MIT/LCS/TR-329

QUALITATIVE MATHEMATICAL REASONING

Elisha Sacks

This blank page was inserted to preserve pagination.

Qualitative Mathematical Reasoning

by

Elisha Sacks

November 1984

©Massachusetts Institute of Technology 1984

This investigation was supported (in part) by National Institutes of Health Grant No. 5 R24 RR01320 from the Division of Research Resources, by Grant No. 1 R01 HL33041 from the National Heart, Lung, and Blood Institute, and by Grant No. 1 P01 LM 03374 from the National Library of Medicine.

Laboratory for Computer Science Massachusetts Institute of Technology

Cambridge, Massachusetts 02139

Qualitative Mathematical Reasoning

by
Elisha Sacks

Abstract

Qualitative analysis is the study of abstract causal reasoning. It explores the mechanisms whereby humans analyze complex systems abstractly, while ignoring unimportant and unknown low-level details. Previous research has focused on *qualitative simulation* techniques, analogous to numerical simulation, that use local information about a system to predict its short-term behavior. This thesis presents a new, calculus based, type of qualitative analysis, called *qualitative mathematical reasoning.* It derives functional descriptions of systems and uses them to predict global behavior.

qualitative mathematical reasoning analyzes a system in two stages; first, it derive a mathematical description of each important parameter, then it determines the qualitative behavior of these descriptions. The first stage utilizes the theory of differential equations, and the second, real analysis. The current version of QMR solves most of the examples from other qualitative analysis papers. More importantly, it provides a clean precise foundation for future systems.

Thesis Advisor: Peter Szolovits
Title: Associate Professor of Computer Science

Table of Contents

1 Introduction 4

2 Qualitative Mathematics 7
- 2.1 Symbolic Arithmetic . . . 7
- 2.2 Data Structures . . . 8
- 2.3 Algorithms . . . 10
  - 2.3.1 Instantiation Algorithms . . . 10
  - 2.3.2 Combination Algorithms . . . 11
  - 2.3.3 Description Algorithms . . . 17
- 2.4 Periodic Functions . . . 19
- 2.5 Summary of QM . . . 25

3 Qualitative Reasoning 26
- 3.1 The QMR Algorithm . . . 26
- 3.2 QMR Examples . . . 26
  - 3.2.1 The Falling Ball . . . 26
  - 3.2.2 Heat Flow . . . 30
  - 3.2.3 Oscillation . . . 32
  - 3.2.4 Damped Oscillation . . . 34

4 Comparison with Related Work 36
- 4.1 Overview of QP . . . 36
  - 4.1.1 Data Structures . . . 36
  - 4.1.2 Algorithms . . . 38
  - 4.1.3 Current Systems . . . 39
- 4.2 Problems with QP . . . 40
- 4.3 Advantages of QM . . . 42

5 Summary and Future Work 43

References 44

A Instantiation Functions 47

B Description Queries 48

List of Figures

2.1 Arithmetic in QM 8
2.2 A Sample ANALYZE run 14
2.3 Sample Description Queries 19
2.4 A Mathematical Description 20
2.5 A Periodic Function 20
2.6 Restriction of a Par-fun-int 22

3.1 Two Sample Nodes 27
3.2 The Ball's Flight 28
3.3 The Ball Network 28
3.4 The Heat Flow Network 30
3.5 The Spring Network 32
3.6 The Damped Spring Network 34

4.1 Value Space for Water 37

List of Tables

2.1 The Function Interval 9
2.2 The ball's fun-ints 9
2.3 Linear Substitution 12
2.4 Linear Composition 12
2.5 The ANALYZE algorithm 13
2.6 Functional Composition 15
2.7 The FIND-DIR algorithm 16
2.8 Parameterized fun-ints 21

3.1 Node Types 27
3.2 Parameterized *fun-ints* 32

1. Introduction

Qualitative analysis is the study of abstract causal reasoning. It explores the mechanisms whereby humans analyze causal systems abstractly, while ignoring unimportant and unknown low-level details. This skill plays a major role in intelligent behavior, ranging from common sense reasoning about everyday events to expert understanding of complex machines. It enables children to conclude that hot stoves burn fingers, and doctors to diagnose diseases. This thesis presents a new, calculus based, type of qualitative analysis, called *qualitative mathematical reasoning*. It argues that QMR provides a clean precise foundation for both expert and common sense causal reasoning programs.

Qualitative analysis divides naturally into two stages, representing problems in a precise formalism and solving them. A *network* model describes a wide class of problems flexibly and concisely; interesting parameters are represented by functions of time and dependencies between parameters by functional equations. Hence, many researchers translate problems into this model by hand and concentrate on the second stage, deducing a network's behavior from an abstract description of its initial state and of the dependencies between its components. I, too, adopt this model and concentrate on qualitative analysis of given networks.

A typical common sense problem, "What happens to a ball when it is thrown straight up?", demonstrates qualitative reasoning in the network model. Although the exact relation between height, velocity and acceleration may be unknown, qualitative analysis can infer that the ball will reach a peak height and fall back to earth. One representation that allows that inference has the following components and initial states

1. $h(t)$—the ball's height at time t with $h(0) = 0$,
2. $v(t)$—its velocity with $v(0) > 0$ and
3. $g(t)$—its acceleration with $g(0) < 0$.

Dependencies between components can be stated as

1. v is h's derivative
2. g is v's derivative
3. g is constant.

The *qualitative reasoning* used to analyze h might be

> Initially, h increases by dependency 1 since v is positive, also v decreases at a constant rate by dependencies 2 and 3 since g is negative. When v reaches zero, h attains a maximum. From then on, v becomes more and more negative, so h decreases until it reaches zero again.

or something along those lines. It would utilize theorems about derivatives such as "a function with a positive derivative increases" and continuity theorems such as "if f is continuous and $f(0)$ is positive then f will remain positive in a neighborhood of 0." The main topics of my thesis are an algorithmic approach to qualitative reasoning and the programs that implement this approach.

