conditions not only include derivative inequalities, but also quantity inequalities which are equivalent to derivative inequalities (such as net-influences) or higher-order derivative inequalities (such as acceleration quantity inequalities being equivalent to velocity derivative inequalities). **Completions**($S_i$,$S_j$) computes all completions of $S_i$, using only the set of $\mathcal{P}'$ propositions in $S_j$ (as their negations). When $S_i$ is a complete state (in terms of $\mathcal{P}'$), $S_j$ will necessarily be so also.

As explained later, **MinimalChange**($S_i$,$S_j$) computes states representing minimal changes from $S_i$ consistent with base next state $S_j$ and **PossiblyDiscontinuous**($S_i$,$S_j$) determines which (derivative) conditions in $S_i$ could change discontinuously due to influence structure changes which occur for the transition from $S_i$ to $S_j$, and

Essentially, this algorithm first finds the set $C_1$ of candidate partial next states which are minimally changed from $S$ and which do not contain (non-inferrable) derivative conditions. For each such partial state $S'$, it determines which derivative conditions ($D$) in $S$ might change

discontinuously because of influence changes due to the minimal changes from $S$ to $S'$. It then recomputes minimal changes from $S$, this time enforcing continuity on all derivative conditions except for those of $D$. Finally, it computes completions to represent the ambiguities in whether each potentially discontinuous derivative relation $D$ changes or not.

#### 3.3.1.3 Approach 2: Sufficient for QPT Models

Although the above approach correctly handles discontinuities in derivatives, it does not allow for discontinuities in quantity conditions or rate relations. Such discontinuities seem to be much rarer, but examples do exist (see Section 3.2.1.1), mainly for actions which jump through quantity space. The following modified version fixes this problem, by not explicitly enforcing continuity except in some very special cases involving `Instant` states. This approach is based on the observation that minimal change can be sufficient for enforcing continuity constraints, as discussed in the next section.

The next states of $S$ due to operator $O$ are given by:

1. Let $S_b = \{p \mid \texttt{Next}(p) \in S\} \cup \mathcal{B}$, where temporal rules F1, F2 and F4 are *not* used to define the `Next` propositions of Table 3.1
2. Let $E = \mathit{Effects}(O)$.
3. Let $C_1 = \textbf{MinimalChange}(S, E + S_b)$.
4. For each $S'$ in $C_1$ do:
   (a) Let $D = \textbf{PossiblyDiscontinuous}(S, S')$.
   (b) Let $C_2 = \textbf{MinimalChange}(S - D, E + S_b)$.
   (c) For each $S'$ in $C_2$ do: $\mathit{Nexts}(S) \leftarrow (\mathit{Nexts}(S) \cup \textbf{Completions}(S', S))$.

#### 3.3.1.4 Continuity Rules Are Still Important

An important consequence of removing most continuity constraints from $S_b$ is that determining whether $O$ is applicable to $S$ requires special care when $O$ is a dynamic operator. Not only must we check that $O$'s conditions hold in $S$, we must also check that they are continuous through the transitions generated above. This is done by applying all forward temporal rules

to $O$'s conditions (augmenting Step 1 of Approach 2 above), to ensure that the dynamic operator can hold throughout those transitions. Whereas an action can arbitrary impose sudden discontinuous change, Nature is more gentle. Specifically, any discontinuities due to dynamic changes will arise from indirect effects of $O$, reflecting our modelling simplifications that violate the underlying continuous nature of Nature's real model.

This application of $O$-specific continuity constraints on top of minimal change is very important. For example, without it, minimal change could suggest that a container `CAN1` might overflow due soley to gravity flow from a source container `CAN2` when `CAN2`'s water level was initially above `CAN1`'s water level but below `CAN1`'s top. However, that overflow could not occur, since the flow would first stop once the levels in the two containers became equal. Minimal change alone would "fix" that obstacle by insisting that the flow's conditions discontinuously change so that the water level of `CAN2` is no longer below the top of `CAN1`. Effectively, minimal change would insist that a change in the levels relation would happen at the same instant that `CAN1`'s water level rises to its top. Such behavior violates our base change assumption, as well as the basic tenants of causality.

By ensuring that continuity holds for the (local) behavior of $O$'s immediate conditions and effects, we ensure that minimal change will not violate the local continuous behavior of Nature. Any other discontinuous changes are fair game, reflecting discontinuities induced by simplifications in how we model the propagation effects of Nature's base change.

#### 3.3.1.5 Encoding Continuity as Minimal Change

At first glance, it might appear that syntactic measures of minimal change cannot enforce continuity when appropriate. Consider the following three sets of propositions, representing $A < B$, $A = B$, and $A > B$ respectively:

$$\{A < B, A \neq B, A \not> B\},$$

$$\{A \not< B, A = B, A \not> B\},$$

$$\{A \not< B, A \neq B, A > B\}.$$

Using those encodings, a change from $A < B$ to $A = B$ involves as many proposition flips (two) as a change to $A > B$. Yet, the former change should be preferred because it is continuous.

However, it turns out that our soft inequality representations enforce continuity when using minimal change measures. Using these encodings, a change from $A < B$ to $A = B$ involves only one propositional flip, whereas a change to $A > B$ involves two:

$$A < B \Rightarrow \{A \not\geq B, B \geq A\},$$

$$A = B \Rightarrow \{A \geq B, B \geq A\},$$

$$A > B \Rightarrow \{A \geq B, B \not\geq A\}.$$

The use of special syntactic encodings to get minimal change to do the "right thing" is not a general solution. Usually, syntactic measures are mainly employed as a convenience, to avoid the complexity of fully addressing deeper issues. However, our use of soft inequality encodings is not simply an encoding trick because it is symmetric. There is no unfair bias for any particular change among inequalities. In a sense, minimal change of soft inequalities simply reflects the basic qualitative nature of the Mean-Value Theorem.

Note that AAE's use of minimal change when there are no continuous nexts usually works because in such cases other inequalities are not required to change, so minimal change does not have to decide between $=$ or $>$ from $\prec$ for them. Thus, the inequalities that must change have already been told how to change by the base effects of the operator.

Temporal rules F1 and F4 follow directly from minimal change because changes in $>$ relations are minimized. Only rule F2 depends on this special encoding based on soft inequalities. Note in particular, however, that rule F3 is still required, to insure that all instantaneous changes from equality occur.

It is important to also note that either encoding scheme will prefer shorter discontinuous changes. Minimal change measures for quantity spaces with multiple, ordered limit points will lead to this preference due to the nature of transitive closures. For example, consider a quantity space for $X$ with known limit points $X_1 < X_2 < X_3$. Changing from $X < X_1$ to $X = X_3$ involves more proposition flips (5) than changing to $X = X_2$ (3):

$$X < X_1 \Rightarrow \{X \not\geq X_1, X_1 \geq X, X \not\geq X_2, X_2 \geq X, X \not\geq X_3, X_3 \geq X\},$$

$$X = X_2 \Rightarrow \{X \geq X_1, X_1 \not\geq X, X \geq X_2, X_2 \geq X, X \not\geq X_3, X_3 \geq X\},$$

$$X = X_3 \Rightarrow \{X \geq X_1, X_1 \not\geq X, X \geq X_2, X_2 \not\geq X, X \geq X_3, X_3 \geq X\},$$

Since we are assuming that discontinuities arise due to quick changes, this preference on shorter discontinuous changes has some merit. After all, $X$ can only travel so far in the time in which the primary change occurs. However, a more accurate way to represent this intuition would be to limit the change in $X$ to within some range of limit points, based on the duration of the primary change and the speed at which $X$ travels based on that change. Unfortunately, such representation would require much knowledge specific to the primary change and its relation to $X$. It is particularly doubtful that such knowledge could be represented qualitatively. As it is, preference on shorter discontinuous changes introduces a certain amount of incompleteness in qualitative prediction. If such incompleteness is significant, minimality measures must be tailored to avoid this.

#### 3.3.1.6 Influence Change Analysis

**PossiblyDiscontinuous**($S_i$,$S_j$) determines which derivative conditions in state $S_i$ could change discontinuously due to influence structure changes which occur for the transition from $S_i$ to $S_j$.

Influence change analysis reflects a simple but important principle: adding a positive influence to an already increasing quantity will not cause that quantity to turn around (discontinuously), whereas adding a negative influence might (and vice versa).

More precisely, for a qualitative change in a derivative to occur, the enlarged conjunctive set of negative influences must not be `Overwhelmed` by the existing conjunctive set of positive influences. Thus, known rate relations must be considered as well.

Our approach is roughly as follows. Consider each proposition $P$ in $S_i$ which is not in **NonDerivs(S)** [5] Assume the form of $P$ is $D(1)$ *vs* $D(2)$. Consider the influences on $Q_1$ and $Q_2$ in turn as quantity $Q$. Gather all minimal support states in which $Q$ could be influenced in the direction opposing $P$. This set of states (call it $M$) is effectively either the ATMS label of (`PossibleD+` $Q$) or (`PossibleD-` $Q$). For each state $M_i$ in $M$, if $M_i$ is compatible with $S_j$ but not $S_i$ then $P$ and its equivalents are among those propositions returned as "possibly discontinuous".

[5] In the case of $P$ being a quantity inequality like a net-influence, let $P$ be its equivalent derivative inequality.

#### 3.3.1.7 Computing Minimal Change

**MinimalChange**($S_i$,$S_j$) computes the minimal changes of state $S_i$'s propositions consistent with base new state $S_j$.

The literature on minimal change is large, with many measures of minimality and many algorithms for each. We will discuss some of these alternatives in Section 3.4, to better understand the approach we have taken. For now, we simply state our approach below.

The set of propositions we minimally change is the state assumption closure of $S_i$. For our purposes, it has proven adequate to compute minimal changes by performing a limited form of depth-first interpretation construction on $S_j$ that is biased to prefer the assumptions true in $S_i$. Consider any point in the construction process, for working state $S \supset S_j$ and a choice of assumption $A$ or $\overline{A}$, where $A$ is true in $S_i$. We explore $S + \{\overline{A}\}$ only if there are no consistent interpretations of $S + \{A\}$. For each $S$, we must consider each assumption in $S_i$ in turn as $A$. It is not sufficient to simply select as $A$ for $S$ the first in some strict ordering of assumptions which is neither true nor false in $S$ (as is sufficient for standard, unbiased, depth-first interpretation construction). This approach reflects a set-inclusion measure of minimality.

We have found that representing each $S$ as its assumption closure is sufficient to avoid backtracking in almost all cases. In fact, we have found it equally sufficient to simply first commit to choices whose negations are inconsistent with $S$, without computing closures of each $S$ during construction. These results, which violently conflict with worst-case analysis, seem to be due to the predominance of binary constraints in QP. For example, for a transition in which mass of a sole contained liquid becomes zero, the level of the contained liquid will become that of the bottom of the container, fundamentally due to a same-relation of mass vs zero and level vs bottom. Once interpretation construction commits to the choice of the mass being zero, the liquid level's relation to bottom is inferrable in the assumption closure.

For general constraint sets, a large $S_i$, and a small $S_j$, it would intuitively seem more efficient to compute minimal change using a top-down approach, which would try to minimally "debug" $S_i$ to make it compatible with $S_j$. In contrast, we perform bottom-up reasoning from $S_j$, adding back each assumption of $S_i$ unless doing is ultimately detected as inconsistent. In the final analysis, our experience that the bottom-up approach is adequate is a direct result of our experience that BCP is typically sufficient to compute assumption closures for QPT models.

### 3.3.2 Handling the Pump-Cycle Sudden-Pump Example

Now we are prepared to discuss our solution to the pump-cycle sudden-pump example in detail. Given an initial state in which water is flowing through the pipe from CAN1 to CAN2, we wish to determine the possible immediate consequences of turning on the pump. Thus, the problem is to compute the next states of $\mathcal{I}$ due to the action change ($\mathcal{A}$):

$$\texttt{Not(On(PUMP))} \rightarrow \texttt{On(PUMP)}.$$

Minimal change results in $\texttt{level}_1 > \texttt{level}_2$, Open(VALVE), and On(PUMP) necessarily holding in any next state of $\mathcal{I}$ due to $\mathcal{A}$. However, influence change analysis indicates that minimal change should not be applied to $D(\texttt{level}_1) < 0$, because $\mathcal{A}$ adds a positive influence on $\texttt{level}_1$ that could qualitatively change influence resolution for $\texttt{level}_1$. Thus, there are three candidate next states:

- S1: $D(\texttt{level}_1) < 0$ (i.e. no qualitative change),
- S2: $D(\texttt{level}_1) = 0$, and
- S3: $D(\texttt{level}_1) > 0$.

We will now demonstrate that S2 is not a valid next state, due to coincidence pruning, whereas both S1 and S3 are valid next states.

Table 3.4 summarizes the state descriptions for this example. I1, I2, I3 represent the refinements of I that necessarily hold at the end of I in order for S1, S2, S3 respectively to occur next. Most of the propositions in the closures of these states are inferrable from others. In particular:

- $D(\texttt{level}_1)$ *vs* $D(0)$ is inferrable from $\texttt{level}_1$ *vs* $\texttt{level}_2$ in I, I1, I2, and I3 (since there is only one influence on $\texttt{level}_1$ when Open(VALVE) and Not(On(PUMP))),
- $D(\texttt{level}_2)$ *vs* $D(0)$ is similarly inferrable from $\texttt{level}_1$ *vs* $\texttt{level}_2$,
- $D(\texttt{virtual}(\texttt{rate}_{\texttt{pipe}}))$ *vs* $D(0)$ is inferrable from $D(\texttt{level}_1)$ *vs* $D(0)$, as is $D(\texttt{virtual}(\texttt{rate}_{\texttt{pipe}}))$ *vs* $D(\texttt{virtual}(\texttt{rate}_{\texttt{pump}}))$, since $\texttt{virtual}(\texttt{rate}_{\texttt{pump}})$ is constant,

- $\texttt{virtual}(\texttt{rate}_{\texttt{pipe}})$ *vs* $\texttt{virtual}(\texttt{rate}_{\texttt{pump}})$ is inferrable in S1, S2, and S3 from $D(\texttt{level}_1)$ *vs* $D(0)$, since the virtual and real rates are equivalent for each flow in those states.

The inferred relation of $D(\texttt{virtual}(\texttt{rate}_{\texttt{pipe}}))$ *vs* $D(\texttt{virtual}(\texttt{rate}_{\texttt{pump}}))$ in I1/I2/I3 and the inferred relation of $\texttt{virtual}(\texttt{rate}_{\texttt{pipe}})$ *vs* $\texttt{virtual}(\texttt{rate}_{\texttt{pump}})$ in S1/S2/S3 together suggest the relation of $\texttt{virtual}(\texttt{rate}_{\texttt{pipe}})$ *vs* $\texttt{virtual}(\texttt{rate}_{\texttt{pump}})$ for I1/I2/I3. For example, $\texttt{virtual}(\texttt{rate}_{\texttt{pipe}}) > \texttt{virtual}(\texttt{rate}_{\texttt{pump}})$ holding in I1 is the only way that it can also hold in S1, given that $\texttt{virtual}(\texttt{rate}_{\texttt{pipe}})$ is dropping faster than $\texttt{virtual}(\texttt{rate}_{\texttt{pump}})$ in I1. We know that changes in $\texttt{virtual}(\texttt{rate}_{\texttt{pipe}})$ *vs* $\texttt{virtual}(\texttt{rate}_{\texttt{pump}})$ due to $\mathcal{A}$ must be continuous because $\mathcal{A}$ does not immediately effect the QC's that constrain those two quantities. This is an example of how explicit reasoning about continuous change is important in QR, beyond computing constraints on the next states.

Note that continuity between I3 and S3 only warrants inferring $\texttt{virtual}(\texttt{rate}_{\texttt{pipe}}) \leq \texttt{virtual}(\texttt{rate}_{\texttt{pump}})$ in I3. However, we also know that `Interval` holds in I1,I2,and I3 because $\mathcal{A}$ is an action, which we insist cannot occur in an instant state. Thus, rule N2 of Table 3.3 leads to the stronger constraint on that quantity inequality for I3.