As the ball example illustrates, qualitative reasoning presupposes a well-behaved network in which all functions are piecewise differentiable. This assumption—corresponding to the intuition that physical phenomena vary smoothly, with the possible exception of a few irregularities—allows the calculus of piecewise continuous functions on $\Re^*$ to be used for network analysis. For example, the aforementioned ball's height function fits the quadratic model

$$h(t) = v_0 t + \frac{g}{2}t^2 \text{ with } \begin{cases} v_0 > 0 \\ g < 0 \end{cases} \tag{1.1}$$

where g is the gravitational constant and v_0 the ball's initial height. Given this representation, its behavior can be determined by inspecting the derivative's sign. Qualitative reasoning consists, no doubt, of many different special-purpose methods, some known to most people and others invented by individuals for their own use. However, I contend that qualitative reasoning programs should follow the strategy that has been used successfully by *experts*. They should utilize calculus methods whenever feasible and only fall back on other, more general, techniques as a last resort. This strategy forms the basis for my work and distinguishes my program from other qualitative reasoners. It unifies the solutions to more and less precisely specified problems in a single formalism, one that permits specialized mathematical reasoning about the former alongside general reasoning about the latter.

This thesis describes QMR, a qualitative reasoning program that uses calculus and other mathematical techniques to analyze networks. QMR derives closed-form arithmetic expressions for each function in a network and analyzes the solutions. Chapter 2 describes QM, a system that analyzes piecewise continuous functions from $\Re$ to $\Re^*$, such as equation 1.1. Chapter 3 presents QR, a simple qualitative reasoner that finds closed-form solutions to networks and uses QM to derive their behavior. The two steps are discussed in reverse order since QM is well developed and fully implemented, whereas the network analysis algorithm is rudimentary. Nevertheless, as several examples show, QR can solve many interesting problems. Chapter 4 reviews related work and compares it with QMR; it argues that QM's

calculus based paradigm is more promising than the naive approach taken by other qualitative reasoners. Chapter 5 summarizes QMR's strengths and shortcomings and outlines plans for future work; the main thrust will be to produce a powerful useful qualitative reasoner by extending and generalizing the existing QM foundation.

2. Qualitative Mathematics

This chapter describes QM, a qualitative mathematics system that represents, manipulates, and describes piecewise continuous functions. QM deserves to be called *qualitative* because it understands parameterized functions, such as Equation 1.1, in addition to purely numerical ones. However, as will be discussed in section 4.3, it does not claim to be a general qualitative reasoner—just the mathematical model for one. The first three sections of this chapter discuss QM's symbolic arithmetic, data structures, and algorithms. They limit themselves to functions that have finitely many turning points and discontinuities, whereas the fourth section introduces an extension to QM that eases this restriction.

2.1 Symbolic Arithmetic

Parameterized functions and expressions mix numbers and unknown symbols, so QM implements generic arithmetic operations[1] similar to those in other symbolic algebra systems. In addition, QM associates qualitative information with expressions in the form of *constraints*. Each constraint limits an expression to an interval—finite or infinite, open, half open, or closed. Constraint information from sub-expressions propagates upwards unless overridden at a higher level, e.g.

$$\begin{cases} e_1 < v_1 \\ e_2 < v_2 \end{cases} \quad \text{implies } e_1 + e_2 < v_1 + v_2. \tag{2.1}$$

The inequality predicates—illustrated in figure 2.1— use this method to compare expressions; if e_1's lower bound is greater than e_2's upper bound then $e_1 > e_2$, if e_1's upper bound is less than or equal to e_2's lower bound then $e_1 \leq e_2$, and so on. This definition guarantees that whenever a predicate *pr* returns the result

$$(pr\ e_1\ e_2) \Longrightarrow \mathrm{T}$$

the *pr* relation actually does hold between e_1 and e_2. However, a predicate sometimes fails, due to limitations in its bounding algorithm, even when the mathematical relation it represents holds. Such cases have never yet arisen in practice, but if they do the comparison algorithm can be augmented with more sophisticated techniques.

[1] The current version of QM uses MACSYMA for algebraic simplification, while previous ones used my own simplifier.

;;; Sample constraints on a and b
(assert+ 'a); $0 < a < \infty$
(add-constraint 'b 1 T 2 NIL); $1 < b \leq 2$
;;; Implications of these constraints
(> 'a 0) $\Longrightarrow$ T
(< 'b 2) $\Longrightarrow$ NIL
(<= 'b 2) $\Longrightarrow$ T
(lb (+ 'a 'b)) $\Longrightarrow$ 1 ; lower bound
(lb (* 'a 'b)) $\Longrightarrow$ 0
(<= 'x (+ 'x (expt 'c 2))) $\Longrightarrow$ T ; $x \leq x + c^2$

Figure 2.1: Arithmetic in QM

2.2 Data Structures

QM represents a function—from $[lb,\ ub] \subseteq \Re$ to $\Re^*$—by a collection of interval descriptors, each corresponding to a closed sub-interval of its domain. The function is continuous, and strictly monotone or constant on the interiors of these interval, whereas the end points mark extrema, discontinuities or domain boundaries. For example, x^2 decreases on $[-\infty,\ 0)$ and increases on $(0,\ \infty]$; the points $-\infty$ and ∞ are domain boundaries, and zero, a minimum. Any function on $[lb,\ ub]$ that has a finite number of discontinuities and turning points, $p_1, p_2, \ldots, p_k$, can be described by a finite number of descriptors,

$$[lb,\ p_1],\ \ [p_1,\ p_2], \ldots, [p_{k-1},\ p_k],\ \ [p_k,\ ub] \tag{2.2}$$

hereafter called *fun-ints*. The set of fun-ints that describes a function is called a *functional descriptor* or FD.

Each fun-int contains the information that high school students use to sketch functions. It includes the function's direction, inflection points, end-point values, convexity, and so on, with NIL denoting unknown or undefined values. For example, the inverse does not exist on constant intervals, and may have no closed form, even on monotone intervals. Table 2.1 contains a complete list of a fun-int's attributes and table 2.2, the actual values for Equation 1.1. The *singularities* entry records derivative singularities as four-tuples

(point type left-derivative right-derivative)

where *type* is either zero, undefined, or infinite. *Convexity* entries record convexity as a list of triples *(lb sign ub)* in which *sign* is the second derivative's sign on the interval *(lb ub)*, one of -1, 0, or 1. All other entries are self explanatory.