Most importantly, the continuity-based inference we make for I2 indicates the change

$$(\texttt{virtual}(\texttt{rate}_{\texttt{pipe}}) > \texttt{virtual}(\texttt{rate}_{\texttt{pump}})) \rightarrow (\texttt{virtual}(\texttt{rate}_{\texttt{pipe}}) = \texttt{virtual}(\texttt{rate}_{\texttt{pump}}))$$

would occur for the transition between I2 and S2. However, this change is not required by $\mathcal{A}$. In particular, I1 is identical to I2 but transitions to S1, which does not entail that change. Therefore, we justify pruning S2 as a valid next state for $\mathcal{I}$ because it involves a coincidental change.

There are a few interesting things to note about this example. First, $\mathcal{I}$ does not commit to $D(\texttt{virtual}(\texttt{rate}_{\texttt{pipe}}))$ *vs* $D(\texttt{virtual}(\texttt{rate}_{\texttt{pump}}))$ and it is not inferrable in $\mathcal{I}$. This is one example of how relevant case splits in $\mathcal{I}$ can be introduced as needed, as opposed to relying on blind completions, as in QS.

Second, if I is case-split by $\texttt{virtual}(\texttt{rate}_{\texttt{pipe}})$ *vs* $\texttt{virtual}(\texttt{rate}_{\texttt{pump}})$, set-inclusion based minimal change would be sufficient to realize that S1 and S3 are valid next states but S2 is not. However, in more complex examples, case-splitting on all consistent combinations of conjunctive rate relations would be required for standard minimal change to work. In contrast, by using

| | | | I | I1 | I2 | I3 | S1 | S2 | S3 |
|---|---|---|---|---|---|---|---|---|---|
| $\texttt{level}_1$ | *vs* | $\texttt{level}_2$ | > | > | > | > | > | > | > |
| Open(VALVE) | | | $T$ | $T$ | $T$ | $T$ | $T$ | $T$ | $T$ |
| On(PUMP) | | | $F$ | $F$ | $F$ | $F$ | $T$ | $T$ | $T$ |
| $D(\texttt{level}_1)$ | *vs* | 0 | (<) | (<) | (<) | (<) | < | = | > |
| $D(\texttt{level}_2)$ | *vs* | 0 | (>) | (>) | (>) | (<) | (>) | (=) | (<) |
| $D(\texttt{virtual}(\texttt{rate}_{\texttt{pipe}}))$ | *vs* | 0 | (<) | (<) | (<) | (<) | (<) | (=) | (>) |
| $D(\texttt{virtual}(\texttt{rate}_{\texttt{pump}}))$ | *vs* | 0 | [=] | [=] | [=] | [=] | [=] | [=] | [=] |
| $\texttt{D}(\texttt{virtual}(\texttt{rate}_{\texttt{pipe}}))$ | vs | $\texttt{D}(\texttt{virtual}(\texttt{rate}_{\texttt{pump}}))$ | (<) | (<) | (<) | (<) | (<) | (=) | (>) |
| $\texttt{virtual}(\texttt{rate}_{\texttt{pipe}})$ | vs | $\texttt{virtual}(\texttt{rate}_{\texttt{pump}})$ | | > | > | < | (>) | (=) | (<) |
| Interval | | | | T | T | T | | | |

**Table 3.4**: Summary of states for pump-cycle sudden-pump example

influence change analysis to avoid enforcing minimal change on possibly affected derivatives, we avoid the need to explicitly consider all such combinations. Only rate relations inferred on I by the alternative next states, via temporal constraints, need be considered. The main reason even that is required is simply to prune states like S2 which are due to coincidental changes in rate relations.

Third, it is important to realize that the coincidence of the transition from $\mathcal{I}$ to S2 is not that the two virtual rates became equal per se, but that action $\mathcal{A}$ would happen to occur at the exact instant that those rates became equal.

Fourth, dynamic changes coinciding with action changes seems particularly worthy of ignoring, since action occurrences are presumably never caused by dynamics. In contrast, two apparently coinciding dynamic changes might at least conceivably be related in ways that the particular model does not reflect.

## 3.4 Comparison to Minimality-Based Approaches

In this section, we survey some key work on minimal change and relate it to our use of minimality. Our ultimate goal is to offer a better understanding of the relationship between continuity and minimality that our work brings out.

Work on minimal change can be broadly characterized as addressing the problems of belief *revision* and belief *update* [36]. That distinction depends on whether the correct set of beliefs

is static (e.g. a historical data base) or dynamic (e.g. an evolving physical system). Naturally, our interest is in the later, so we will focus on theories of minimality-based update.

### 3.4.1 Possible Worlds Versus Possible Models

The use of complete states in QS corresponds most directly to the use of *possible models* [56] in update systems such as IMMORTAL [3], By reasoning with complete state descriptions, these approaches avoid the possibility of not tracking over time changes in specific propositions which turn out to be critical later in pruning certain invalid state transitions. However, as a consequence of such conservatism, they are likely to require reasoning over many more alternative state descriptions than necessary.

Ginsberg and Smith present an approach in [31], called the *possible worlds approach* (PWA) that does not work directly with logical models, as Winslett's *possible models approach* (PMA) does, but rather with logical formulas. Thus, PWA minimizes changes in logical formulas (i.e. incomplete state descriptions), whereas PMA minimizes changes in logical models (i.e. complete state descriptions). Unfortunately, as Winslett shows in [56], PWA can lead to inferences which are unsound with respect to PMA. For example, consider a state description formula

$$F1: \ A \wedge B \wedge C$$

and an update formula

$$F2: \ \overline{A}$$

for a simple theory consisting of just two formulas:

$$T1: \ A \Rightarrow C,$$

$$T2: \ A \Rightarrow D.$$

Update via PWA would result in

$$F3: \ \overline{A} \wedge B \wedge C,$$

whereas update via PMA would result in

$$F4: \ \overline{A} \wedge B \wedge C \wedge D.$$

Minimal change via PMA results in D necessarily persisting because D is in every model of F1.

Performing PWA update by F2 on the closure of formulas F1, T1, and T3 would give the correct result for that example, but would introduce new problems for other examples. However, specific new problems mentioned by Winslett are due to PWA's employment of the "when in doubt throw it out" approach, which gives a final result which is the intersection of every minimal change extension. Our approach, which describes states by their assumption closure and explores all minimal extensions (i.e. does not intersect them), seems as correct as PMA for Winslett's examples. In order to see how our partial state approach might fail, we now explore the issues that del Val raises in his attempt to directly manipulate formulas while achieving the same semantics as PMA.

### 3.4.2 PMA Update Using DNF States

In [18] and [17], del Val provides syntactic characterizations of the PMA update operator. His results are general, handling formulas in disjunctive normal form (DNF), conjunctive normal form (CNF), and negation normal form (NNF). In order to better understand our approach, we reformulate some of his results in terms of partial state definitions below.

Let the update problem be to minimally change the propositions of state $I$ to be consistent with state $G$ according to PMA semantics. Recall that we view each state as a conjunctive set of propositions. To directly interpret del Val's work, we assume for now that states are described using all propositions in $\mathcal{P}$, not just the assumptions defined by $\mathcal{P}'$. Let DNF(S) for state $S$ denote a DNF formula representation of the union of a CNF formula corresponding to the scenario model $\mathcal{M}$ and a CNF formula corresponding to $S$'s defining set of propositions. Let DNF-STATES(S) be the set of states representing DNF(S), one state per disjunct of DNF(S). In our terminology, del Val's work indicates how one can update DNF-STATES($I$) using DNF-STATES($G$), resulting in a set of states whose completions correspond exactly to the models given by PMA.

We can view del Val's update operation as updating each state $S_i$ in DNF-STATES($I$) independently for each $S_g$ in DNF-STATES($G$) and then unioning the results into a final set of DNF-STATES. For each $S_i$,$S_g$ pair, let the base of the update be $S = S_i \cup S_g - Diff(S_i, S_g)$, where $Diff(a, b) \equiv \{p \in a | \not p \in b\}$ defines the difference between two conjunctive sets. Then, "patch" S by adding to it all consistent combinations of one proposition being negated from each $S_k$ in DNF-STATES($S_g$) for which $Diff(S_g, S_i) \not\subseteq S_k$. Actually, only the propositions of

$S_k - (S_i \cup S_g)$ are candidates for these negations, since doing otherwise would contradict the unpatched $S$. The purpose of these negations is simply to ensure that no other resulting DNF-STATES will have completions which are set-inclusion closer to $S_i$ than the patched versions of $S$.

For a simple example, consider the following version of our previous example:

$$I: \ A \wedge B \wedge C,$$

$$G: \ \overline{A}$$

for the scenario model $\mathcal{M}$

$$T1: \ A \Rightarrow C,$$

$$T2: \ A \Rightarrow D,$$

$$T3: \ \overline{A} \Rightarrow \overline{C},$$

$$T4: \ X \vee Y \vee Z.$$

DNF-STATES(I) yields $\{I1, I2, I3\}$, defined as follows:

$$I1 = \{A, B, C, D, X\},$$

$$I2 = \{A, B, C, D, Y\},$$

$$I3 = \{A, B, C, D, Z\}.$$

Similarly, DNF-STATES(G) yields $\{G1, G2, G3\}$, defined as follows:

$$G1 = \{\overline{A}, \overline{C}, X\},$$

$$G2 = \{\overline{A}, \overline{C}, Y\},$$

$$G3 = \{\overline{A}, \overline{C}, Z\}.$$

Updating I by G via our state-based interpretation of del Val's approach yields a DNF-STATES set of three states:

$$S1 = \{\overline{A}, B, \overline{C}, D, X\},$$

$$S2 = \{\overline{A}, B, \overline{C}, D, Y\},$$

$$S3 = \{\overline{A}, B, \overline{C}, D, Z\}.$$

A key thing to note about this example is that each state $S$ in any DNF-STATES set has the property that every conceivable completion of $S$ is consistent except for those adding the negation of some proposition in $S$. For example, state S1 above has four completions, reflecting all combinations of the statuses of Y and Z. Essentially, this approach satisfies each disjunctive clause of $\mathcal{M}$ for $I$ and for $G$ in all consistent ways and operates with each resulting partial state (which also corresponds to a partial logical model). Note also that this approach operates, at the very least, with the closure of $S$, since every state in DNF-STATE(S) will contain every proposition in the closure of $S$. In general, the level of state detail articulated by this approach is between that of ours and that of QS. This approach indicates one way in which a QR approach adhering to PMA update semantics could avoid reasoning with complete states.

In terms of addressing the problem of irrelevant state detail in QR, the utility of such an approach is unclear. One the one hand, it provides a formal basis of defining state descriptions which are sufficient for guarantying PMA update semantics. Thus, it defines a maximal level of detail that might be required. At first glance, defining such a level might seem to be an insightful accomplishment, Unfortunately, it appears that for QR tasks that the states in DNF-STATES(I) would generally not be much smaller than complete QS states. This hypothesis is not based on the fact that DNF-STATES(I) distinguishes on all propositions in $\mathcal{P}$ whereas QS states distinguish only on those of $\mathcal{P}'$. That difference seldom matters, since states which completely commit to statuses of all $\mathcal{P}'$ propositions will also commit to the statuses of all propositions in $\mathcal{P}$, due to strong closed world assumptions. The real problem is that significant representational savings relative to QS might occur only if there are many large disjunctions in $\mathcal{M}$. For example, the three states S1,S2,S3 above represent eight complete states, due to the size-three disjunct $X \vee Y \vee Z$. Unfortunately, large disjunctions are relatively rare in QPT models; mainly arising due to closed-world assumptions on large influence sets. Furthermore, it is our experience that general QPT domain models usually relate almost every proposition to every other one, under some contexts. Thus, large, isolated disjunctions, analogous to $X \vee Y \vee Z$ in the above example, would seem to be very rare indeed.

Perhaps an even more fundamental problem with this approach is that propositions corresponding to independent physical subsystems would nevertheless be intermingled in DNF-STATES(I) and DNF-STATES(G). For instance, if we extend our example to include "$T4 : V \vee W$" in $\mathcal{M}$ then this approach would consider states containing both X and V, even though

their constraints do not interact. Although this problem arises in QS as well, our goal-directed approach does avoid this problem, at least when the goal $\mathcal{G}$ does not contain propositions from multiple subsystems.

### 3.4.3 Justification-Based Measures

Minimal change, even on state assumption closures, is generally insufficient to reflect the underlying QR constraints. Changes which look non-minimal at a partial state level can actually be minimal at the complete state level. For instance, this phenomena can arise when rate relations are unspecified, as in the initial state for our pump-cycle sudden-pump example. The main reason that we perform influence change analysis is to avoid the need for explicitly distinguishing states by all possible conjunctive rate relations. That allows us to handle special cases like the pump-cycle example without requiring complete states. In this section we explore the more general issue of whether other special types of cases can arise, at least for QPT models, that require explicit reasoning about the underlying justification structure of state propositions.

The need to reason about underlying justification structure to determine how to properly persist derived information was explored by Myers and Smith in [42]. They presented three simple examples. Example 1 notes that a state containing the proposition A necessarily implies clause $A \vee B$ as well. Yet, a change to that state which causes $\overline{A}$ to hold removes the justification for persisting $A \vee B$. Furthermore, example 3 notes that the semantics of the domain rule

$$\texttt{Empty(Cup)} \Rightarrow \texttt{Liftable(Cup)}$$

does not warrant `Liftable(Cup)` persisting over a change in which `Empty(Cup)` is negated. On the other hand, example 2 notes that the semantics of the domain rule

$$\texttt{OnTable(Cup)} \Rightarrow \texttt{Liftable(Cup)}$$

may warrant `Liftable(Cup)` persisting over a change in which `OnTable(Cup)` is negated. For example, imagine that that domain rule simply reflects knowledge that a cup must have once been liftable if it ends up on the table top. Their solution to reflecting these semantics is to explicitly note in the domain model that `Empty` is *essential* to the persistence of `Liftable` whereas `OnTable` is *inessential*. Their result is roughly that a proposition $P$ persists by default

from a state $I$ to a state $G$ only if $P$ is true in $I$ and that, for some justification of $P$ holding in $I$, each antecedent is either true in $G$ or is inessential to $P$.

To consider how their results might apply to QR, we first need examples of both essential and inessential relations between state assumptions. Imagine that domain knowledge tells us that, when the water level (L) of a container is above a particular height (H), the resulting pressure due to gravity guarantes that the flow rate (F) of water flowing out is greater than the rate of a constant-flow pump pumping water in. The relation between $L > H$ and F seems essential, since when L drops below H the resulting pressure might well result in F being lower than the pump rate. Consider the water level L again, this time in relation to the mass (M) of water in the container. The relation between $L > bottom$ and $M > 0$ might also seem very essential, since when $L > bottom$ becomes false, $M > 0$ necessarily cannot persist.

Now, consider two pumps, one pumping in and one pumping out, connected to a container, each pump having its own constant flow rate. In that case, the direction of the derivative of the water level of that container is inessential to the inequality of the virtual rates of those two pumps, even though the direction may imply the virtual rates relation in some states. For instance, if the derivative is negative when the inward pump's source is not dry, then the inward pump's virtual pump rate is less than the outward pump's virtual pump rate. Regardless of other future changes in the on/off status of the pumps or that derivative, that relation between the virtual rates will persist. The key point here is that the virtual rates are constant, therefore they necessarily persist. Their being constant is the reason that their comparison is clearly inessential to the direction of the water level's derivative.

However, a deeper analysis suggests that most relations between state assumptions of QPT models will, in fact, be inessential in the Meyers and Smith sense. The underlying reason is that QR models are inherently biased toward continuous behavior. Therefore, they generally identify (symbolically, though usually not quantitatively) the exact turning point at which the negation of one proposition causes the negation of another related proposition.

For example, a reasonable QPT scenario model of their example 3 would involve constraints like the following:

$$(\texttt{mass(in-cup)} < \texttt{critical-mass}) \Leftrightarrow \texttt{liftable(cup)},$$

$$(\texttt{mass(in-cup)} = 0) \Rightarrow (\texttt{mass(in-cup)} < \texttt{critical-mass}),$$

$$(\texttt{mass(in-cup)} = 0) \Leftrightarrow \texttt{empty(cup)},$$

where `liftable(cup)` and `empty(cup)` are views and the other propositions are quantity conditions. Therefore, when the mass of the contents inside the cup is zero (i.e. the cup is empty), the cup is inferred to be liftable. When the mass becomes greater than zero, the cup is no longer empty, yet the cup is still liftable. Eventually, if the mass becomes great enough, the cup becomes unliftable. Even though the model does not commit to a specific value for the critical mass for liftability of the cup, QR refers to that concept to properly model changes over time.