Item	Meaning
direction	up, down, or constant
fun	the functional form $f(t)$
inverse	$f^{-1}(t)$
derivative	$f'(t)$
singularities	a list of derivative singularities
der2	$f''(t)$
convexity	a record of the second derivative's sign
lb	the interval's lower bound
lb-val	$f(lb)$
lb-r-lim	$\lim_{t \downarrow lb} f(t)$
ub	the interval's upper bound
ub-val	$f(ub)$
ub-l-lim	$\lim_{t \uparrow ub} f(t)$

Table 2.1: The Function Interval

	Interval 1	Interval 2
direction	up	down
lb	0	$-\frac{v_0}{g}$
ub	$-\frac{v_0}{g}$	$-\frac{2v_0}{g}$
fun	$\lambda t\ v_0 t + \frac{g}{2}t^2$	$\lambda t\ v_0 t + \frac{g}{2}t^2$
lb-val	0	$-\frac{v_0^2}{2g}$
lb-r-lim	0	$-\frac{v_0^2}{2g}$
ub-val	$-\frac{v_0^2}{2g}$	0
ub-l-lim	$-\frac{v_0^2}{2g}$	0
inverse	$\lambda t\ \frac{\sqrt{2gt+v_0^2}-v_0}{g}$	$\lambda t\ -\frac{\sqrt{2gt+v_0^2}+v_0}{g}$
derivative	$\lambda t\ v_0 + gt$	$\lambda t\ v_0 + gt$
singularities	NIL	NIL
der2	$\lambda t\ g$	$\lambda t\ g$
convexity	$((0\ -1\ -\frac{v_0}{g}))$	$((-\frac{v_0}{g}\ -1\ -\frac{2v_0}{g}))$

The fun-ints describing the ball's height: $h(t) = v_0 t + \frac{g}{2}t^2$ with $\begin{cases} v_0 > 0 \\ g < 0 \end{cases}$

Table 2.2: The ball's fun-ints

2.3 Algorithms

QM implements three classes of algorithms: instantiation, combination and analysis. *Instantiation* algorithms create functional descriptions, FD's, for a large family of *primitive* functions from a few basic FD's by linear substitution and scaling. For example, *i-log* starts with the basic FD for $\log y$, substitutes $y = bx + c$, multiplies by a and adds k in order to instantiate $a\log(bx + c) + k$. *Combination* algorithms apply functional operators, such as composition and addition, to FD's. Finally, *description* algorithms derive the qualitative functional behavior of FD's; they answer questions about directionality, extrema, asymptotes, and more. The following sections explain these three classes of algorithms in detail.

2.3.1 Instantiation Algorithms

All instantiation algorithms have the canonical form:

1. Find the qualitative regions in which each argument might lie.
2. Create an FD for each choice in 1 by appropriate substitutions, shifts, and scalings.

The first step performs a preliminary case analysis to determine the result's *form* and the second derives the *exact* effect of the parameters on this form. For example, the *i-quadratic* function—which instantiates $ax^2 + bx + c$—chooses a quadratic or a linear form in step 1 depending on whether a is nonzero or zero. In step 2, it applies appropriate transformations to the chosen form in order to determine its direction, convexity, and other attributes from the given parameter values; for instance, if $a > 0$ the function decreases, reaches a minimum, and increases but if $a < 0$ it increases, reaches a maximum, and decreases. Each instantiation function has its own first step determined by the particulars of its functional forms; the exponential b^x, for example, must test for $b < 0$, $b = 0$, $b = 1$, and so on.

Instantiation functions use three transformations in step 2 to produce a wide range of FD's (listed in Appendix A) from a few basic ones; these are linear substitution, linear composition, and restriction. Linear substitution produces a fun-int for $f(ax + b)$ from that of $f(x)$ by mapping each fun-int $[lb, ub]$ onto

$$\begin{cases} \left[\frac{lb}{a}, \frac{ub}{a}\right] & a > 0 \\ \left[\frac{ub}{a}, \frac{lb}{a}\right] & a < 0 \end{cases} \tag{2.3}$$

and scaling the derivatives, singularities and convexities appropriately. The new derivative values are $af'(ax+b)$ and $a^2f''(ax+b)$ by the chain rule; singularities are scaled and multiplied by a, but convexity regions are just scaled since multiplying

by a^2 leaves their values unchanged. The case $a = 0$ yields a constant function, $y = f(b)$. Linear composition produces a fun-int for $a \cdot f(x) + b$ from that of $f(x)$ without changing the existing *lb* and *ub* values. The new direction is the same as the old when $a > 0$ and the opposite when $a < 0$. The function is scaled by a and shifted by b, the derivatives and singularities are scaled by a, and the inverse is transformed appropriately. Once again, $a = 0$ yields a simple special case, $f(x) = b$. Finally, the restriction operator derives an FD for $f(x)$ restricted to a subinterval of its original domain, $[lb_1, ub_1]$, from the original FD, F_0. If lb_1 is greater than F_0's old lower bound, the *lb* value of F_0's first fun-int is replaced with lb_1 and the *lb-val* and *lb-r-lim* values by $f(lb_1)$; analogously, if ub_1 is less than the old upper bound then the *ub*, *ub-val*, and *ub-l-lim* of F_0's last fun-int are replaced by ub_1, $f(ub_1)$, and $f(ub_1)$ respectively. This procedure is valid since, by construction of F_0, f is continuous on (lb_1, ub_1).

Tables 2.3 and 2.4 demonstrate linear substitution and composition. A comparison of the tables shows the fundamental difference between the two operations—substitution scales the function's *domain* and composition, its *range*. Restriction, in turn, limits both range and domain but does not alter the function's behavior at all. All three instantiation functions share an important characteristic; they perform perfectly when certain qualitative information is available—a's sign for substitution and composition, whether $lb_1 > lb$ and $ub_1 < ub$ for restriction. If ambiguity exists, for instance $a \leq 0$, they create an FD for every possible case and record the appropriate assumption in each one. The next section describes combining algorithms that are more powerful and general than instantiation ones. However, unlike instantiation function, they do not always produce complete FD's due to limitations in their methods.

2.3.2 Combination Algorithms

Combination functions implement the functional operators: composition, addition and multiplication. They accept arbitrary FD's as input but have been used mainly on the primitive functions of Appendix A, so far. In particular, the ANALYZE program, appearing in Table 2.5, creates an FD for any expression composed of elementary mathematical functions. It casts the input, $g(x)$, into the form $a \cdot f(x) + b$, analyzes f, and scales the result. First, ANALYZE tries to match f with a pattern from its library and use the corresponding instantiation function to create an FD. For example, $2x^2 + bx$ matches[2] the pattern

$$?v_1 x^2 + ?v_2 x + ?v_3 \text{ with } ?v_1 = 2,\ ?v_2 = b \text{ and } ?v_3 = 0 \qquad (2.4)$$

[2] A simple SNOBOL type pattern matcher is used.

	x^2	$a > 0$	$a = 0$	$a < 0$
direction	*up*	*up*	*constant*	*down*
fun	x^2	$(ax+b)^2$	b^2	$(ax+b)^2$
inverse	$\sqrt{x}$	$\frac{\sqrt{x}-b}{a}$	NIL	$\frac{\sqrt{x}-b}{a}$
derivative	$2x$	$2a(ax+b)$	0	$2a(ax+b)$
der2	2	$2a^2$	0	$2a^2$
lb	0	$-\frac{b}{a}$	$-\infty$	$\frac{1-b}{a}$
lb-val	0	0	NIL	1
lb-r-lim	0	0	b^2	1
ub	1	$\frac{1-b}{a}$	∞	$-\frac{b}{a}$
ub-val	1	1	NIL	0
ub-l-lim	1	1	b^2	0
singularities	NIL	NIL	NIL	NIL
convexity	((0 1 1))	$((-\frac{b}{a}\ 1\ \frac{1-b}{a}))$	$((-\infty\ 0\ \infty))$	$((\frac{1-b}{a}\ 1\ -\frac{b}{a}))$

Linear substitution $f(ax+b)$ with $f(x) = x^2$ on $[0,1]$.

Table 2.3: Linear Substitution

	x^2	$a > 0$	$a = 0$	$a < 0$
direction	*up*	*up*	*constant*	*down*
fun	x^2	ax^2+b	b	ax^2+b
inverse	$\sqrt{x}$	$\sqrt{\frac{x-b}{a}}$	NIL	$\sqrt{\frac{x-b}{a}}$
derivative	$2x$	$2ax$	0	$2ax$
der2	2	$2a$	0	$2a$
lb	0	0	0	0
lb-val	0	b	b	b
lb-r-lim	0	b	b	b
ub	1	1	1	1
ub-val	1	$a+b$	b	$a+b$
ub-l-lim	1	$a+b$	b	$a+b$
singularities	NIL	NIL	NIL	NIL
convexity	((0 1 1))	((0 1 1))	((0 0 0))	((0 −1 1))

Linear composition of $a \cdot f(x) + b$ with $f(x) = x^2$ on $[0,1]$

Table 2.4: Linear Composition

Table 2.5: The ANALYZE algorithm

1. Represent the input, $g(x)$ as $a \cdot f(x) + b$.
2. If the function f matches some primitive function instantiate it and stop.
3. Decompose f into a sum, product, or composition of components f_i with $i = 1, \ldots, j$.
4. Analyze each f_i.
5. Apply the appropriate combination algorithm to the f_i.
6. Multiply the result from step 5 by a and add b.

so the associated *i-quadratic* function can create its FD. If no pattern matches f, it is decomposed into a sum, product, exponentiation, or functional composition of two or more functions, they are recursively analyzed, and the results are combined by a combination function. Figure 2.2 demonstrates ANALYZE on the function $(a-x^2)^{-\frac{1}{2}}$; familiarity with MACSYMA's internal syntax is presupposed, so I include translations for the uninitiated.

The composition program produces an FD for $f \circ g$ from those of f and g by mapping each fun-int g_i onto a set of fun-ints that describes $f \circ g$ on $(lb_i,\ ub_i)$ and concatenating the results. If g increases then $f \circ g$ has the same directional behavior on $(lb_i,\ ub_i)$ as f on $(lb\text{-}r\text{-}lim_i,\ ub\text{-}l\text{-}lim_i)$; if g decreases, the direction is reversed; and if g stays constant, $f \circ g$ does the same. In each case, one $f \circ g$ fun-int of known direction is created for each f fun-int; also, their *fun, derivative, der2* and *inverse*[3] entries follow from the definition of composition. Each singularity of g's transfers to $f \circ g$ and each f singularity d maps to the same type of FD at $g^{-1}(d)$; in both cases left and right values switch when g decreases. Special care must be taken when a g singularity s coincides with an f singularity $g(s)$ but this case too can be resolved directly or, if worst comes to worst, by falling back on the derivative's definition and taking limits. Table 2.6 demonstrates composition of $\sqrt{x}$ and $a - x^2$ into $\sqrt{a - x^2}$, the algorithm used by ANALYZE in figure 2.2.

The functional addition algorithm creates an FD for $f+g$ from those of f and g. It divides their domain into intervals on which neither function changes direction, creates one or more fun-ints for each interval, and concatenates the results. As in composition, the *fun, derivative* and *der2* entries follow directly from the definition of functional addition. The end-point values and limits generally can be determined by adding the respective f and g results, but the entries must be calculated from

[3]Recall that $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$

Create an FD for $\frac{1}{\sqrt{a-x^2}}$ with $a > 0$ on the interval $[-\sqrt{a}, \sqrt{a}]$.

```
(assert+ 'a)
(setq f (analyze 'f                    ;name
                 (- (expt 'a 1/2)) ;lower bound
                 (expt 'a 1/2)     ;upper bound
                 (expt (- 'a (expt 'x 2)) -1/2))) ;expression
              1
          ------------
Analyzing           2
          SQRT(A - X )

              1
          ------------
  Matching           2
          SQRT(A - X )
  Match failed.
                                        2  -1/2
  Decomposing the exponential: ; (A- X )
                  2
    Analyzing A - X

              2
      Matching X  ; note step 1.
      Instantiating (I-QUADRATIC
                     NIL ;name
                     ((MTIMES) -1 ((MEXPT) A 1/2)) ;lower bound
                     ((MEXPT) A 1/2) ;upper bound
                     1 0 0)) ;a=1, b=0, c=0 in ax^2+bx+c.
                         2
    Analysis of ( A - X )  succeeded.
                     2       -1/2
  Composing ( A - X )  with X
Analysis succeeded. (of top level)
```

Figure 2.2: A Sample ANALYZE run

	$a - x^2$	$\sqrt{a - x^2}$	$a - x^2$	$\sqrt{a - x^2}$
direction	*up*	*up*	*down*	*down*
lb	$-\sqrt{a}$	$-\sqrt{a}$	0	0
ub	0	0	$\sqrt{a}$	$\sqrt{a}$
fun	$a - x^2$	$\sqrt{a - x^2}$	$a - x^2$	$\sqrt{a - x^2}$
inverse	$\sqrt{a - x}$	$\sqrt{a - x^2}$	$\sqrt{a - x}$	$\sqrt{a - x^2}$
derivative	$-2x$	$-\frac{x}{\sqrt{a - x^2}}$	$-2x$	$-\frac{x}{\sqrt{a - x^2}}$
der2	-2	$-a(a - x^2)^{-3/2}$	-2	$-a(a - x^2)^{-3/2}$
lb-val	0	0	a	$\sqrt{a}$
lb-r-lim	0	0	a	$\sqrt{a}$
ub-val	a	$\sqrt{a}$	0	0
ub-l-lim	a	$\sqrt{a}$	0	0
singularities	NIL	NIL	NIL	NIL
convexity	$((-\sqrt{a}\ -1\ 0))$	$((-\sqrt{a}\ -1\ 0))$	$((0\ -1\ \sqrt{a}))$	$((0\ -1\ \sqrt{a}))$

This table shows the fun-ints of $a - x^2$ and the corresponding ones of $\sqrt{a - x^2}$ with $a > 0$. The fun-int for $\sqrt{x}$ appears below.