Similarly, to model our earlier example above, we would typically define a critical height $H_c \leq H$ at which the flow rate equals the pump rate. Furthermore, the relation between $L > bottom$ and $M > 0$ need not be viewed as essential because the constraint $(L \leq bottom) \Rightarrow (M \leq 0)$ (due to `Same-Rel`$((L, bottom)(M, 0))$) directly indicates the conditions in which the persistence of $M > 0$ should be defeated.

In summary, we treat all justification relations between assumptions as inessential because QR models are naturally at a sufficient level of detail, due to modelling predominately continuous behavior, to allow doing so.

### 3.4.4 Priority-Based Measures of Minimality

From our perspective, it seems fair to say that priority-based measures of minimality essentially encode the hierarchy of the justification structure as priority levels. This approximation of justification-based minimal change appears to often suffice for examples in the minimal change literature. For example, the IMMORTAL work [3] demonstrated the application of priority-based PMA on a couple of small qualitative simulation examples. However, that approach is not sufficient for reasoning about discontinuities in derivatives like those that arise in the pump-cycle sudden-pump example. In this section, we explore the issue of why we do not need to prioritize state assumptions, even though we do not use a justification-based measure of minimality.

To reflect justification structure, antecedents would be given higher priority than consequences. Changes at lower priority levels are less important because they generally do not affect as many propositions. Consider complete states I, G1, and G2, where some update of I leads to two candidate results, G1 and G2. Let i denote the highest priority level for which

pairs (I,G1) and (I,G2) differ in their minimal change measures of closeness when it is applied to exactly those propositions at priority level i. A natural priority-based measure of minimal change is to prefer G1 over G2 if G1's level i priority propositions change less from I than G2's (and vis versa). That is the approach taken by prioritized PMA.

#### 3.4.4.1 Assumptions Provide the Key Prioritization

For QPT models, the most significant priorization occurs in our selection of what propositions to consider as assumptions. Assumptions are trivially at higher priority that non-assumptions, since we do not even explicitly consider changes in non-assumptions. In particular, non-assumptions such as process/view statuses are always inferrable from quantity condition and precondition statuses.

To appreciate the way in which our assumption/non-assumption distinction seems to naturally define the only necessary prioritization, consider an variant of Winslett's living room domain. In Winslett's formulations, the domain contains constraints such as:

$$\texttt{on(box,duct)} \Rightarrow \texttt{blocked(duct)},$$

$$\texttt{blocked(duct)} \Rightarrow \texttt{stuffy(room)}.$$

If `on` predicates are not prioritized higher than `blocked` and `stuffy` predicates, one can get counter-intuitive results. For example, moving the box off of the duct could allow some other box to move onto the duct, preserving the status of `blocked(duct)` in exchange for changing the `on` status of the second box. With prioritization, the exchange would not be equal, the duct becomes unblocked would be preferred over the second box jumping to the duct.

In natural QPT formulations of that domain, continuous quantities would be introduced to define this concept of `on` as a view, leading to constraints such as:

$$\texttt{location(duct)} = \texttt{location(box)} \Rightarrow \texttt{on(box,duct)}.$$

Since the quantity condition `location(duct) = location(box)` is a state assumption, its changes will be minimized, whereas the other propositions (which are simply view statuses) will not. Thus, the QPT formulation effectively automatically creates the appropriate prioritization for such examples.

#### 3.4.4.2 Why We Do Not Need to Prioritize Assumptions

The natural question to ask now is whether there might be cases in which some assumptions should be prioritized over other assumptions. After all, clearly there are justification structures in which assumptions justify other assumptions and priorities are really just an approximate way to reflect justification structure. However, we have not been able to imagine, nor come across during our QPT modelling experiences, examples requiring multiple priority levels for assumptions. In the end, it seems that the reason prioritization is not required among assumptions is the same reason that we can view all relations between assumptions as inessential, as discussed at the end of Section 3.4.3.

### 3.4.5 Set-Inclusion Versus Cardinality Measures of Closeness

Cardinality is an extremely efficient measure of minimality: simply count the number of propositions which change status. Unfortunately, it can also be extremely syntax-dependent. For example, consider updating a state $\{A, B, C, D\}$ by $\overline{A}$, with $\mathcal{M}$ containing $A \vee \overline{B} \vee \overline{C}$ and $C \Leftrightarrow D$. Cardinality-based minimality would prefer the update result $\{\overline{A}, \overline{B}, C, D\}$ (2 changes) over $\{\overline{A}, B, \overline{C}, \overline{D}\}$ (3 changes). In contrast, set-inclusion measures would prefer neither result over the other, since neither is a proper subset of the other. Thus, the set-inclusion result is not unreasonably biased by the particular encoding used to define the domain.

Forbus' AAE approach use a cardinality measure based, whereas our approach uses set-inclusion. AAE often gets away with the simpler cardinality measure because it does not consider a proposition an assumption when its status is inferrable in every state by other assumptions. For example, AAE does not consider as an assumption the sign of the derivative of a water level if there is only one process influencing that level. The statuses of the quantity conditions and preconditions (which are assumptions) of that process are sufficient to infer the Ds value of the water level, so AAE does not bother making it an assumption because it is never needed to distinguish a state.

In contrast, we consider all propositions suitable for $\mathcal{P}'$ as assumptions. Thus, we require set-inclusion measures of minimality because our assumption closures would otherwise lead us to the problems mentioned above.

#### 3.4.5.1 Why AAE's Cardinality Measure Is Insufficient For QR

Essentially, AAE avoids treating as assumptions those propositions which are equivalent to propositions which are already considered assumptions. The key point to make is that such an approach is insufficient to avoid the general problems with cardinality-based measures. One reason is that some assumptions will effectively be equivalent to others only in certain contexts.

For example, the pressure (P) at the bottom of a container might be zero when the level (L) of contained liquids is at the bottom, but only if the gas pressure ($P_g$) is assumed to be zero. If the gas pressure is positive, then the equivalence between $P = 0$ and $L = bottom$ does not hold. Thus, if $P = 0$, $L = bottom$, and $P_g = 0$ are all quantity conditions, then AAE would consider all three to be assumptions. Let proposition $C$ denote $P = 0$ and $D$ denote $L = bottom$ and proposition $X$ denote $P_g = 0$. With $X$ holding in all states for the example defined in terms of A,B,C,D above, these denotations illustrate how AAE's cardinality measure would lead to improper, syntactically-biased results — if the change in $A$ represented a discontinuous action change.

Such contextual constraints on the equivalences of assumptions appear to be rare in the relatively simple examples that have been rigorously examined in the QR literature to date. Their rareness is probably why such phenomena, and the importance of set-inclusion based measures of minimality, was not realized in the earlier AAE work. But for what it is worth, our analysis of the proper interplay of causality and minimality seems to put us on much firmer ground now.

# Chapter 4

# Finding State Paths

In this chapter we address the problem of finding paths of partial states sufficient to summarize all of the ways that any state compatible with goal state $\mathcal{G}$ could arise from any state compatible with initial state $\mathcal{I}$. Such a set of paths forms the "backbone" of our SUDE representation.

There are many techniques that one could use to search for sequences of state transitions which transform an initial state $\mathcal{I}$ into a goal state $\mathcal{G}$. Our work has focussed on the use of goal regression, analogous to STRIPS-like planning. Our main emphasis is identifying and addressing the special issues that arise due to the special nature of regression through QP constraints. Fundamentally, this involves fully appreciating the subtle differences between change due to derivatives and change due to actions.

## 4.1 Suitability of Regression for Qualitative Reasoning

First, we present some brief justification of why regression seems particularly well-suited for QR. After all, regression would typically be intractable for the related problem of showing all ways that a goal quantitative state could arise. The well-known general problem with regression is that it can explore pasts that cannot actually arise from the initial state. However, we suggest that qualitative regression should generally lead to less "dead ends" than qualitative simulation in connecting $\mathcal{I}$ with $\mathcal{G}$. First, for many tasks, $\mathcal{G}$ will be smaller than $\mathcal{I}$. For example, $\mathcal{G}$ for diagnosis or control tasks might be specified as a couple of fault/desired propositions, whereas $\mathcal{I}$ might be a complete set of observable propositions. Second, qualitative ambiguity appears to make excessive forward branching in QS common while at the same time minimizing the

```
(defOperator (Open-Valve ?pipe)
  Individuals ((?pipe :conditions (Has-Controllable-Valve ?pipe)))
  Conditions  ((Interval)
               (not (Aligned ?pipe)))
  Effects     ((Aligned ?pipe)))
```

**Figure 4.1**: Example action operator

need for regressing on large sets of conditions. For example, many physical processes have few conditions (on which to regress) and often one process being active is sufficient for a particular qualitative effect to be achievable.

Even if the backward search space is larger than the forward search space for a particular task, we suggest that the backward search space is more likely to be reusable across many future tasks. Compiling a large backward space from key goals is more likely to be useful over many tasks than compiling a large forward space from the particular initial states seen so far. The intuition is that for many tasks it may be more typical that there is a small common set of goal states than a small common set of initial states. For example, particular components may be most likely to fail or certain conditions (such as overflows) might be most important to control to avoid. Useful applications of QR for knowledge compilation, one of the motivations for constructing envisionments, seem more promising if the emphasis is on all ways to achieve specific $\mathcal{G}$'s rather than all futures of specific $\mathcal{I}$'s.

## 4.2 Types of Regression Operators

### 4.2.1 Action Operators

We adopt standard STRIPS-like operators for actions by merging add/delete lists into one effects list. Figure 4.1 gives an example operator for opening a valve on a pipe.

Recall that we insist that actions occur in `Interval` states, for reasons described in Section 3.1.3.

In continuity-based modelling, the effects of actions usually directly control the preconditions of processes or views. It is rare, and a dangerous modelling practice, to have actions in a dynamical world which directly assert quantity conditions. Since there is often a one-to-one

```
(defOperator (Flip-Precondition ?PC)
  Individuals ((Controllable ?PC))
  Conditions  ((Interval)
               (not ?PC))
  Effects     (?PC))
```

**Figure 4.2**: Simple action operator to flip preconditions

correspondence between PC's and actions, we introduce the following simplifying convention. Preconditions which are defined as controllable, such as

`Controllable(Aligned pipe12)`,

are interpreted as being controlled by action operators of the form specified in Figure 4.2.

Of course, one could model actions with multiple conditions and effects around single PC flips. The existing QPT modelling mechanisms are sufficient for doing so. Simply define views whose conditions are the multiple conditions and whose relations field asserts the `Controllable` proposition for a PC and define additional views conditioned on that PC whose relations field define the multiple direct effects. However, we have not explored specific examples of such extensions, since our focus has been on reasoning about dynamics — not the problem of regressing action operators.

### 4.2.2 Dynamics Operators

It is natural to distinguish the dynamic operators available to Nature into two broad categories: divergence and convergence. Figures 4.3 and 4.4 illustrate the general form of these two types of dynamic operators.

Note that we use a soft-inequality to represent the effect of each convergence operator. Doing so is convenient because then each soft-inequality proposition $P$ has exactly one generic operator that has $P$ as its effect. Thus, we can talk in terms of *the* generic operator for achieving a given soft-inequality QC proposition. In particular, we can achieve $Q1 = Q2$ either by explicitly achieving either $Q1 \leq Q2$ or $Q1 \geq Q2$. The other one will be implicitly achieved because we enforce continuity on all conditions of a dynamic operator. Which one we attempt

```
(defOperator (Divergence ?Q1 ?Q2)
  Instance-of ((Quantity-Condition ?Q1 ?Q2))
  Conditions  ((Instant)
               (= ?Q1 ?Q2)
               (> (D ?Q1) (D ?Q2)))
  Effects     ((> ?Q1 ?Q2)))
```

**Figure 4.3**: Generic divergence operator

```
(defOperator (Convergence ?Q1 ?Q2)
  Instance-Of ((Quantity-Condition ?Q1 ?Q2))
  Conditions  ((Interval)
               (< ?Q1 ?Q2)
               (> (D ?Q1) (D ?Q2)))
  Effects     ((>= ?Q1 ?Q2)))
```

**Figure 4.4**: Generic convergence operator

to explicitly achieve during regression search will depend on which one is known to be false in $\mathcal{I}$.

The derivative inequality condition of each dynamic operator indicates an assumption that the quantity condition is changing in the way required for the desired effect. For the sake of regression, we make that assumption as long as we can find a portion of influence structure, involving at least one process that could cause that change, which is sufficient for that derivative inequality to be consistent. Each such portion results in an alternative instantiation of the generic dynamic operator for a given quantity condition. We will discuss our mechanisms for performing such instantiation soon.

Compared to action operators, dynamics operators have two additional requirements for the effects to hold:

1. the conditions must last "long enough" (for a convergence to occur);
2. the influences supporting the derivative inequality assumption must not be dominated by opposing influences.

In short, an effect of a dynamics operator does not necessarily hold at any time after all the conditions hold.

Satisfying requirement 2 depends on the closed-world assumption on influences and the specific state it is applied in.

Some dynamics operators satisfy requirement 1 trivially. For example, consider a transition from $X > 0$ to $X = 0$ for a simple spring-block system, where X is the position of the block. $X > 0$ is the sole condition for a negative influence on X, ultimately due to the spring force. Thus, requirement 1 is satisfied until the change to $X = 0$ occurs. Unfortunately, such phenomena seem quite rare in QR.

In the majority of cases, where neither requirement is satisfied, we simply assume during regression that Nature cooperates as needed to allow $\mathcal{I}$ to reach $\mathcal{G}$. Accounting for situations in when Nature does not cooperate is the focus of the next chapter.

## 4.3 Formulating Dynamic Operators

One way in which our work differs from classical planning is that we automatically formulate our operators from the constraints of qualitative models. In particular, we refer to influence structure to instantiate the generic convergence and divergence operators in all possible minimal ways that could consistently support a particular dynamic change. In this section, we explain how we do this.

### 4.3.1 Sufficient Support States for Derivative Inequalities

We define the *sufficient support states* for a derivative inequality $D(Q1) < D(Q2)$ to be those partial states which minimally describe the conditions under which the proposition `(Possibly (< (D Q1) (D Q2)))` is necessarily true. Figures 4.5 and 4.6 present the rules with which we augment QPT models to support this reasoning. The basic idea behind these rules is that a particular derivative inequality can possibly hold as long as there some influence on one of those quantities in the necessary direction and it is not overwhelmed by other influences in the opposite direction for the same quantity. An influence is overwhelmed when there is an opposing influence whose rate is at least as strong.

More precisely, these rules will define the smallest alternative sets of influences necessary to overcome known competing influences. To support this, we introduce conjunctive rates as needed to define new propositions representing *conjunctive direct influences*, based on multiple processes, which together are not overwhelmed by those competing influences. Our generalization of direct influences into conjunctive direct influences is why we use the predicates `:I+` and `:I-` to represent direct influences in our possible derivative rules.

For example, consider a quantity Q with its only three influences being

`(I+ Q rate1)`,

`(I+ Q rate2)`,

`(I- Q rate3)`

and a single universal rate constraint

`(Rate>= rate3 rate1)`.