	$\sqrt{x}$
direction	*up*
lb	0
ub	∞
fun	$\sqrt{x}$
inverse	x^2
derivative	$\frac{1}{2\sqrt{x}}$
der2	$-\frac{1}{4x\sqrt{x}}$
lb-val	0
lb-r-lim	0
ub-val	NIL
ub-l-lim	∞
singularities	NIL
convexity	$((0\ -1\ \infty))$

Table 2.6: Functional Composition

Table 2.7: The FIND-DIR algorithm

1. If both functions have the same direction, return it; if one is constant return the other's direction.
2. If $(f+g)'$ is positive, zero or negative on the interval, return *up*, *constant* or *down* respectively.
3. If $ub - lb < \epsilon$, lb is very small, or ub is very big, assume that no more turning points exist.
4. If a turning point p can be found, recursively analyze $(lb,\ p)$ and $(p,\ ub)$ and combine the results. Otherwise, find the midpoint m and combine the results from $(lb,\ m)$ and $(m,\ ub)$.

their definitions when the sums are undefined. For example,

$$\lim_{x\to\infty} e^{ax} + bx \text{ with } \begin{cases} a > 0 \\ b < 0 \end{cases} \tag{2.5}$$

can not be calculated by adding the two limits since they equal ∞ and $-\infty$, so the new limit must be taken directly. However, the expression's limit at $-\infty$ does equal the sum of the two sub-limits, $0 + \infty = \infty$. Singularities can only occur at points appearing in the f or g *singularities* since the linearity of differentiation guarantees that $(f+g)'$ exists wherever f' and g' do. A *singularities* entry must be created for each singularity of f or g by adding the appropriate left and right values—read from their *singularities* or derived from their *derivative*; once again, the limits must be found directly when the sums do not exist.

The directional behavior of $f+g$ can not be determined by any general algorithm, so several special purpose techniques must be used. Table 2.7 outlines the direction finding algorithm, FIND-DIR. First, it checks whether f and g go in the same direction or whether $(f+g)'$ is non-positive or non-negative on the interval. In either of these cases FIND-DIR determines the sum's direction directly. Otherwise, the derivative must change sign at some point, p. FIND-DIR attempts to find p and analyze the two subintervals $(lb,\ p)$ and $(p,\ ub)$ recursively. If this fails—zeros of symbolic expressions don't always exist in closed form and sometimes can't even be estimated by the current algorithms—the interval is split in half and each piece recursively analyzed. This algorithm would not terminate if infinitely many turning points exist, so it assumes the heuristic that "very small" intervals have no turning points; similarly, it assumes that a largest and smallest turning point exist.

Functional multiplication parallels addition in many ways. Once again, the *fun*, *derivative*, *der2*, *lb* and *ub* follow from the definition of multiplication. End point

values and limits can be calculated by multiplying the appropriate component values and falling back on definitions when these products fail, e.g. $0 \cdot \infty$. The equality

$$(fg)' = f'g + fg' \tag{2.6}$$

guarantees that $(fg)'$ exists everywhere, except possibly for end points and singularities of f and g fun-ints. These derivatives are evaluated by the above rule, or by definition when it fails. The product's direction must be determined by FIND-DIR, as explained previously.

In summary, this section has explained the combining algorithms used by QM to implement composition, addition and multiplication on FD's. Other operators can be constructed directly from these basic ones. Functional subtraction consists of linear composition followed by addition; and functional division, of composition with $\frac{1}{x}$ followed by multiplication; that is

$$f - g = f + (-g) \text{ and } \frac{f}{g} = f \cdot \left(\frac{1}{x} \circ g\right) \tag{2.7}$$

respectively. Other operators can be implemented by relying more heavily on the *fun* expression. Differentiation, for example, need not examine the fun-int except for end-point information; it can differentiate the *fun* and call ANALYZE on the result. However, one might argue that such methods should not be classified as qualitative reasoning since they use purely syntactic knowledge. Section 4.3 will return to this issue and argue that all methods which produce results deserve recognition.

2.3.3 Description Algorithms

Description algorithms derive the functional behavior of FD's by straightforward mathematical means; this includes

- directionality,
- convexity,
- extrema,
- discontinuities,
- limits,
- singularities,
- asymptotes,

and more. They fall into two conceptual classes, *point* algorithms describing behavior at a point and *interval* algorithms describing intervals. Thus, "Is f continuous at a?" would be answered by a point algorithm and "Is f bounded on $[c, d]$?" would be answered by an interval one. However, there are no clear implementation differences between the two classes, so the rest of this section will not distinguish between them.

Most information needed by description algorithms can be derived directly from the FD's fun-ints, e.g. convexity in the *convexity* entry and directionality in the *direction* entry, but some, such as asymptotes, requires deeper analysis. For example, the functions EVAL and EVAL-INV calculate the value and inverse of a function f, represented by the FD F_0. The first one calculates $f(a)$ by finding the fun-int i in which a falls and returning

$$f(a) = \begin{cases} lb\text{-}val_i & \text{if } a = lb_i \\ ub\text{-}val_i & \text{if } a = ub_i \\ f_i(a) & \text{otherwise} \end{cases} \tag{2.8}$$

where f_i is the fun-int's *fun* form. The second calculates $f^{-1}(a)$ by finding each fun-int i such that

1. $a = lb\text{-}val_i$,
2. $a = ub\text{-}val_i$, or
3. $lb\text{-}r\text{-}lim_i < a < ub\text{-}l\text{-}lim_i$

and returning lb_i, ub_i, or $f_i^{-1}(a)$ respectively. In both cases, the intermediate value theorem guarantees correct results on the fun-ints' interiors, but the end points must be treated specially. The other description algorithms work similarly, using straightforward analytic techniques to derive functional behavior from FD's. Figure 2.3 demonstrates a few description queries; appendix B contains a complete list.

Higher level descriptions can be constructed out of the basic queries as desired. For the moment, I have written two English based programs; one of my projects for the future is implementing graphic descriptions, such as qualitative sketches. The first program, MDESCRIBE, summarizes an FD's *mathematical* behavior; it lists directionality, discontinuities, convexity, singularities and turning points. Figure 2.4 demonstrates MDESCRIBE's description of the FD $\frac{1}{\sqrt{a-x^2}}$ from Table 2.6. The second, QDESCRIBE, summarizes the joint *qualitative* behavior of an FD set. The description divides time into intervals according to the FD's directions; each time one FD changes direction a new interval begins. The boundary points between intervals, corresponding to extrema or discontinuities, are described in detail. QDESCRIBE accepts a list of *significant values* (which defaults to 0) as an optional argument;

;;; Point descriptions of $|x|$
;;; Is $|x|$ continuous at 0?
(send i-abs1 :continuousp 0) $\Longrightarrow$ T
;;; Is $|x|$ differentiable at 0?
(send i-abs1 :differentiablep 0) $\Longrightarrow$ NIL

;;; Interval descriptions of $\frac{1}{x}$
;;; Is it bounded on $(0, \infty)$?
(send i-hyp1 :s-boundedp :lb1 0) $\Longrightarrow$ NIL
;;; on $(1, \infty)$?
(send i-hyp1 :s-boundedp :lb1 1) $\Longrightarrow$ T
;;; What are its asymptotes?
(send i-hyp1 :s-asymptotes) $\Longrightarrow$ $[x = 0,\ y = 0]$

Figure 2.3: Sample Description Queries

every point at which any FD takes on a significant value is mentioned. The next chapter illustrates QDESCRIBE on several networks.

2.4 Periodic Functions

QM, as described so far, limits itself to functions that have finitely many turning points and discontinuities. This section extends the model to *periodic* functions by parameterizing the corresponding fun-ints; for example, $\cos x$ can be represented by two fun-ints, one for $[2n\pi,\ (2n+1)\pi]$ and the other for $[(2n+1)\pi,\ 2(n+1)\pi]$, with n taking on each of the values $\ldots -1, 0, 1 \ldots$ in turn. In fact, any function that fits this model, such as $floor(x)$ and $ceiling(x)$, can be represented—not just periodic ones. QM records a parameterized function as a *par-fun-int*, a finite list of fun-ints along with lower and upper bounds for the parameter—hereafter n. For example, table 2.8 contains the fun-ints for a square wave with linear rise time r and period w; the function's first period, corresponding to $n = 0$, appears in figure 2.5.

The instantiation functions of section 2.3.1 extend to par-fun-ints; linear substitution and composition are simply applied to each parameterized fun-int in turn, but restriction requires additional effort. Figure 2.6 contains the algorithm for restricting to lb' the lower bound of a par-fun-int, *par-int*, made up of fun-ints: $int_1,\ int_2 \ldots, int_p$, in which n varies between *n-min* and *n-max*; the upper bound algorithm is analogous. The expression exp^k denotes the result of substituting k for n in *exp*.

```
Description of the function F defined between -SQRT(A) and SQRT(A)
The function is undefined at - SQRT(A)
The right limit is INFINITY
The function decreases monotonically between -SQRT(A) and 0
The derivative is undefined at - SQRT(A)
The right derivative is -INFINITY
                                  1
The function's value at 0 is -------
                               SQRT(A)
The point is a local minimum.
The function increases monotonically between 0 and SQRT(A)
The derivative is undefined at SQRT(A)
The left derivative is INFINITY
The function is undefined at SQRT(A)
The left limit is INFINITY
```

Mathematical description of $\frac{1}{\sqrt{a-x^2}}$ produced from the FD in Figure 2.2.

Figure 2.4: A Mathematical Description

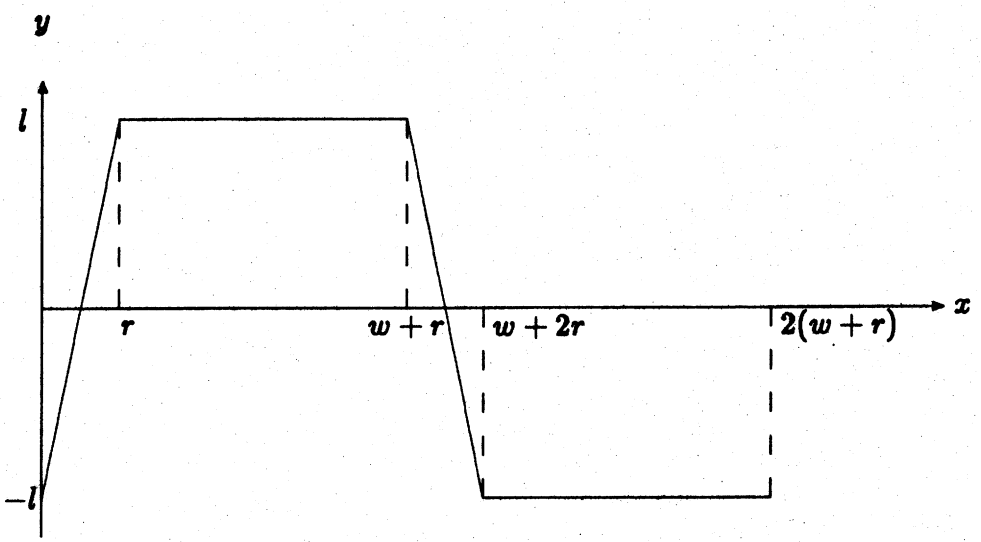

Figure 2.5: A Periodic Function

	Interval 1	*Interval* 2	*Interval* 3	*Interval* 4
direction	*up*	*constant*	*down*	*constant*
lb	$2n(w+r)$	$2n(w+r)+r$	$(2n+1)(w+r)$	$(2n+1)(w+r)+r$
ub	$2n(w+r)+r$	$(2n+1)(w+r)$	$(2n+1)(w+r)+r$	$2(n+1)(w+r)$
lb-val	$-l$	l	l	$-l$
lb-r-lim	$-l$	l	l	$-l$
ub-val	l	l	$-l$	$-l$
ub-l-lim	l	l	$-l$	$-l$
fun	$\frac{2l}{r}(x-lb)-l$	l	$l-\frac{2l}{r}(x-lb)$	$-l$
inverse	$\frac{(x+l)r}{2l}+lb$	NIL	$\frac{(l-x)r}{2l}+lb$	NIL
derivative	$\frac{2l}{r}$	0	$-\frac{2l}{r}$	0
der2	0	0	0	0
singularities	$((lb\ 0\ \frac{2l}{r}))$ $((ub\ \frac{2l}{r}\ 0))$	$((lb\ \frac{2l}{r}\ 0))$ $((ub\ 0\ -\frac{2l}{r}))$	$((lb\ 0\ -\frac{2l}{r}))$ $((ub\ -\frac{2l}{r}\ 0))$	$((lb\ -\frac{2l}{r}\ 0))$ $((ub\ 0\ \frac{2l}{r}))$
convexity	$((lb\ 0\ ub))$	$((lb\ 0\ ub))$	$((lb\ 0\ ub))$	$((lb\ 0\ ub))$