`(Overwhelmed (:I+ Q rate1))` will hold under the conditions in which `(I- Q rate3)` holds. In contrast, `(Overwhelmed (:I+ Q (+ rate1 rate2)))` will be false under those same conditions, due to our enforcing CWA's on all `Rate>=` propositions. So, the conditions supporting the conjunctive direct influence is the union of the conditions support the individual direct influences. This is reflected by adding justifications like the following for each conjunctive direct influence introduced:

$$\texttt{(:I+ Q rate1)} \wedge \texttt{(:I+ Q rate2)} \Rightarrow \texttt{(I+ Q (+ rate1 rate2))}.$$

For example, we have used this approach to determine that the minimal support for a water level rising when there were two pumps into the container and one pump out required both pumps, because the pump out was known to be stronger than either other one individually. We have also explored the use of `Rate>=` constraints to model non-universal constraints, such as a pump strong weaker than a gravity flow when the water level is below some particular limit point. However, we have not thoroughly explored the ramifications of their use. Our primary concern was to identify a plausible general view of how we should define sufficient support states in terms of influence structure.

```
(==> (and (Quantity-Condition ?Q1 ?Q2)
          (or (Possibly-Decreasing ?Q1) (Possibly-Increasing ?Q2)))
     (Possibly (< (D ?Q1) (D ?Q2))))

(==> (or (PossibleD- ?Q ?rate)
         (and (PossibleD- ?Q1 ?rate) (:Q+ ?Q ?Q1))
         (and (PossibleD+ ?Q1 ?rate) (:Q- ?Q ?Q1)))
     (Possibly-Decreasing ?Q))

(==> (or (PossibleD+ ?Q ?rate)
         (and (PossibleD- ?Q1 ?rate) (:Q- ?Q ?Q1))
         (and (PossibleD+ ?Q1 ?rate) (:Q+ ?Q ?Q1)))
     (Possibly-Increasing ?Q))

(==> (and (:I+ ?Q ?rate) (not (Overwhelmed (:I+ ?Q ?rate))))
     (PossibleD+ ?Q))

(==> (and (:I- ?Q ?rate) (not (Overwhelmed (:I- ?Q ?rate))))
     (PossibleD- ?Q))

(==> (and (:I+ ?Q ?rate+) (:I- ?Q ?rate-) (Rate>= ?rate- ?rate+))
     (Overwhelmed (:I+ ?Q ?rate+)))

(==> (and (:I+ ?Q ?rate+) (:I- ?Q ?rate-) (Rate>= ?rate+ ?rate-))
     (Overwhelmed (:I- ?Q ?rate-)))

(==> (I+ ?Q ?rate) (:I+ ?Q ?rate))

(==> (I- ?Q ?rate) (:I- ?Q ?rate))
```

**Figure 4.5**: Possible derivative rules

Recall from Section 2.3.3 that we assume that all rates are non-negative. That assumption also avoids the need to include here support for `PossibleD+` (`PossibleD-`) propositions due to `I-` (`I+`) propositions.

```
(==> (or (Qprop+ ?Q1 ?Q2)
         (M+ ?Q1 ?Q2)
         (and (:Q+ ?Q1 ?Qi) (:Q+ ?Qi ?Q2))
         (and (:Q- ?Q1 ?Qi) (:Q- ?Qi ?Q2)))
     (:Q+ ?Q1 ?Q2))

(==> (or (Qprop- ?Q1 ?Q2)
         (M- ?Q1 ?Q2)
         (and (:Q- ?Q1 ?Qi) (:Q+ ?Qi ?Q2))
         (and (:Q+ ?Q1 ?Qi) (:Q- ?Qi ?Q2)))
     (:Q- ?Q1 ?Q2))
```

**Figure 4.6**: Qprop chain rules

We also note that we actually implement the qprop chain rules procedurally rather than declaratively, because it is much more efficient to do so. Otherwise, excessively many intermediate `:Q+` and `:Q-` propositions get introduced that are never useful in providing justification structure to the `Possibly` propositions that we are interested in. It is for similar reasons that we introduce conjunctive direct influences procedurally as well.

### 4.3.2 Computing Complete Sets of Sufficient Support States

There is a one-to-one correspondence between sufficient support states and ATMS labels. In particular, the complete set of sufficient support states for a `Possibly` proposition is the ATMS label of the ATMS node representing it. The ATMS label represents minimal alternative sets of QC and PC assumptions that support that particular proposition being true. Each ATMS environment (i.e. set of assumptions) corresponds to a sufficient support state. Because we explicitly make CWA's that the model $\mathcal{M}$'s influence structure is complete, we can assume that the label represents a complete set of sufficient support states.

Unfortunately, the overall complexity of an ATMS can easily be prohibitive for this task. A standard ATMS precomputes and maintains the labels of all nodes, not just the ones of particular interest, such as our `Possibly` propositions. A key underlying intuition of an ATMS is that doing so may provide amortized savings versus a node by node computation as needed. However, we have found this not to be the case for us.

During the development of this work, we have come to realize that the only labels of explicit interest to us are, in fact, those of the `Possibly` propositions. It turns out to be extremely efficient to compute the label for each such proposition via straight-forward backchaining from its node through the ATMS justification structure and grounding out at QC and PC assumptions. By computing assumption closures on each environment in the resulting label, we are guaranteed that those labels are sound, minimal, and complete.

The reason that backchaining to compute the labels of `Possibly` nodes is particularly efficient comes from the causal, hierarchical nature of QPT process/view structure. The justification structure of each `I+` or `I-` proposition forms a simple acyclic AND/OR graph, whose leaves are all QC and PC propositions and whose internal AND nodes represent the status propositions of views. The top-level AND represents a process status proposition, whose single justification's antecedents are the quantity conditions, preconditions, and the view propositions referenced in the individuals and instance-of fields. The OR nodes represent the fact that we allow the same view status proposition to be justified more than once, representing alternative ways for that view to hold.

Note that the individuals of processes and views generally act like plan filters [7]. No attempts to achieve them will be made during regression, since they are either true or assumed false due to CWA's. The exception is when individuals contain views, but generally we place views in the instance-of field.

For efficiency, we cache the resulting labels of all nodes we touch during backchaining. We further cache updates to labels that occur during BCP, when we discover that particular nodes are true for particular query environments (such as the computation of state assumption closures). Essentially what we have done is provide an incremental means of computing ATMS labels on an as-needed bias. It essentially starts out acting like a JTMS [19] but over time compiles enough label information to begin reaping the benefits of an ATMS. Our experience suggests that such hybrid TMS approaches are necessary to avoid the up-front overhead of an ATMS, which would otherwise defeat our attempts to conform the complexity of QR to the task as hand, as opposed to the complexity of the specific model.

### 4.3.3 Using Sufficient Support States to Formulate Dynamic Operators

We formulate each dynamic operator as follows.

1. Let P be a QC soft-inequality proposition.
2. Let OP represent *the* generic dynamic operator for P.
3. Let SSS be the set of sufficient support states for the `Possibly` proposition for the derivative inequality (D) in OP's conditions.
4. For each state S' in SSS do:
   (a) Initialize state S as OP's conditions minus D.
   (b) Union S' into S (i.e. replacing D with S').
   (c) Update S to be its state assumption closure.
   (d) Apply all forward temporal rules (of Figure 3.1) for S.
   (e) Initialize effect state E to be the set $\{P|\texttt{Next(P)} \in S\}$.
   (f) Union OP's effects (namely $\overline{P}$) into E.
   (g) Update E to be its state assumption closure.
   (h) If E is contradictory, then the transition $S \rightarrow E$ cannot occur (no matter how long S lasts).
   (i) Otherwise, store a new dynamic operator for P, with the conditions being S and the effects being E.

We will explain a specific example of this process below shortly.

Notice that we replace D with S' to provide base QC/PC conditions for the instantiated operator. We drop D to maximize the opportunities that other SUDE paths will be able to pass through states we build upon S during regression. The fact that D must hold at least at the end of S in order for it to transition to E can easily be recovered by local reasoning with the specific state built upon S and state build upon E. In general, our regression techniques follow the principle of only adding state details which are necessary in order to propagate global constraints. If D necessarily holds in S, it will be recovered when S's assumption closure is computed.

#### 4.3.3.1 Pruning Operators Due To Discontinuities

It is important to realize that effect state E becomes contradictory if sufficient support state S' would have to change discontinuously to allow $P$ to occur in the next state of S'.

For example, consider the goal of emptying a container of liquid. With $L$ representing the liquid level in the container, that goal is specified as $L = bottom$. One sufficient support state for possibly decreasing L is based on an overflow process, whose key quantity condition is $L > top$. However, trying to empty a container by overflowing it is futile. [1] Forward continuity on $L > top$ insists that $L \geq top$ holds in E. Since $top > bottom$ universally holds for containers, the closure of E will contain $L > bottom$. Yet, that is inconsistent with P insisting that $L = bottom$ is in E. Thus, E is inconsistent and the operator is not produced, indicating that overflow is not a valid process for achieving $L = bottom$.

#### 4.3.3.2 Refining Operators Due to Continuity

Sometimes continuity constraints also indicate additional conditions for the operator. Having preidentified such conditions during formulation is particularly valuable when it comes time to order candidate operators during regression.

Consider the example of two containers of water (CAN1 and CAN2) connected by a horizontal pipe connecting their bottoms. Let us consider a change toward overflowing CAN1, represented by

$$P \equiv \texttt{level}_1 \geq \texttt{top}_1.$$

The sole sufficient support state for $\texttt{level}_1$ rising is based on a water flow process from CAN2 to CAN1. Continuity constraints further tell us that $\texttt{level}_2 \geq \texttt{top}_1$ is required in order for this flow process to last long enough to achieve $P$. Below we present a successive formulation of the dynamic operator which reflects these constraints. When presenting the results of closures and continuity, we only show enough to justify the resulting operator, since there are many other equivalent assumptions whose presentation would merely be distracting.

[1] Of course, one can imagine special cases, such as overflowing allowing enough liquid to be lost that some limited pump process, that might otherwise fail, finishes the job. However, explicit qualitative conditions specifying exactly when overflowing before pumping is required seem difficult to formulate. In any case, when such conditions are not present, there seems to be no point in bothering to consider both happening in the same behavior path.

```
conditions:
   (Interval)
   (<  (height (liquid-in CAN1)) (height (top CAN1)))
   (>  (D (height (liquid-in CAN1))) (D (height (top CAN1))))
effects:
   (>= (height (liquid-in CAN1)) (height (top CAN1)))
```

**Figure 4.7**: Stage 1: Instantiating the generic convergence operator for P

```
conditions:
   (Interval)
*  (<  (pressure (bottom CAN1) :ABSOLUTE) (pressure (bottom CAN2) :ABSOLUTE))
   (<  (height (liquid-in CAN1)) (height (top CAN1)))
*  (>  (height (liquid-in CAN2)) (height (bottom CAN2)))
*  (>  (mass (C-S WATER LIQUID CAN1)) 0)
*  (>  (mass (C-S WATER LIQUID CAN2)) 0)
effects:
   (>= (height (liquid-in CAN1)) (height (top CAN1)))
```

**Figure 4.8**: Stage 2: Inserting sufficient support state (before closure)

We use *'s to denote additions from the previous stage.

```
conditions:
   (Interval)
   (<  (pressure (bottom CAN1) :ABSOLUTE) (pressure (bottom CAN2) :ABSOLUTE))
*  (<  (height (liquid-in CAN1)) (height (liquid-in CAN2)))
   (<  (height (liquid-in CAN1)) (height (top CAN1)))
   (>  (height (liquid-in CAN2)) (height (bottom CAN2)))
   (>  (mass (C-S WATER LIQUID CAN1)) 0)
   (>  (mass (C-S WATER LIQUID CAN2)) 0)
effects:
   (= (height (liquid-in CAN1)) (height (top CAN1)))
*  (<= (height (liquid-in CAN1)) (height (liquid-in CAN2)))
```

**Figure 4.9**: Stage 3: After conditions closure and forward temporal constraints

Note that there is a condition that the water mass of CAN1 must be positive, which might at first glance appear unnecessary. That condition in the sufficient support state comes from the domain constraint that a qprop relation between mass and level only holds when mass is positive. Mass is what is directly influenced by the flow process, so that qprop is what conveys the sufficient support state of increasing mass to increasing level. If the domain stated that the qprop relation held when mass was zero, that would incorrectly imply that decreasing zero mass would negatively influence level. The bottom line is that such tedious attention to detail that QPT requires due to its continuity-based nature will be reflected in the dynamic operators.

Note that stage 3 infers the weak effect $\texttt{level}_2 \geq \texttt{level}_1$. The stronger $\texttt{level}_2 > \texttt{level}_1$ is not warranted by local continuity constraints, even though a local minimal change approach would make that strong inference. In fact, $\texttt{level}_2 = \texttt{level}_1$ *is* a valid result of this operator. For example, if there is another container (CAN3) flowing into CAN2 and $\texttt{level}_2 = \texttt{top}_1$, then $\texttt{level}_2$ might remain steady while, in effect, CAN3 flows into CAN1. Such a special case requires that the flow rates from CAN1 to CAN2 and from CAN3 to CAN2 are identical. The point is that we avoid inferences due to minimal change during the formulation of dynamic operators to avoid the unsoundness that could result, as the above example illustrates.

The key to making the new inference in the conditions of Stage 4 is to realize that if $\texttt{level}_2 < \texttt{top}_1$ was added instead, then transitioning to the effects closure would be impossible. We perform this reasoning by checking, for each proposition $p$ in the effects closure, whether adding $\overline{p}$ to the conditions of the operator would make the operator inconsistent.

```
conditions:
   (Interval)
   (<  (pressure (bottom CAN1) :ABSOLUTE) (pressure (bottom CAN2) :ABSOLUTE))
   (<  (height (liquid-in CAN1)) (height (liquid-in CAN2)))
   (<  (height (liquid-in CAN1)) (height (top CAN1)))
*  (>= (height (liquid-in CAN2)) (height (top CAN1)))
   (>  (height (liquid-in CAN2)) (height (bottom CAN2)))
   (>  (mass (C-S WATER LIQUID CAN1)) 0)
   (>  (mass (C-S WATER LIQUID CAN2)) 0)
effects:
   (>= (height (liquid-in CAN1)) (height (top CAN1)))
   (<= (height (liquid-in CAN1)) (height (liquid-in CAN2)))
*  (>= (height (liquid-in CAN2)) (height (top CAN1)))
```

**Figure 4.10**: Stage 4: After effects closure and new change checks

The reason that a condition of $\texttt{level}_2 < \texttt{top}_1$ would conflict with the effects is fundamentally that it would require both $\texttt{level}_1$ and $\texttt{level}_2$ to converge to $\texttt{top}_1$ at the same time. Since $\texttt{level}_2$ begins higher than $\texttt{level}_1$, there is no way for that to be the next change involving these particular conditions. Either $\texttt{level}_2$ converging to $\texttt{top}_1$ or $\texttt{level}_1$ converging to $\texttt{level}_2$ would occur first. In either case, those changes would violate the conditions of the operator before the effects are achieved.

```
conditions:
   (Interval)
   (<  (height (liquid-in CAN1)) (height (liquid-in CAN2)))
   (<  (height (liquid-in CAN1)) (height (top CAN1)))
   (>= (height (liquid-in CAN2)) (height (top CAN1)))
   (>  (mass (C-S WATER LIQUID CAN1)) 0)
   (>  (mass (C-S WATER LIQUID CAN2)) 0)
effects:
   (>= (height (liquid-in CAN1)) (height (top CAN1)))
```

**Figure 4.11**: Stage 5: Final operator for P (minimized conditions/effects)

Finally, after stage 5, we obtain the final dynamic operator. For presentation purposes, we have removed all propositions for which closures or forward temporal rules would immediately recover. However, the pre-minimized conditions and effects are retained, since they provide seeds for later closures upon these conditions and effects sets during regression.

Note that this particular operator assumes that $\texttt{level}_2$ is above or at $\texttt{top}_1$ already. It is certainly conceivable, however, that the flow process underlying this operator could be moving $\texttt{level}_1$ toward $\texttt{top}_1$ even while $\texttt{level}_2 < \texttt{top}_1$ holds. If there was some other process (like a flow from a CAN3) which was simultaneously rising $\texttt{level}_2$, then certainly the $\texttt{level}_2 \geq \texttt{top}_1$ condition is not required when the actual flow from CAN2 to CAN1 begins. All our operator specification is saying is that the portion of behavior space which is *sufficient* to achieve $\texttt{level}_1 = \texttt{top}_1$ can be captured by focusing on behavior once $\texttt{level}_2 \geq \texttt{top}_1$ occurs.

### 4.3.4 Using Dynamic Operators to Assess QP Models

We have found the exercise of formulating all dynamic operators for all quantity conditions to be a useful modelling tool as well. Dynamic operators indicate interesting local chunks of causality. Many types of bugs in a qualitative model can be deduced from the absence of expected dynamic operators or the identification of counter-intuitive ones. We suggest that such analyses might offer much higher insight-to-cost ratios than those based on examining QS-generated envisionments.

## 4.4 Goal-Based Regression

Armed with a set of action and dynamics operators, we regress from $\mathcal{G}$ to $\mathcal{I}$ in a manner similar in style to state-based planners. In this section we outline the control strategy we use. In the next section we discuss a variety of special issues that arise due to the differences between dynamics-based QR and action-based planning.

Calling FIND-PATHS with arguments $\mathcal{I}$ and $\mathcal{G}$ results in a forest of trees, each tree having as its root a refinement of $\mathcal{G}$ and its leaves being refinements of $\mathcal{I}$. Call this forest $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$. Generally, the leaves will be larger than the roots, since regression picks up additional conditions but we do not forward project such conditions unless they necessarily persist.

```
Procedure FIND-PATHS (I,G):

1.  Let global priority queue PQ contain single entry I,G.
2.  Until PQ is empty do:
3.    Let i,g = Dequeue(PQ).
4.    REGRESS(i,g).
```

**Figure 4.12**: Procedure FIND-PATHS

```
Procedure REGRESS (I,G):

 1.  Let FLIPS = Falses(I,G).
 2.  If FLIPS ≠ ∅
       then
 3.      Let D = ∅.
 4.      For each F ∈ FLIPS do:  Let D = REGRESS-TO-FLIP(I,G,F,D).
 5.      HANDLE-DISCONTINUOUS-OPS(I,G,D,FLIPS).
       else
 6.       S = UNION-STATES(I,G).
 7.       if S is consistent
 8.         then
 9.            Add S to Refinements(I) and to Refinements(G).
10.            Record ''transition'' from S to G, say due to "I-G-collapse".
11.         else  REGRESS-TO-FLIP-DURATION(I,G).
```

**Figure 4.13**: Procedure REGRESS

Summary: For each PC and QC proposition in G which is false in I, explore all regressions for achieving it (i.e. "flip" it). If there are no differences and the union of I and G is consistent, paths from $\mathcal{I}$ to $\mathcal{G}$ has been found. Otherwise, try flipping the duration proposition of G, in case I happens to conflict from G only in duration.

```
Procedure REGRESS-TO-FLIP (I,G,F,D):

1.  For each OP ∈ OPS(F) do:
2.    Let PREVS = PREVIOUS-STATES-VIA-OP(G,OP).
3.    If PREVS = ∅
4.      then Add OP to D.
        else
5.        Let global RegressFlag = true.
6.        For each S ∈ PREVS do:
7.          Record transition from S to G via OP.
8.          Enqueue (I,S) onto PQ.
9.  Return D.
```

**Figure 4.14**: Procedure REGRESS-TO-FLIP

Summary: Regresses on all applicable operators for flipping proposition F from false to true and also returns operators which cannot regress (due to discontinuities between G and OP's conditions).

```
Procedure HANDLE-DISCONTINUOUS-OPS (I,G,D,FLIPS):

1.  Let NEWS = ∅.
2.  For each OP ∈ D do:
3.    Let S = CONDITIONS(OP).
4.    For each F ∈ Falses(S,G) do: If F ∉ FLIPS then Add F to NEWS.
6.  D = ∅.
7.  For each F ∈ NEWS do: Let D = REGRESS-TO-FLIP(I,G,F,D).
8.  If D ≠ ∅ then HANDLE-DISCONTINUOUS-OPS(I,G,D,FLIPS ∪ NEWS).
```

**Figure 4.15**: Procedure HANDLE-DISCONTINUOUS-OPS

Summary: Provides a simple way for handling conflicting conjunctive goals.

- `UNION-STATES(S1,S2)` unions the assumption closures of S1 and S2 and returns the assumption closure of the union.
- `OPS(P)` returns all operators applicable to proposition P.
- `PREVIOUS-STATES-VIA-OP(G,OP)` applies our minimal change algorithm (see Section 3.3.1.3) to G unioned with the effects of OP. The update is the conditions of operator OP. Recall that the effects of OP include the forward temporal constraints from its conditions. Thus, `PREVIOUS-STATES-VIA-OP` returns $\emptyset$ when G would require one of OP's conditions to change discontinuously.

**Figure 4.16**: Summary of functions referenced by calls to FIND-PATHS

### 4.4.1 Conjunctive Subgoal Conflicts Seem Rare

`HANDLE-DISCONTINUOUS-OPS` provides a simple way to handle cases of Sussman's Anomaly, where one must move away from some subgoal proposition $g \in G$ which is consistent with I, in order to solve some other subgoal proposition $f \in G$ which is false in I. These cases arise when there is some operator for achieving $f$ whose conditions include $\overline{g}$.

Figure 4.17 shows an example that arose in our work. That dynamic operator requires a discontinuous change in `location`'s relation to `right-end(stove)`. This is detected by a conflict between G and the effects of the operator for that inequality.

Note that an approach based purely on minimal change would, incorrectly, assume that it could achieve the base effect of this operator. It would not realize there are other effects, due to continuity holding on the other conditions of this operator. Thus, it would simply update G as needed to be consistent with its conditions. That would cause `location < right-end(stove)` to hold in the new resulting subgoal — essentially performing wishful thinking which violates causality.

It should be noted that we have found conjunctive subgoal conflicts to be very rare in qualitative models. Our experience is that they mainly arise due to spatial conditions conflicting with other dynamic conditions, as was the case in the example given above. Few domains that have been explored by the QP community have sufficiently complex interaction constraints for such conflicts to arise. Furthermore, it has proven difficult to imagine many such interactions

```
I: location > right-end(stove)
   mass(gas) = 0

G: location > right-end(stove)
   mass(gas) > 0

Operator:
  Conditions:   location < right-end(stove)
                location > left-end(stove)
                mass(water) > 0
                mass(gas) = 0
  Effects:      mass(gas) > 0
                location <= right-end(stove)
                location >= left-end(stove)
                mass(water) >= 0
```

**Figure 4.17**: Example of a Discontinuous Dynamics Operator for I and G

This operator is based on a boiling process which is active when a pot of water is on a stove.

for arbitrary domains. This seems to be part of the reason that our regression techniques tend to not lead to many dead ends.

### 4.4.2 Types of Subgoals

Figure 4.18 partitions the propositions of G into four disjoint sets. To compute these partitions, we first compute a set of *BaseFacts* for G. *BaseFacts* consists only of PC and QC propositions, since those are the only propositions which can have operators in our approach. Furthermore, *BaseFacts* is minimized to not include propositions which are inferable from the others in that set. Of course, such minimization can result in alternative sets. For our purposes, finding an optimally small set is not critical. In fact, our only reason for computing *BaseFacts* is to avoid redundant regression.

We use a simple three-step greedy heuristic to compute *BaseFacts*. First, initialize *BaseFacts* to the set of *base* PC and QC propositions in G. [2] Second, remove from *BaseFacts* all propositions which necessarily persist from I (including all universally propositions). Third,

[2]We cache the initial set of (base) propositions used to specify a given state S. Thus, even when we update S to be its assumption closure, we have access to that base subset, for seeding the computation of *BaseFacts*.

$Falses(I, G) \equiv \{\forall p \in BaseFacts(G) \mid \overline{p} \in I\}$

$Trues(I, G) \equiv \{\forall p \in BaseFacts(G) \mid p \in I\}$

$Unknowns(I, G) \equiv \{\forall p \in BaseFacts(G) \mid \overline{p} \notin I \wedge p \notin I\}$
$NonBaseFacts(G) \equiv \{\forall f \in G \mid f \notin BaseFacts(G)\}$

**Figure 4.18**: Four Types of Subgoals

remove from *BaseFacts* each proposition known to be universally dependent on others still in that set.

One could try further removals, computing a new closure set after each so that it can be returned if the closure does not infer it. However, we found the complexity of such reasoning, even if done once per proposition, unnecessary. Our three-step approach tends to be sufficient to find optimal *BaseFacts*. This seems to due, in part, to the heavy bias in QR on binary constraints that we noted in Chapter 2. Besides, we can safely restrict regression to those propositions in *BaseFacts* only if we can ensure that achieving each of those propositions will necessarily result in the full closure of G being achieved. In any case, we are willing to accept the small possibility of slightly redundant regression search time in exchange for fast computation of *BaseFacts* (which must be done for every regression state).

## 4.5 Special Issues

Having presented the basic ideas of our regression algorithm, we now turn to discusing several special issues that we have identified.

### 4.5.1 Incomplete Initial State

We typically compute all of $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$ while only subgoaling on propositions in *falses*. We only subgoal on *trues* in those rarer cases where `HANDLE-DISCONTINUOUS-OPS` realizes that doing so is necessary to handle goal conflicts. However, it is important to realize that we never subgoal on propositions in *unknowns* during the development of $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$. This reflects an assumption that it is sufficient to know how $\mathcal{G}$ can arise from the states on the outer fringe of $\mathcal{I}$'s corresponding phase space region. The main consequence of this assumption is that

$\text{SUDE}_{\text{base}}(\mathcal{I}, \mathcal{G})$ does not articulate the space of behavior *within* partial state $\mathcal{I}$, whereas $\text{AE}(\mathcal{I})$ does.

Subgoaling on propositions in *unknowns* is particularly important when one wants to identify the states within $\mathcal{I}$'s phase space region from which $\mathcal{G}$ would never arise. We explore this issue in the next chapter.

### 4.5.2 The Need For Simultaneous Dynamic Operators

Enforcing continuity constraints between the conditions and effects of operators can lead to subtle results that do not arise in planning with discrete operators. In particular, we discovered that there are cases in which we must regress on multiple dynamic operators conjunctively, to account for simultaneous processes which necessarily depend on each other to cause a desired change.

Consider our earlier example of two containers (CAN1 and CAN2) connected at their bottoms by a horizontal pipe. Now, add a third container (CAN3), whose bottom is lower than the other two, and connect a horizontal pipe from the bottom of CAN2 to the side of CAN3. Consider only water in the containers. In order to achieve a goal of getting CAN1 empty, it is necessary for CAN2 to empty into CAN3 while CAN1 empties into CAN2. Thus, a flow process from CAN1 to CAN2 and a flow process from CAN2 to CAN3 are both required to occur simultaneously.

Figure 4.19 outlines the flow of our algorithm on this example. We only present enough details of each state to illustrate how continuity and closures are used to infer the need to regress on a conjunction of dynamic operators.

The key enhancement required to the algorithm we presented earlier is the realization that one should regress on new propositions like $L2 \leq B1$. We assume that $L2 \leq B1$ being newly inferable in the closure of G', after applying forward continuity from S1 to G, makes it worth regressing on. [3] In general, we assume that we should regress on any new propositions which are inferable in the closure after applying forward continuity.

[3]Note that we regress on $L2 \leq B1$ (of G1) instead of $L2 = B1$. Recall that hard-inequality $L2 = B1$ is actually encoded as the conjunction of soft-inequalities $L2 \leq B1$ and $L2 \geq B1$. Since $L2 \geq B1$ is universally true, we need only regress on $L2 \leq B1$.

Notice that our approach also realizes that achieving $\mathcal{G}$ requires that CAN3's water level is initially below its side portal height. If $\texttt{level}_3$ was above its side port height, there would be no way for CAN2 to empty into CAN3.

Admittedly, the need for conjunctive dynamic operators appears to be a quite rare phenomena. Perhaps it is even unique to non-turbulent flows. In any case, it does appear to be an issue which is unique to continuous domains and thus has not arisen from previous work in classical planning.

### 4.5.3 Allowing Instant-Instant Transitions

As discussed in Section 3.1.1, allowing unstable equilibrium is a standard way to account for changes in higher-order derivatives. It requires strict alternation of `Instant` and `Interval` states. However, it turns out that allowing successive `Instant` states leads to a more natural regression algorithm.

For example, consider our previous three container example. Assume that $\mathcal{G}$ specifies that CAN1 contains water and $\mathcal{I}$ specifies that CAN1 and CAN2 are empty but $\texttt{level}_3$ is above CAN3's side portal. The natural solution is that CAN3 instantly starts flowing into CAN2 and then CAN2 instantly starts flowing into CAN1. Our regression algorithm finds this solution because it uses forward temporal rule f7 instead of rule F7.

### 4.5.4 Avoiding Forced Clobberings

Assume that regression is currently considering the transition

$$S \rightarrow G$$

to achieve currently subgoal G. Before regressing on S, it might be useful to consider what further conditions on S might prevent that transition from occurring For the most part, we do not reason during regression about how a transition may or may not occur, since we advocate doing that in a later qualification sweep over $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$. However, toward ensuring that $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$ is sound and to avoid garden-path regression, we have found it useful to allow one exception. In particular, we refine S as needed to ensure that the transition is *possible.*

For example, consider an example from our kitchen-world domain. Our goal is for our container to have steam but no water, given an initial state in which the container is empty. In

```
Initial problem definition:
I:      L1 >  B1
G:      L1 =  B1

S1 <-- regressing on G for dynamic operator for flow from CAN1 to CAN2:
S1:     L1 >  B1
        L1 >  L2      ;; due to process conditions for flow
G:      L1 =  B1

G1 <-- closure of applying forward continuity from S1 to G:
S1:     L1 >  B1,  L1 >  L2
G1:     L1 =  B1
        L1 >= L2      ;; via forward continuity from S1
        L2 <= B1      ;; via transitivity on B1=L1 and L1>=L2
        L2 <= B2      ;; via transitivity on L2<=B1 and universal B1=B2
        L2 =  B2      ;; via L2<=B2 and universal L2>=B2

S2 <-- regressing upon S1 for L2=B2, via op for flow from CAN2 to CAN3:
S2:     L1 >  B1,  L1 >  L2,  L2 >  L3
G1:     L1 =  B1,  L1 >= L2,  L2 <= B1,  L2 =  B2

G2 <-- closure of applying forward continuity from S2 to G':
S2:     L1 >  B1,  L1 >  L2,  L2 >  L3
G2:     L1 =  B1,  L1 >= L2,  L2<= B1,  L2 =  B2
        L2 >= L3      ;; via forward continuity from S2
        L3 <= B2      ;; via transitivity on B2=L2 and L2>=L3
        L3 <= SIDE3   ;; via transitivity on L3<=B2 and universal B2=SIDE3

S3 <-- try to regress on L3 <= SIDE3, realize that has no dynamic operators:
S3:     L1 >  B1,  L1 >  L2,  L2 >  L3
        L3 <= SIDE3   ;; back persist from G2 since otherwise G2 is impossible
        L3 <  SIDE3   ;; since L3=SIDE3 would not allow transition to G2
G2:     L1 =  B1,  L1 >= L2,  L2 =  B2
        L2 >= L3,  L3 <= B2,  L3 <= SIDE3

Detect no more regression from G2 possible, ready to regress from S3.
Detect that S3 is compatible with I, so stop regression.

Result: S3 is refinement of I which must hold for G to be a next of I.
```

**Figure 4.19**: Handling the multiple emptying containers example

For presentation simplicity we use the following abbreviations for container i. SIDE3 refers to the height of the side portal in CAN3, Li refers to $\texttt{level}_i$, and Bi refers to $\texttt{bottom}_i$.

our kitchen there is a faucet between our goal destination (counter3) and a stove. A reasonable plan is to move to the faucet, turn on the faucet, move to the stove, turn on the stove, wait until all water is boiled into steam, and finally move back to counter3. Unfortunately, if we forget to turn off the faucet, we will necessarily add water back to the container when we pass under it on the return trip. Thus, it is important to realize that we must turn off the faucet before we pass under it on the return trip.

We handle such cases by ensuring that we augment S in all minimal ways to be inconsistent with some conditions of all dynamic operators whose necessary effects include the negation of some proposition in G. The operator representing flow from the faucet causing a positive liquid level in the container is such an operator, since we model no negative influences on the container's water mass when it is under the faucet. That operator is conditioned on:

1. the faucet being on,
2. the container being under the faucet, and
3. the level of liquid in the container being empty.

We cannot negate condition 3, since that would clobber a condition we are trying to regress. Similarly, we cannot negate condition 2, since that would be discontinuous with the goal of moving to the limit point between the faucet region and the counter3 region. So, for this example, we merely include in S that the faucet must be off.

Even for cases where more than one condition is added to S, we believe that it is generally desirable to explicitly regress such alternative necessary conditions. However, if G is ultimately unachievable, introducing these alternative ways to regress will multiple the cost in undesirable regression search. Clearly, better control strategies will be required to address this issue.

It is important to acknowledge that clobberings due to necessary effects are rather special. Ultimately, they are based on the simple notion that a quantity being at a limit point will be clobbered if there is an unopposed influence on it. This approach does not address the related notion that turning off the faucet would maximize the chance of achieving a goal of keeping the final water level below some limit point.