Table 2.8: Parameterized fun-ints

The composition, addition and multiplication algorithms have not been implemented on par-fun-ints, nor have many description algorithms. However, MDESCRIBE extends to par-fun-ints by summarizing the parameterized behavior of their fun-ints. QDESCRIBE describes the first period of a par-fun-int in detail and summarizes all other periods. If the function is periodic or damped periodic, the summary states that fact along with the damping ratio (if relevant) and limit at infinity; otherwise, it calls MDESCRIBE to provide a mathematical description. A par-fun-int is considered periodic if each of its fun-ints satisfies:

1. the *ub, singularities* and *convexity* entries are a fixed distance, independent of n, from the *lb*, and
2. the *fun* satisfies $fun(x)^k = fun(x)^{k+1}$, $fun(x)^k > fun(x)^{k+1}$ or $fun(x)^k < fun(x)^{k+1}$ independently of k and x.

The first condition guarantees that the interval's shape is the same for all values of n, and the second, that the function's values are scaled uniformly on successive periods. Periodic functions are further classified as cyclic, decreasing or increasing, according to the predicate, from step 2, that they satisfy. This section concludes with mathematical and qualitative descriptions of the square wave par-fun-int from table 2.8, performed by MDESCRIBE and QDESCRIBE. In order to improve legibility, I have typeset the former in a regular font.

1. Find the least k for which $lb' \geq lb(int_1)^k$.
2. Find the least i for which $lb' < ub(int_i)^k$.
3. Restrict the lower bound of int_i^k to lb'.
4. Adjust n to vary between $k+1$ and *n-max*.
5. Return the intervals: $int_i^k, \ldots, int_p^k$ and *par-int*.

Restrict a par-fun-int's lower bound to lb'.

Figure 2.6: Restriction of a Par-fun-int

Mathematical Description of the Square Wave

A unique description exists.
Description of the function IMP defined between 0 and ∞
Constraints: $0 < r$, $0 < l$, $0 < w$
The function's value at 0 is $-l$.
The point is a right minimum.
The function is described parametrically for $n = 0$ to ∞.

The function increases monotonically on the intervals $2n(w+r)$ to $2n(w+r)+r$
The function's value at $2n(w+r)$ is $-l$
The derivative is undefined at $2n(w+r)$
The left derivative is 0 and the right is $\frac{2n}{r}$
The derivative is undefined at $2n(w+r)+r$
The left derivative is $\frac{2n}{r}$ and the right is 0
The function is linear between $2n(w+r)$ and $2n(w+r)+r$

The function is constant on the intervals $2n(w+r)+r$ to $(2n+1)(w+r)$
The function's value at $2n(w+r)+r$ is l
The derivative is undefined at $(2n+1)(w+r)$
The left derivative is 0 and the right is $-\frac{2n}{r}$

The function decreases monotonically on the intervals $(2n+1)(w+r)$ to $2(n+1)(w+r)$
The function's value at $(2n+1)(w+r)$ is l
The derivative is undefined at $(2n+1)(w+r)+r$
The left derivative is $-\frac{2n}{r}$ and the right is 0
The function is linear between $(2n+1)(w+r)$ and $(2n+1)(w+r)+r$

The function is constant on the intervals $(2n+1)(w+r)+r$ to $2(n+1)(w+r)$
The function's value at $(2n+1)(w+r)$ is $-l$
The derivative is undefined at $2(n+1)(w+r)$
The left derivative is 0 and the right is $\frac{2n}{r}$

The function has no limit at infinity.

Qualitative Description of the Square Wave

```
There is a unique qualitative description of IMP.
-------------------------------------------------
Time point 1: 0
IMP's value is - L

Between points 1 and 2:
IMP increases and is linear.                    R
IMP goes through the significant value 0 at     -
--------------------------                      2
Time point 2: R
It is significant because IMP reaches a maximum
IMP's value is L

Between points 2 and 3:
IMP is constant.
--------------------------
Time point 3: W + R
IMP's value is L

Between points 3 and 4:

IMP decreases and is linear.                    2 W + 3 R
IMP goes through the significant value 0 at     ---------
--------------------------                          2
Time point 4: W + 2 R
It is significant because IMP reaches a minimum
IMP's value is - L

Between points 4 and 5:
IMP is constant.
--------------------------
Time point 5: 2 (W + R)
IMP's value is - L
-------------------------------------------------
IMP repeats the qualitative behavior from [ 0 , 2 (W + R) ]
 on [ 2 N (W + R) 2 (N + 1) (W + R) for n=1 to infinity
It is periodic.
```

2.5 Summary of QM

This chapter has described QM, a qualitative mathematics system for piecewise continuous real-valued function of one variable. QM represents a function as a collection of fun-ints on which it is monotone and continuous. Instantiation algorithms produce FD's for a wide class of elementary functions by applying linear substitution, scaling and shifting to a few basic FD's. Combination algorithms, as shown in table 2.5, implement functional operators on FD's and allow any expression made up of elementary function to be analyzed by decomposition. Finally, description algorithms use straightforward analytic techniques to derive qualitative and quantitative functional behavior from FD's. The next chapter will demonstrate QM's role in qualitative reasoning.

3. Qualitative Reasoning

As explained in chapter 1, QMR analyzes dependency networks in two stages. First, QR derives explicit closed-form expressions that characterize the network's behavior, then QM analyzes these expressions to produce FD's. This chapter describes the first component, QR; the second was explained in the previous chapter.