# Chapter 5

# Qualifying State Paths

The $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$ trees resulting from the regression method of the previous chapter do not present a full discriminatory-based account of the possible behavior of the system. The paths in $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$ identify sufficient conditions for $\mathcal{G}$ to possibly arise from $\mathcal{I}$, if Nature cooperates. In particular, $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$ does not account for:

- refinements of $\mathcal{I}$ which would make $\mathcal{G}$ impossible.
- refinements of $\mathcal{I}$ which would make $\mathcal{G}$ inevitable.
- for each state S in $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$, what happens if S transitions to a next state incompatible with the nexts recorded in $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$. We refer to such behavior as "falling off" the regression paths.

In this chapter, we present techniques for addressing these shortcomings in $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$, resulting in a complete SUDE, which we will refer to as $\texttt{SUDE}(\mathcal{I}, \mathcal{G})$. We refer to the transformation of $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$ into $\texttt{SUDE}(\mathcal{I}, \mathcal{G})$ as *qualifying* $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$.

We believe that our two-stage approach to developing SUDE's — sketching possibilities with regression and then qualifying those possibilities — is well-suited for many tasks. For example, consider the task of controlling to avoid negative conditions. It seems reasonable to first be aware of how those negative conditions might arise due to dynamics, before expending significant effort to understand exactly what conditions will decide whether they really happen. For instance, if we identify that one way the negative conditions might occur would involve a

slow process (such as evaporation), we may wish to forego deeper analysis of that portion of the possible behavior space.

Although somewhat similar in spirit to hierarchical planning, our approach actually seems more like the opposite: we always consider all conditions of an operator, but successively consider more details about its alternative (qualitatively ambiguous) effects.

## 5.1 Impossibility

Let us first consider the problem of finding all minimal refinements of $\mathcal{I}$ from which $\mathcal{G}$ is impossible. If $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$ contains no paths, and yet $\mathcal{I}$ and $\mathcal{G}$ are incompatible, then $\mathcal{G}$ is (trivially) impossible from $\mathcal{I}$. Otherwise, impossibility basically arises when some unknowns of $\mathcal{I}$ are actually false (in a complete version of $\mathcal{I}$) and together their falsity prevents the achievability of the conditions of some operator in each path from $\mathcal{I}$ to $\mathcal{G}$ in $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$.

Recall that our regression assumes that unknowns are true as needed. For determining impossibilities, we essentially reverse that assumption, assuming unknowns are false whenever that conflicts with conditions of operators that would otherwise be applicable. The idea is to see how bad the initial conditions have to be before a cooperative Nature is no longer enough to make $\mathcal{G}$ possible.

For instance, consider three containers of water, where CAN1 and CAN2 are connected by a valved horizontal pipe, as are CAN2 and CAN3. Figure 5.1 notes the results of our technique for $\mathcal{I}$ being that CAN3 is not overflowing and $\mathcal{G}$ being that it is. Figure 5.2 depicts those results graphically. Figure 5.3 notes the results for $\mathcal{I}$ being that CAN2 is not overflowing and $\mathcal{G}$ being that it is. Note that these refinements of $\mathcal{I}$ which lead to impossibility are minimal. For example, none of these refinements refer to valves being open, since those conditions are not necessary for impossibility to arise. Further note that the second example is slightly more complex than the first. This is because in the both flows (into CAN2) must always be defeated for the second example, whereas sometimes only one flow (into CAN3) need be defeated for the first example.

Consider an arbitrary transition $S_i \rightarrow S_j$ in $\texttt{SUDE}_{\texttt{base}}(\mathcal{I}, \mathcal{G})$, where $S_c \subset S_i$ represents the conditions of the operator which causes that transition. We alternatively consider refining $\mathcal{I}$ by the negation of each unknown which is true in $S_c$, and then perform regression with $\mathcal{I}$ being

```
I:      L3 <= T3

G:      L3 >  T3

Minimal refinements of I from which G is impossible:

I1:     closed23

I2:     closed12  and  L2 <= T3

I3:     L1 <= T3  and  L2 <= T3
```

**Figure 5.1**: Impossibles for the Overflow-Can3 example

For presentation simplicity we use the following abbreviations for container i. Li refers to the water level in container i (i.e. $\texttt{level}_i$). Ti refers to the height of the top of container i (i.e. $\texttt{top}_i$). Furthermore, `open12` refers to the valve on the pipe between `CAN1` and `CAN2` being open (i.e. allowing flow), whereas `closed23` refers to the valve on the pipe between `CAN2` and `CAN3` being closed.

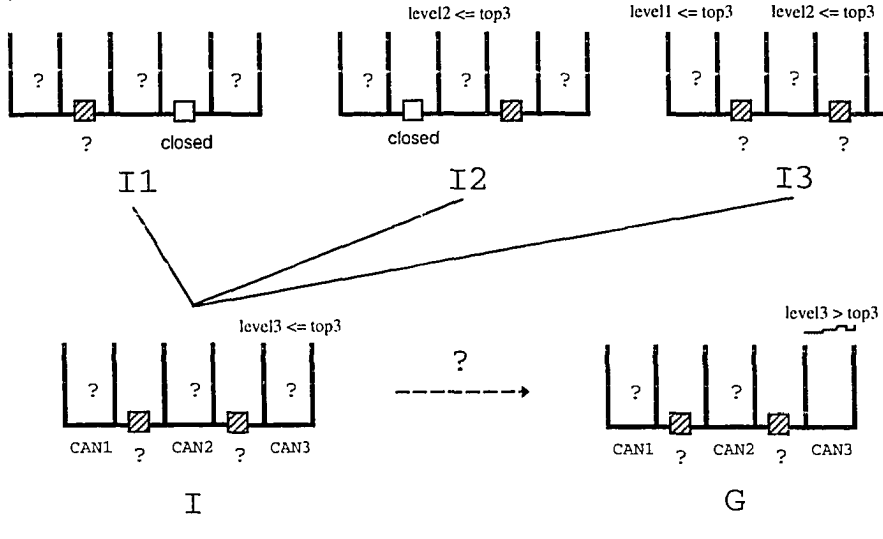


**Figure 5.2**: Graphical depiction of the Overflow-Can3 example

```
I:      L2 <= T2

G:      L2 >  T2

Minimal refinements of I from which G is impossible:

I1:     closed12  and  closed23

I2:     closed12  and  L3 <= T2

I3:     L1 <= T2  and  closed23

I4:     L1 <= T2  and  L3 <= T2
```

**Figure 5.3**: Impossibles for the Overflow-Can2 example

the newly refined version and $\mathcal{G}$ being $S_j$. However, for these regressions, we continue the bias in assuming unknowns are false, when they conflict with operator conditions. When a refined version (I') of $\mathcal{I}$ is sufficient to prevent the achievement of $S_j$ (i.e. no applicable operators), we cache I' in $S_j$'s IMPOSSIBILES set. In the end, the IMPOSSIBLES set for state $\mathcal{G}$ itself tells us which refinements of $\mathcal{I}$ cause $\mathcal{G}$ to be impossible.

Armed with the above sketch of our approach, we are now ready to examine some of the important details that arise in getting it right.

### 5.1.1 Extending REGRESS to Compute Seed Impossibility Sets

To support the computation of IMPOSSIBLES sets, we add a call to the new procedure `HANDLE-UNKNOWNS(I,G)` as the new first line of the procedure `REGRESS` that was presented in the previous chapter. However, this does not add significant overhead during the computation of $\text{SUDE}_{\text{base}}(\mathcal{I}, \mathcal{G})$. As suggested below, the cost of `HANDLE-UNKNOWNS` involves processing that is local to the given states, mainly to decide what refinements to unknowns should be considered. Lowest priority is given to entries to `FIND-PATHS`'s priority queue (PQ) (used to invoke `REGRESS`) which are made during the execution of `HANDLE-UNKNOWNS`. That ensures that the computation of $\text{SUDE}_{\text{base}}(\mathcal{I}, \mathcal{G})$ completes before incurring the costs of regressions for updating IMPOSSIBLES sets.

```
Procedure HANDLE-UNKNOWNS(I,G)

1.  Let U = UNKNOWNS-TO-NEG-FOR-I(I,G).
2.  For each F ∈ U do:
3.    REF = UNION-STATES(I,F̄).
4.    Let global RegressFlag = false.
5.    Let D = REGRESS-TO-FLIP(REF,G,F,∅).
6.    HANDLE-DISCONTINUOUS-OPS(I,G,D,F).
7.    When RegressFlag = false do:
8.      Add REF to IMPOSSIBLES(G).
9.      Add G to HAS-SEED-IMPOSSIBLES.
```

**Figure 5.4**: Procedure HANDLE-UNKNOWNS

HANDLE-UNKNOWNS first decides which unknown propositions are worthy of considering as refinements of I. For each of them, it refines I by it and checks if there are any applicable operators. If not, then that refinement is a candidate for G's IMPOSSIBLES set. Otherwise, the calls of lines 5 and 6 queue-up regressions to perform later. It will be during those regressions that further refinements of I, by the remaining unknowns, will be considered (in HANDLE-UNKNOWNS).

Note that although REF is used for the call to REGRESS-TO-FLIP, only I is used for the call to HANDLE-DISCONTINUOUS-OPS. This distinction is critically important to the overall complexity of computing impossibility sets. In fact, before we made this important realization, our attempts to compute impossibility sets lead to enormous search spaces for very simple problems. The reason is that the version of I given to those two functions will be the one queued for later regressions. For every refinement of I, we will duplicate many of the same regression paths beyond G, as regression moves towards that refinement. The only differences will involve regressions achieving the conditions of operators achieving the true versions of the falsied unknowns.

In short, we would rather queue on PQ simply $\{G\}$, rather than some $\{I', G\}$, since most of the regression work for G will be identical regardless of which I' is used. Unfortunately, we must be aware of version of I for which we are performing these regressions, otherwise we will not know what versions ultimately make $G$ impossible. Nevertheless, the key point is that the call to HANDLE-DISCONTINUOUS-OPS does *not* need this information. The reason is that its sole purpose is to ensure that we regress on any remaining unknowns (other than F) which would have to be achieved in G *before* F is. Recall that the reason we call the operators of set D

```
Procedure UNKNOWNS-TO-NEG-FOR-I(I,G)

 1.  If there is a transition from G (via OP) and OP is not a duration flip
 2.    then Let S = CONDITIONS(OP) else Let S = G.
 3.  Let REFS = ∅.
 4.  For each U ∈ Unknowns(I,S) do:
 5.    Add UNION-STATES(I,Ū) to REFS.
 6.  For each U ∈ Unknowns(I,G) do:
 7.    REF = UNION-STATES(I,Ū).
 8.    If REF is proper subset of some state in REFS then  Add REF to REFS.
 9.  Remove all proper supersets from REFS.
10.  Let FS = ∅.
11.  For each U ∈ Unknowns(I,G) do:
12.    If UNION-STATES(I,Ū) ∈ REFS then  Add F to FS.
13.  Return FS.
```

**Figure 5.5**: Procedure UNKNOWNS-TO-NEG-FOR-I

"discontinuous" is that their effects, in addition to containing F, contain the negations of other propositions in G.

### 5.1.2 Determining Which Unknowns to Falsify

HANDLE-UNKNOWNS determines which unknowns to falsify in refinements of I by calling UNKNOWNS-TO-NEG-FOR-I. By default, UNKNOWNS-TO-NEG-FOR-I first considers only those unknowns in the conditions in the regression operator leading to G. Since *Unknowns* is defined in terms of the *BaseFacts* of G, that set is typically rather minimal already.

This default restriction is very important for avoiding nonsensical refinements of I. For example, consider our domain fact that the top of every container is above the bottom of the special container named OUTSIDE. If G knows that a water level L is at the top of a container, it also knowns (via transitivity) that L above the bottom of OUTSIDE. Refining I by L being at or below the bottom of OUTSIDE would be a major mistake. First, it has no impact on the applicability on any operators in our domains, so it cannot make any behavior impossible. Second, as we noted above, excessive refining of I would greatly increase our complexities (due to duplications of regression subspaces).

```
I: level3 < middle3

G: level2 > level3
   level3 = middle3
   level2 > middle3        ;;; inferred by transitivity over other two

I-REF1: level3 <  middle3
        level2 <= level3   ;;; candidate negation to add to I
        level2 <  middle3  ;;; inferred by transitivity
        level2 <= middle3  ;;; soft ineq trivially true given hard ineq

I-REF2: level3 <  middle3
        level2 <= middle3  ;;; candidate negation to add to I
```

**Figure 5.6**: Example of selecting the proper I refinements

We choose I-REF2 over I-REF1 since I-REF2 is a subset of I-REF1 (i.e. more precise). By default, we would have expected `level3 > level3` to be the one to negate, since it is a condition for the operator underlying G (due to a flow process).

Unfortunately, that default set is not always minimal enough. Figure 5.6 gives a counter example. Whereas lines 3 – 5 compute the default set of refinements, lines 6 – 9 update that set to account for such counter examples. Lines 10 – 12 translate the chosen set of refinements into a set of single alternative propositions to refine by.

Note that lines 1 and 2 of `UNKNOWNS-TO-NEG-FOR-I` makes the simplifying assumption that all states other than refinements of $\mathcal{G}$ have exactly one effect transition recorded for them during regression. That assumption is not particularly critical to our theory for computing impossibilities, but it is interesting to note that we have observed no counter examples.

## 5.2 Falling Off Regression Paths

### 5.2.1 Propagating Impossibility

After PQ has been exhausted for the call `FIND-PATHS`$(\mathcal{I},\mathcal{G})$, $\texttt{SUDE}_{\texttt{base}}(\mathcal{I},\mathcal{G})$ and seed IMPOSSIBLES sets will have been computed. However, at this point IMPOSSIBLES($\mathcal{G}$) will yet not contain the minimal refinements of $\mathcal{G}$. Only those states for which impossibility was first de-

```
Procedure GATHER-IMPOSSIBILITY(G)

1.  Let global TODO = ∅.
2.  For each S ∈ HAS-SEED-IMPOSSIBLES do:
3.    For each N ∈ REGRESSION-NEXT-STATES(S) do:  Enqueue N onto TODO.
4.  Until TODO = ∅ do:
5.    Let S = Dequeue(TODO).
6.    UPDATE-IMPOSSIBLES(S).
```

**Figure 5.7**: Procedure GATHER-IMPOSSIBILITY

tected will have non-empty IMPOSSIBLES sets. Those states are the ones put into the set HAS-SEED-IMPOSSIBLES by `HANDLE-UNKNOWNS`

To compute the final IMPOSSIBLES($\mathcal{G}$), we must propagate IMPOSSIBILITY sets from states in HAS-SEED-IMPOSSIBLES to $\mathcal{G}$ itself. That is what a call to `GATHER-IMPOSSIBILITY(`$\mathcal{G}$`)` does. The result is that IMPOSSIBLES($\mathcal{G}$) contains all minimal refinements of $\mathcal{I}$ from which $\mathcal{G}$ is impossible.

REGRESSION-NEXT-STATES(S) denotes the set of effect states of S, as recorded during regression. As noted earlier, we have never observed these sets to contain more than one state, although we allow for the general case here. In any case, we expect that the size of these sets will typically be very close to one.

The heart of `GATHER-IMPOSSIBILITY` is performed by `UPDATE-IMPOSSIBLES`. It operates successively on each proposition P in *BaseFacts* in S, gathering all states which could achieve P on their own. It cross-products the IMPOSSIBLES sets of all such states, effectively ensuring that *all* operators which can achieve P are impossible in states produced by those cross-products. For example, if a container has two possible input flows, both flows must be unable to lead to overflowing that container in order for the overflow to be impossible.

`POSSIBLE-I?(S1,S2)` performs checks whether there is actually a path using the transitions recorded during regression for which S1 leads to S2. Since we expect REGRESSION-NEXT-STATES sets to be size one, linear time forward search from S1 to S2 is generally sufficient for this check. The reasons that this check is required in our approach is explained in the next section.

```
Procedure UPDATE-IMPOSSIBLES(S)

 1.  Let IMPOSSIBLES = IMPOSSIBLES(S).
 2.  For each P in BaseFacts(S) do:
 3.    Let NEWS = ∅.
 4.    For each T in REGRESSION-PREV-TRANS(S) with P ∈ BASE-CHANGES(T) do:
 5.      NEW = IMPOSSIBLES(PREV-STATE(T)).
 6.      Unless NEW = ∅ do:
 7.        If NEWS = ∅
 8.          else  Let NEWS = NEW.
 9.          then  Let NEWS = Cross-product(NEWS,NEW).
10.    Remove from NEWS any S_i for which POSSIBLE-I?(S_i,S) = true.
11.    Let IMPOSSIBLES = IMPOSSIBLES ∪ NEWS.
12.  Remove all proper supersets from IMPOSSIBLES.
13.  When IMPOSSIBLES ≠ IMPOSSIBLES(S)
14.    Let IMPOSSIBLES(S) = IMPOSSIBLES.
15.    For each N ∈ REGRESSION-NEXT-STATES(S) do:  Enqueue N onto TODO.
```

**Figure 5.8**: Procedure UPDATE-IMPOSSIBLES

Figure 5.9 explains previously undefined sets which are referenced here.

Line 12 ensures that IMPOSSIBLES sets remain minimal. We effectively treat IMPOSSIBLES sets like the ATMS labels.

We should point out that the cross-product computations of lines 4–9 could alternatively have been effectively been done during regression. Doing so would require refining I in HANDLE-UNKNOWNS by multiple propositions at a time, typically representing all alternative ways in which one condition from each operator achieving proposition P could be negated. However, we prefer the approach we have taken for the following reasons. First, we have been arguing that it is best to minimize, during the computation of $\mathtt{SUDE}_{\mathtt{base}}(\mathcal{I}, \mathcal{G})$, any overhead for supporting impossibility computations. Second, cross-products during regression which did not eventually lead to the identification of actual impossibilities could easily result in massively redundant search through regression space (once for each cross-producted refinement of I). In our approach, cross-products are performed only once impossibilities are identified.

REGRESSION-PREV-TRANS(S) denotes the set of transitions recorded during regression which end at state S.

BASE-CHANGES(T) denotes the proposition (and its logical equivalents) for which the regression generated transition T.

PREV-STATE(T) denotes the state at the start of transition T.

**Figure 5.9**: *Some definitions used by Procedure UPDATE-IMPOSSIBLES*

### 5.2.2 Ensuring Soundness of Impossibility Sets

We believe that the base IMPOSSIBLES sets provided by procedure `HANDLE-UNKNOWNS` are sound. That is, every state they claim cannot reach a state in set HAS-SEED-IMPOSSIBLES really cannot. This seems to be a reasonable conclusion, since we explore all applicable operators at all points and thus find some path between states if there is one. Soundness in impossibility sets seems particularly valuable. For example, we would not want to say a dangerous goal state cannot happen when it could.

Therefore, it is desirable to retain this soundness property when we propagate impossibility to $\mathcal{G}$ using `UPDATE-IMPOSSIBLES`. We can ensure soundness for each state S by adding a state $S_i$ to IMPOSSIBLES(S) only if $S_i$ is in the IMPOSSIBLES sets of every (previous) state for which REGRESSION-NEXT-STATES contains S.

Unfortunately, IMPOSSIBLES sets can be incomplete, particularly if resource limitations are imposed on regression for large problems. [1] We designed our approach with such incompleteness in mind, to allow partial IMPOSSIBILITY sets to be computed even if a large regression space has not been exhausted yet.

But the main problem is that the same state can occur at different path lengths from two states in HAS-SEED-IMPOSSIBLES. In such cases, propagation can have timing problems, where some previous states of a state might not yet known their fullest impossibility set.

`UPDATE-IMPOSSIBLES` handles such cases conservatively. It first assumes that, if any previous state of $S$ says some state $S_i$ cannot possibly reach it, then all previous states of $S$ will eventually say that $S_i$ cannot possibly reach them either. The check via `POSSIBLY-I?` invalidates that

[1] As stated, our algorithm is also incomplete for the IMPOSSIBLES of eden states (i.e. states with no previous states). However, that case is trivially handled by explicitly defining their IMPOSSIBLES sets as $\{\mathcal{I}\}$.

assumption if there is actually some path of recorded transitions to $S$, from some refinement of $\mathcal{I}$ compatible with $S_i$.

Nevertheless, it is not clear if such conservative measures are actually necessary. Careful propagation, from the states farest from $\mathcal{G}$ first, might be able avoid the timing problems mentioned above. However, it is unclear whether some impossibility sets could be more incomplete than others, particularly when regression space is not exhausted. If so, our conservative approach could help propagate impossibilities that would otherwise be dropped. Given that impossibility sets tend to be rather small anyway, avoiding such loses could be quite significant.

Furthermore, the observed and expected efficiency of `POSSIBLE-I?` checks makes the practical benefits of a more sophisticated propagation scheme unclear. The overriding cost in our approach is the generation of the regression space itself.

### 5.2.3 The Significance of Impossibility Sets

Reasoning about impossibility seems to more relevant to QR than general planning. In many planning problems, the available operators are reversible and plentiful. For example, there are relatively few planning problems for which there is no solution. Thus, the emphasis in most planning work is finding optimal/good plans, not determining the conditions under which no successful plans are possible.

For example, consider the Schopper's blocks world, which contains a mischievous child who likes to knock down our partial towers [48]. In such a world, it is typically just a matter of time before we will succeed in building any tower we want. We do not have to anticipate the child's actions, we just have to be patient and persistent. Only if the child always acted by special rules, such as one compelling her to immediately knock down a tower of size N, would we have some limits on our goals.

In contrast, QR domains are full of constraints and many of Nature's operators are not reversible. Therefore, it would appear that our work on identifying impossibility conditions might have more utility than planning work might suggest. Even in planning work focusing on resource limitations, the emphasis is not on identifying what limitations would be fatal, but rather how to optimize in spite of them.

# Chapter 6

# Conclusions

We conclude by first summarizing the contributions and limitations of our work and then discussing some related and future work.

## 6.1 Summary of Contributions

We can summarize our major contributions according to our impact on three key issues:

1. what partial representations of across-time behavior are adequate for QR?
2. how should goal states be exploited for efficient QR?
3. how should causality-based QR deal with discontinuous change?

To address the first two issues, we have introduced the SUDE (SUfficient Discriminatory Envisionment) representation, along with goal-directed techniques for computing SUDE's. To address the third issue, we have developed a method for integrating theories of minimality-based change and continuity-based change that appears to overcome the individual limitations of each. In the following sections, we summarize some of the more interesting results of our work.

### 6.1.1 Towards Sufficiency Conditions for QR

Whereas numeric simulation is a gold standard for the soundness and completeness of QR, total-state qualitative simulation serves a similar role in defining sufficient complexities for QR. We

offer our SUDE representation and algorithms as a potential new standard for the complexity sufficient to satisfy the actual requirements of QR tasks. SUDE's represent all alternative conditions across time which are sufficient for explaining when initial state $\mathcal{I}$ leading to goal state $\mathcal{G}$ is possible, impossible, or inevitable. We have argued that this level of discrimination appears to be particularly relevant for many QR tasks.

### 6.1.2 Turning the Weakness Of QP Constraints Into A Strength

It has become clear that there are strong limitations on the predictive power of QP constraints [50]. Our work provides a way to turn that weakness into a strength, since weaker QP constraints lead to smaller, easier to compute, SUDE's.

### 6.1.3 Integrating Theories of Minimal and Continuous Change

Our combination of continuity-based and minimality-based change addresses the limitations of each. Specifically, we enforce continuity on the conditions of the dynamic operator underlying a state transition, while enforcing minimal change in the other state propositions. We identified one key exception: that minimal change cannot be enforced on derivatives of quantities whose underlying influence sets change in ways that make all changes plausible. We have argued in detail that well-known counter-intuitive results of minimality-based change arise from ignorance of the causal constraints provided by QR models. Yet, our restricted use of minimal change accounts for discontinuities due to sudden actions or modelling simplifications. To facilitate this integration, we have argued that our single base change assumption is reasonable, based on the notion that true achievability of goal states does not depend on the occurrence of truly coincidental changes.

### 6.1.4 Integrating QR With Planning

STRIPS-like planning operators precompile all of their knowledge about physics into their discrete effects. Such approaches are inadequate when the condition sets under which effects are unambiguous are either intractably large or even undefinable. Typical QR tasks suffer from both of those problems, since qualitative modelling strives both for broad coverage via first principles and for graceful degradation of inference under uncertainty.

By using SUDE's, we are able to integrate QR with planning without importing the traditional complexity of QR, as has been the case in other approaches [26]. To support this, we have presented means for automating the formulation of dynamic operators that model the potentially ambiguous behavior of Nature between actions.

By avoiding precompilation of physics into action operators for the most part, actions themselves become mostly simple flips in the status of process preconditions. Thus, in our approach goal conflicts in planning are best viewed as discontinuities in the conditions of dynamics operators.

Furthermore, our approach supports the automatic generation of STRIPS action operators in those rarer cases where dynamics are unambiguous. Specifically, our impossibility sets for a goal consisting of proposition $P$ correspond to the conditions of operators which necessarily *immediately* achieve $\overline{P}$. Similarly, our inevitability sets for goal $P$ correspond to the conditions of operators which necessarily *eventually* achieve $P$. In both cases, intermediate dynamics could be ignored during planning, as long as no conflicting intermediate actions are introduced.

## 6.2 Challenges to the QR Community

Our work presents several challenges to the QR community. We have challenged some assumptions underlying existing QR work and have claimed that some relative simple techniques seem to adequately satisfy the needs of actual tasks employing QR. Unfortunately, most of these claims have elluded both empirical and theoretical proof. However, we would not be terribly surprised or disappointed if some exceptions to our claims do exist. It is our hope that our claims will facilitate the search for special counter-examples which would illuminate when the complexities of other approaches are truly required. Toward this goal, we briefly restate some of our claims below.

### 6.2.1 The Suitability of Envisionments

One of the original motivations of generating envisionments was to determine the functions of systems by classifying their gross behavior, such as oscillatory systems, closed thermal systems, etc. However, for goal-directed QR tasks, it seems be more appropriate to concentrate on

possibility paths and impossibility/inevitability sets. For what goal-directed tasks might the articulation of all possible internal transitions of a system be useful or even critical?

### 6.2.2 The Adequacy of Bottom-Up QR

Our techniques are founded on the belief that QP constraints are generally so weak that it is far better to work up from individual constraints than to work down from complete states.

We define the base assumptions in the descriptions of states as follows:

1. the base assumptions of $\mathcal{I}$ and $\mathcal{G}$ are as defined by the user,
2. the base assumptions of other states are defined by the result of regressing specific state by a specific operator.

We further consider a state assumption P to be *relevant* to a state description S, warranting the replacement of S by their union, in following cases:

1. P is in the assumption closure of S,
2. P necessarily persists from $\mathcal{I}$,
3. P necessarily persists from all previous states of S in $\texttt{SUDE}(\mathcal{I}, \mathcal{G})$,
4. P necessarily persists from some earlier state of S in $\texttt{SUDE}(\mathcal{I}, \mathcal{G})$ and is inconsistent with S (i.e. pruning an unsound transition). [1]

Thus, we avoid tracking propositions forward through existing $\texttt{SUDE}(\mathcal{I}, \mathcal{G})$ structure except under some cases of necessary persistence. Recall that these simplifications are designed to avoid case-splitting of S unless necessary to achieve global soundness. Under what conditions might these simplifications still lead to global unsoundness? What other criteria might make forward tracking useful?

One of the underlying justifications for these simplifications is that knowing assumption closures seems sufficient for propagating important local constraints between adjacent states. Under what conditions might parallel forward tracking of explicit case-splits still be necessary for global soundness?

[1] In general, this inconsistency may involve multiple P's all satisfying this condition for S.

## 6.3 Summary of Limitations

Our approach suffers from some notable limitations, mainly due to its reliance on causal qualitative models.

### 6.3.1 Reliance on Causal Models

Our work fundamentally depends on the availability of the influence structures provided by causal qualitative models, such as those provided by QPT models. Our methods for computing state transitions employ influence change analysis and our methods for regression ultimately rely on dynamic operators formulated from sufficient chunks of influence structure.

Being so closely aligned with influence structure has its advantages. In particular, influence structure often provides an especially appropriate substrate for meaningful explanations. Nevertheless, it is important to realize that some physical phenomena appear less conducive to causal modelling. For example, the constraints imposed on the movements of mechanical linkages seem much easier to model as global constraints than as causal influences [37].

The lack of influence structure poses no particular problem for QS, since QS can easily be viewed as a constraint-satisfaction problem. Indeed, the work of [37] was readily implemented using the qualitative simulator QPE. Moreover, the theory underlying QSIM makes no reference to influence structure. The importance of influence structure becomes prevalent only when one needs to focus on sufficient conditions under which particular changes may or may not occur, as we do.

### 6.3.2 Qualitative Constraints Are Very Weak

The general import of our work is limited by the ability of QP constraints to model significant real-world phenomena.

For example, goals which are qualitatively impossible or inevitable are relatively rare in most domains. Most examples of impossibility or inevitability that we can imagine involve the status of preconditions, not quantity conditions. Since preconditions are generally due to actions and quantity conditions are due to dynamics, most examples tend to identify static conditions for impossibility/inevitability, as opposed to dynamic ones. Dynamic impossibility, for example,

seems to fundamentally involve either the disappearance of any influence in the right direction or the occurrence of dead-locks between the conditions of conjunctively necessarily influences.

These limitations are not specific to our approach — the same inferential weakness exists for QS. It is simply that our approach makes these limitations more apparent, since our goal is making interesting inferences using minimal conditions. The excessive complexity of QS has, in some sense, clouded the true elegance — and limitations — of qualitative constraints.

## 6.4 Related Work

We have mentioned much of the related work in earlier chapters. We discuss other related work below.

### 6.4.1 Temporal Reasoning

Hanks addressed the problem of predicting possible outcomes of linear plans in uncertain worlds by projecting only relevant conditional effects of actions [33]. He initially *bundles* the alternative conditional effects of an action into one *outcome set*. He then unbundles outcome sets into smaller outcome sets which each differ in their answer to task queries. Such unbundlings produce internal queries (to decide when action conditions might occur), invoking new rounds of unbundlings. Eventually, a tree of relevant outcomes (rooted at the original uncertain world) is articulated.

Hanks suggests in [32] that his approach might address the problem of irrelevant branching in QR. However, upon closer examination it appears that his approach is likely to make the complexity of QR increase, not decrease. Assume that we adopt his action operator formulation for Nature's (dynamic) actions. Figure 6.1 illustrates a formulation of a flow process into his operator representation. Due to qualitative ambiguity, the conditions of many of Nature's actions would contain Hanks's `chance` predicate, except in highly restricted cases, such as this example, where no other possible changes exist. Note that formulations for which multiple flows could be active at the same time would lead to combinatorial explosions in the conditions.

Our influence-based approach avoids the need to distinguish whether the level of can2 reaches top1 or top2 first. In Hanks's approach, such distinctions must necessarily be made within the action schema, to account for all possible eventual outcomes. Given the weak na-

ture of the qualitative constraints, this approach is clearly unreasonable. It would result in combinatorial explosions in the outcome sets with little added predictive power.

Williams argues in [54] that bottom-up reasoning with temporal constraints can address the problem of excessive forward branching in QS. He presents temporal constraint propagation techniques which consider relations between events only when their interaction can result in the change of other quantities. Although this approach has the intuitive appeal of only relating state propositions when their relation has causal significance, it does not really address the issue of branching per se. In fact, Williams says nothing about how to introduce case-splits as needed to explore relevant branching. The sole QR example he presents, a simple spring-block oscillator, involves no branching. Taken at face value, Williams' work appears to simply maintain the intersection of all predictions that are possible from a partial initial state. Furthermore, Williams offers no insight into how his approach might be modified to take advantage of goals.

### 6.4.2 Planning

Hogge [35] [34] explored the idea of compiling QPT models into temporal constraints and operators and then applying a temporal planner [1] to plan in a dynamic world. Although there is some merit to that idea, Hogge's implementation seemed to get bogged down in a sea of temporal constraints. Given that qualitative constraints are often rather weak, the need to process many temporal constraints to make relatively simple qualitative inferences has questionable utility. This seems particularly so given that Hogge's approach had to make many simplifications (such as assuming Nature always cooperates as needed) for tractability. In contrast, our SUDE-based approach works directly with states which are sufficient for showing possibility paths. Hogge's work did not demonstrate that its indirect route (using temporal constraints) provides any computational advantages over a more direct state-based approach, particularly when most changes are dynamic and ambiguous.

### 6.4.3 Nonmonotonic Reasoning

Crawford and Etherington have presented a nice summary of how QR's clean separation of model selection and model simulation help QR avoid many of the counter-intuitive results in the nonmonotonic reasoning (NMR) community [6]. By committing to closed-world assumptions

```
(action (flow mass ?substance ?can1 ?can2 ?pipe)
 (if (not (and (aligned ?pipe)
               (> (level ?can1) (level ?can2))
               (> (mass (C-S ?substance LIQUID ?can1)) 0)))
     (outcomes INFEASIBLE)
     (if (chance 0.5) ;; flow stops before any change occurs due to flow:
         (outcomes (= (level ?can1) (level ?can2)))
         (cond
            ((< (level ?can2) (height (top ?can2)))
             (cond
              ((< (level ?can2) (height (top ?can1)))
               (if (chance 0.5)
                   (outcomes (= (level ?can2) (height (top ?can1))))
                   (outcomes (= (level ?can2) (height (top ?can2))))))
              ((= (level ?can2) (height (top ?can1)))
               (outcomes (> (level ?can2) (height (top ?can1)))))
              ((> (level ?can2) (height (top ?can1)))
               (outcomes (= (level ?can2) (height (top ?can2)))))))
            ((= (level ?can2) (height (top ?can2)))
             (cond
              ((< (level ?can2) (height (top ?can1)))
               (outcomes (> (level ?can2) (height (top ?can2)))))
              ((= (level ?can2) (height (top ?can1)))
               (outcomes (> (level ?can2) (height (top ?can1))))
                         (> (level ?can2) (height (top ?can2))))
              ((> (level ?can2) (height (top ?can1)))
               (outcomes (> (level ?can2) (height (top ?can2)))))))
            ((> (level ?can2) (height (top ?can2)))
             (cond
              ((< (level ?can2) (height (top ?can1)))
               (outcomes (= (level ?can2) (height (top ?can1)))))
              ((= (level ?can2) (height (top ?can1)))
               (outcomes (> (level ?can2) (height (top ?can1)))))
              ((> (level ?can2) (height (top ?can1)))
               ;; flow is active forever (not really consistent)
               (outcomes NULL)))))))
```

**Figure 6.1**: Hanks-like operator for a flow process

This operator assumes that this flow is the only influence on quantity
`(mass (C-S ?substance LIQUID ?can1))`.

before reasoning about possible behaviors over time, QR avoids the trap of trying to change simulation to avoid state inconsistencies.

For this reason, we accept the restrictions that come from making strong closed-world assumptions on influence structure. As Rayner showed in [45], failing to make closed-world assumptions can lead to counter-intuitive results when discontinuities are allowed. He showed that Sandewall's [46] nonmonotonic formulation for minimalizing discontinuities fails without explicit axioms conditioning when discontinuities should not be allowed. Of course, by requiring explicit conditions of when discontinuities are allowed, much of the potential import is lost.

### 6.4.4 Modal Logics

Modal logics are motivated by the desire to conveniently and concisely distinguish between necessary truth and possible truth. Since such distinctions are relevant to qualitative reasoning about goal state achievability, this section discusses the relevance of modal logic to our work. We base this discussion on the conventions used in [49].

A *Kripke structure* is modal logic's analog of extensional logic's models/interpretations. It consists of *possible worlds* and an *accessibility relation* among them. In QR, possible worlds correspond to complete states and accessibility relations are based on state transitions. The various modal logic systems differ on their properties for the accessibility relation. For reasoning about goal achievability, the accessibility relation should be transitive and reflexive, but not symmetric (i.e. the so called $S_4$-system). Thus, $\mathcal{G}$ being accessible from $\mathcal{I}$ does not mean $\mathcal{I}$ is accessible from $\mathcal{G}$.

Common modal operators include:

- agent *knows* logical formula $\varphi$ (denoted $\mathtt{K}\varphi$),
- agent *believes* $\varphi$ (denoted $\mathtt{B}\varphi$),
- $\varphi$ *necessarily* holds in the future (denoted $\Box\varphi$),
- $\varphi$ *possibly* holds in the future (denoted $\Diamond\varphi$), and
- $\varphi$ holds in the *next* state (i.e. for discrete time points) (denoted $\bigcirc\varphi$).

Model operators $\Box$ and $\Diamond$ correspond to our notions of goal inevitability and possibility, respectively (goal impossibility corresponds simply to $\Box$ for the negation of the goal). The

semantics of a modal operator (for a particular possible world) typically depends on whether it is based on *all* accessible possible worlds or just *some*. In our case, $\Box$ is based on all and $\Diamond$ is based on some.

Given the existence of very formal, concise modal logics for $\Box$ and $\Diamond$, one might question the significance and role of our work with respective to them. There are a few important responses to that question, which we examine in turn below.

First, one must be sure to distinguish between proof and model theory. Modal logic formalisms crisply define the semantics of the operators, but they do not commit to particular algorithms for computing them. Nor do modal systems commit to particular ontologies for defining sets of model tokens and functions for which such algorithms are efficient and adequate for given tasks. Such concerns are exactly what much of qualitative physics primarily addresses and what our work impacts.

Second, even though they do not compute the Kripke structure, modal logicals must at least define it (to define the semantics). Defining the accessibility relation for QR inherently requires a formalism of qualitative temporal reasoning. Thus, the trade-off fundamentally boils down to whether one defines qualitative temporal reasoning declaratively or procedurally. The experience of QR work in general, and our experience in properly integrating minimal-based and causal-based change, argues that the procedural approach is preferable, both in terms of computational complexity and in terms of soundness/completeness.

Third, it is well-known that anything representable in any modal logic is also representable in some first-order logic. We argue that most potential advantages that a modal logic formalism might have over our envisionment-based formalism would be due more to the advantages of first-order formalism over ours, for particular tasks. For example, propositional logic allows one to represent a disjunctive-normal formula as one (DNF) formula, whereas our state-based approach requires case-splitting into multiple states, each corresponding to one conjunctive-normal formula.

It is unclear exactly when this difference gives propositional logic a real advantage over our approach for actual tasks. Reasoning about the subtle ways in which physical components may interact over time requires some ability to explicitly case-split the context of each. In any case, it is especially unclear whether the use of modal logics could ever reduce the case-splitting required for QR using a first-order logic.

In summary, our focus on goal-directed reasoning with partial states and on leveraging the inherent ambiguity of QR might better be viewed as potentially reducing the computational complexity of QR-like reasoning under modal logics than in viewing the employment of modal logics per se as potentially improving the process of QR itself. The specific commitments which define QR and characterize its efficient execution come from qualitative physics work, not from generic modal logics about possibility and time. To claim otherwise is to claim that work in modal logics subsumes work in qualitative physics.

## 6.5 Future Work

Based on our experience from this work, we believe that the following issues are especially worthy of further exploration.

### 6.5.1 Formal Theories of SUDE's

We have not been able to develop crisp declarative definitions of the representational properties of SUDE's. We have largely defined SUDE's procedurally — in terms of how regression and qualification compute them. However, it is possible that such formal definitions exist.

Furthermore, we do not have solid soundness or completeness results for SUDE's. For example, SUDE's would be more globally unsound than AE's if our assumption that tracking only necessarily persisting facts is adequate turns out to be false.

### 6.5.2 Comparative Analysis and Relative Probability

There is much intuition behind the idea that qualitative models are more useful for comparative analysis than prediction. The underlying reason seems to be that even when ambiguity makes precise prediction impossible, precise comparative analysis is possible. The very notion of a qualitative influence is that, all else being equal, adding an influence (or increasing an existing one) will increase the influenced quantity. For example, even though we often cannot say qualitatively whether a container might overflow, we can usually say that turning on a new pump into that container will increase the chance that the container will overflow.

For many tasks, it may well be that the relative probability of behaviors arising from perturbations is more interesting than our distinctions of possibility, impossibility, and inevitability.

One might also be able to achieve stronger results in terms of relative probability than in terms of goal discrimination per se. Whereas it is easy to imagine SUDE's for which $\mathcal{G}$ is never (or seldom) impossible or inevitable, it seems much harder to imagine examples where particular perturbations do not affect the probability of whether $\mathcal{G}$ eventually occurs.

### 6.5.3 Better Control Strategies

We stayed within a phase-space trajectory view of behavior in this work because it was easier to compare our results with those of QR in general. However, it is well worth exploring the possibility of using partially-ordered behavior trajectories to avoid many occurrence branching problems.

Our work emphasized backward regression over forward projection because we believe that backward reasoning is more likely to quickly identify the dynamic discriminatory information available in qualitative behavior space. Nevertheless, surely there are cases where the reverse is true. However, we suspect that most of these cases will involve static spatial constraints, such as the path of a flying ball being most constrained early in flight. For non-spatial constraints, the conditions discriminating whether a goal is reachable generally seem to occur late, if at all.

# References


[1] James F. Allen and Johannes A. Koomen. Planning using a temporal world model. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 741–747, August 1983.

[2] Mark Boddy, Bob Schrag, and Jim Carciofini. Extended abstract: Managing disjunction for practical temporal reasoning. In *Working Notes of the AAAI Spring Symposium on Practical Approaches to Scheduling and Planning*, pages 20–24, March 1992.

[3] Seng-Cho Timothy Chou. *Reasoning With Model-Based Belief Revision Semantics: Theory and Implementation.* PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1992.

[4] Enrico W. Coiera. Qualitative superposition. *Artificial Intelligence*, 56:171–196, 1992.

[5] John Collins and Dennis DeCoste. CATMS: An ATMS which avoids label explosions. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, July 1991.

[6] James M. Crawford and David W. Etherington. Formalizing reasoning about change: A qualitative reasoning approach (preliminary report). In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 577–583, 1992.

[7] Wilkins David E. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann, 1988.

[8] Johan de Kleer. Causal and teleological reasoning in circuit recognition. Technical Report 529, Massachusetts Institute of Technology, Artificial Intelligence Lab, Cambridge, September 1979.

[9] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2):127–162, March 1986.

[10] Johan de Kleer. Extending the ATMS. *Artificial Intelligence*, 28(2):163–196, March 1986.

[11] Johan de Kleer. Exploiting locality in a tms. In *Proceedings of the Nineth National Conference on Artificial Intelligence*, pages 264–271, August 1990.

[12] Johan de Kleer and John Seely Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[13] Thomas Dean and Greg Siegle. An approach to reasoning about continuous change for applications in planning. In *Proceedings of the Nineth National Conference on Artificial Intelligence*, pages 132–137, 1990.

[14] Thomas L. Dean and Drew V. McDermott. Temporal data base management. *Artificial Intelligence*, 32, 1987.

[15] Thomas L. Dean and Michael P. Wellman. *Planning and Control.* Morgan Kaufmann, 1991.

[16] Dennis DeCoste. Dynamic across-time measurement interpretation. *Artificial Intelligence*, 51:273–341, 1991.

[17] Alvaro del Val. Computing knowledge base updates. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, 1992.

[18] Alvaro del Val. Syntactic characterizations of belief change operators. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1993.

[19] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[20] Daniel Dvorak and Benjamin Kuipers. Model-based monitoring of dynamic systems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1238–1243, August 1989.

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

[22] R. Fikes and Nils Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[23] Kenneth D. Forbus. A study of qualitative and geometric knowledge in reasoning about motion. Technical Report TR-615, Massachusetts Institute of Technology, Artificial Intelligence Lab, Cambridge, 1981.

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

[25] Kenneth D. Forbus. Qualitative process theory. Technical Report TR-789, Massachusetts Institute of Technology, Artificial Intelligence Lab, Cambridge, 1984.

[26] Kenneth D. Forbus. Introducing actions into qualitative simulation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1273–1278, August 1989.

[27] Kenneth D. Forbus. The qualitative process engine. In Daniel S. Weld and Johan de Kleer, editors, *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).

[28] Kenneth D. Forbus and Brian Falkenhainer. Self-explanatory simulations: An integration of qualitative and quantitative knowledge. In *Proceedings of the Nineth National Conference on Artificial Intelligence*, pages 380–387, August 1990.

[29] Pierre Fouche and Benjamin Kuipers. Abstracting irrelevant distinctions in qualitative envisionments. In *Proceedings of the Fifth Workshop on Qualitative Physics*, Austin, Texas, May 1991.

[30] Pierre Fouche and Benjamin Kuipers. An assessment of current qualitative simulation techniques. In Boi Faltings and Peter Struss, editors, *Recent Advances in Qualitative Physics*, pages 263–278. The MIT Press, 1992.

[31] Matthew L. Ginsberg and David E. Smith. Reasoning about action i: A possible worlds approach. In Matthew L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning.* Morgan Kaufmann, 1987.

[32] Steve Hanks. Practical temporal projection. In *Proceedings of the Nineth National Conference on Artificial Intelligence*, pages 158–163, August 1990.

[33] Steven Hanks. *Projecting Plans for Uncertain Worlds.* PhD thesis, Department of Computer Science, Yale University, January 1990. (Technical Report 756).

[34] John Hogge. The compilation of planning operators from qualitative process theory models. Master's thesis, University of Illinois at Urbana-Champaign, September 1987. (Technical Report UIUCDCS-R-87-1368).

[35] John Hogge. Compiling plan operators from domains expressed in qualitative process theory. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, July 1987.

[36] Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge database and revising it. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, 1991.

[37] Hyun-Kyung Kim. *Qualitative Kinematic and Dynamic Mechanics.* PhD thesis, University of Illinois at Urbana-Champaign, May 1992.

[38] Richard E. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33:65–88, 1987.

[39] Benjamin Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

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

[41] Benjamin Kuipers and Charles Chiu. Taming intractible branching in qualitative simulation. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1079–1085, August 1987.

[42] Karen L. Myers and David E. Smith. The persistence of derived information. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 496–500, 1988.

[43] Edwin Pednault. Formulating multiagent, dynamic-world problems in the classical planning framework. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 675–710. Morgan Kaufmann, 1990.

[44] Mark A. Peot and David E. Smith. Conditional nonlinear planning. In *First International Conference on AI Planning Systems*, pages 189–197, 1992.

[45] Manny Rayner. On the applicability of nonmonotonic logic to formal reasoning in continuous time. *Artificial Intelligence*, 49:345–360, 1991.

[46] Erik Sandewall. Combining logic and differential equations for describing real-world systems. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 412–420, 1989.

[47] Erik Sandewall. Filter preferential entailment for the logic of action in almost continuous worlds. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 894–899, 1989.

[48] Marcel Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1039–1046, August 1987.

[49] Yoav Shoham. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence.* The MIT Press, 1988.

[50] Peter Struss. Problems of interval-based qualitative reasoning. In Daniel S. Weld and Johan de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 288–305. Morgan Kaufmann, 1990.

[51] Dan Weld. *Theories of Comparative Analysis.* PhD thesis, Massachusetts Institute of Technology, May 1988.

[52] Daniel S. Weld and Johan de Kleer, editors. *Readings in Qualitative Reasoning about Physical Systems.* Morgan Kaufmann, 1990.

[53] Brian Williams. Qualitative analysis of MOS circuits. Master's thesis, Massachusetts Institute of Technology, July 1984. (MIT AI Lab Technical Report 767).

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

[55] Brian C Williams. MINIMA: A symbolic approach to qualitative reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 264–269, 1988.

[56] Marianne Winslett. Reasoning about action using a possible models approach. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 89–93, 1988.

# Vita

Dennis DeCoste was born on April 22, 1964 in Chicago, Illinois. He received a Bachelor of Science degree in Computer Science from the University of Illinois at Urbana-Champaign (UIUC) in May of 1986. In the Fall of 1986 he entered the graduate program in Computer Science at UIUC and began research under the direction of Professor Ken Forbus in the Qualitative Research Group at UIUC. His initial graduate work lead to the development of the DATMI system, which dynamically maintained the space of all qualitative behaviors of a physical system consistent with an incomplete set of observed data. For that work he received the Master of Science in Computer Science degree from UIUC in December of 1989. DATMI was reported in 1991 in the journal *Artificial Intelligence.*

Motivated by the limitations of DATMI, he decided to explore the fundamental computational and representational problems of qualitative simulation. This lead to development of a goal-directed form of qualitative simulation in which the detail of qualitative states varies over time, to reflect the dynamic, contextual nature of relevancy. For this work he received the Ph.D. in Computer Science from UIUC in May of 1994.

He is currently a member of the technical staff at the NASA Jet Propulsion Laboratory in Pasadena, California. The current focus of his research at JPL is in modelled-based reasoning, particularly as it applies to the tasks of on-line mission monitoring and off-line preparation for mission operations.