3.1 The QMR Algorithm

A network consists of edges and nodes, representing functions and functional operators. Each node has a single out-edge and zero or more in-edges. The out-edge functionally equals the result of applying the node's operator to its in-edges. For example, node 1 in Figure 3.1 has no inputs and outputs the fixed function f while node 2 has two inputs and outputs their sum. Table 3.1 lists all existing node types along with the relations between their in and out nodes.

QR translates a network into a set of simultaneous differential equations, each expressing the relation between the *in* and *out* links of a single node. It solves the equations, using the given initial values, and calls ANALYZE (table 2.5) to parse the solutions and create FD's. This rudimentary algorithm fails on networks that lack closed-form solutions; in fact, it only solves linear equations with constant coefficients and other simple cases[1]. Nevertheless, as the next section shows, QR solves many interesting problems. Most of these examples appear in other qualitative reasoning papers, so section 4.3 returns to them when it compares QMR with other systems.

3.2 QMR Examples

3.2.1 The Falling Ball

This example contains QR's solution to a problem from Kuipers' work [13,16]. A ball is thrown straight up from height 0 with velocity v_0 at time 0; it is pulled down by constant negative gravitional acceleration, g. QR derives closed form solutions for its velocity, v, and height, h, from the network in figure 3.3. ANALYZE produces FD's, and QDESCRIBE describes the results. Figure 3.2 depicts the solution pictorially by graphing the ball's height over time with arrows indicating its velocity. This

[1]It calls MACSYMA but could just as well use an alternate differential equation solver.

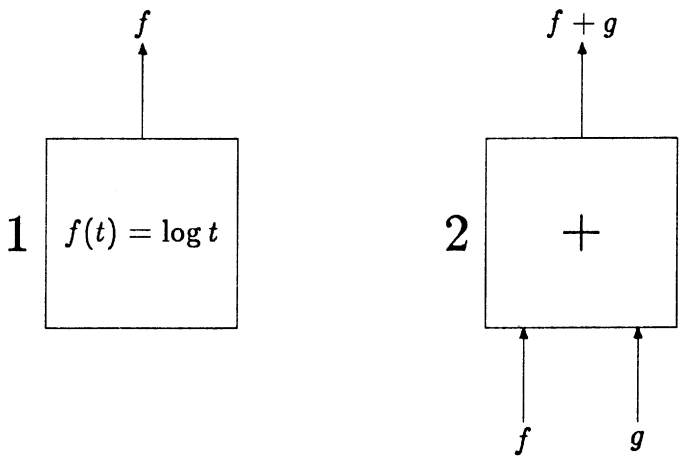

Figure 3.1: Two Sample Nodes

Type	*In* **Links**	*Out* **Links**
fixed	none	the given function
derivative	$f(t)$	$f'(t)$
integral	$f(t)$	$\int_0^t f(s)ds$
sum	$f_1(t), f_2(t) \ldots, f_n(t)$	$\sum_{i=1}^{n} k_i f_i(t)$ with k_i given constants
product	$f_1(t), f_2(t) \ldots, f_n(t)$	$k \prod_{i=1}^{n} f_i(t)$ with k a given constant
composition	f, g	$f \circ g$

Table 3.1: Node Types

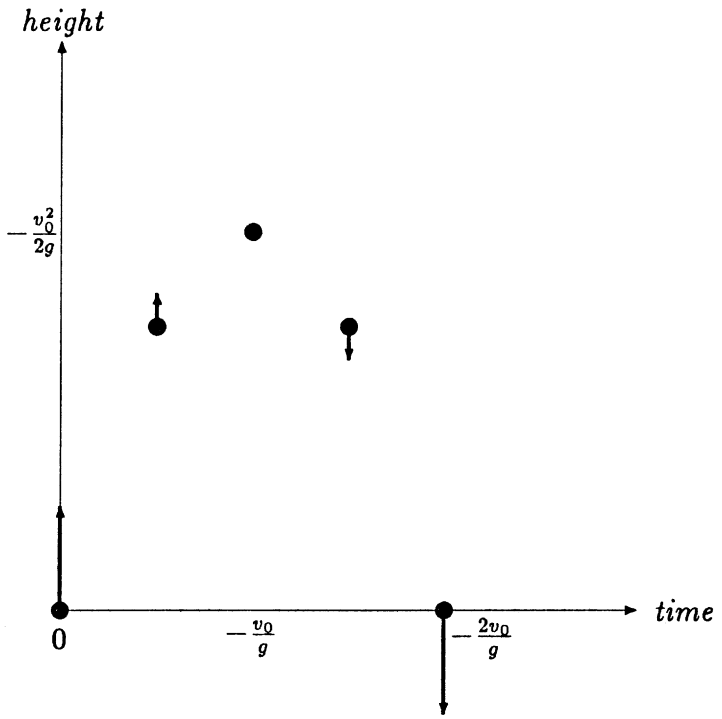

Figure 3.2: The Ball's Flight

simple model predicts that the ball will fall indefinitely since it ignores the effect of distance from the center of the earth on gravitation, or indeed the presence the earth's surface.

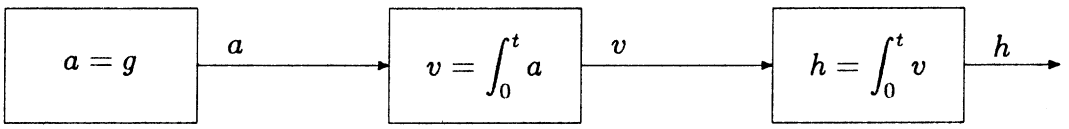

Figure 3.3: The Ball Network

```
(solve-net ball) ;solve the ball network
                2
          G X  + 2 VO X
[[H(X) = -------------, V(X) = G X + VO, A(X) = G]]
                2
;;; parsing the solutions
Instantiating (I-QUADRATIC NIL 0 INFINITY G (* 2 VO) 0) ; H(x)
Instantiating (I-LINEAR NIL 0 INFINITY G VO) ;V(x)

(qdescribe ball) ; describe the solutions
There is a unique qualitative description of H, V.
-------------------------------------------
Time point 1: 0
H's value is 0
V's value is VO

Between points 1 and 2:
H increases and decelerates.
V decreases and is linear.
--------------------------
                 VO
Time point 2: - --
                 G
It is sigificant because H reaches a maximum
                  2
                VO
H's value is - ---
                2 G
V's value is 0

Between points 2 and 3:
H decreases and accelerates.
V decreases and is linear.                    2 VO
H goes through the significant value 0 at - ----
                                               G
V's value is - VO
--------------------------
Time point 3: INFINITY
H approaches -INFINITY
V approaches -INFINITY
```

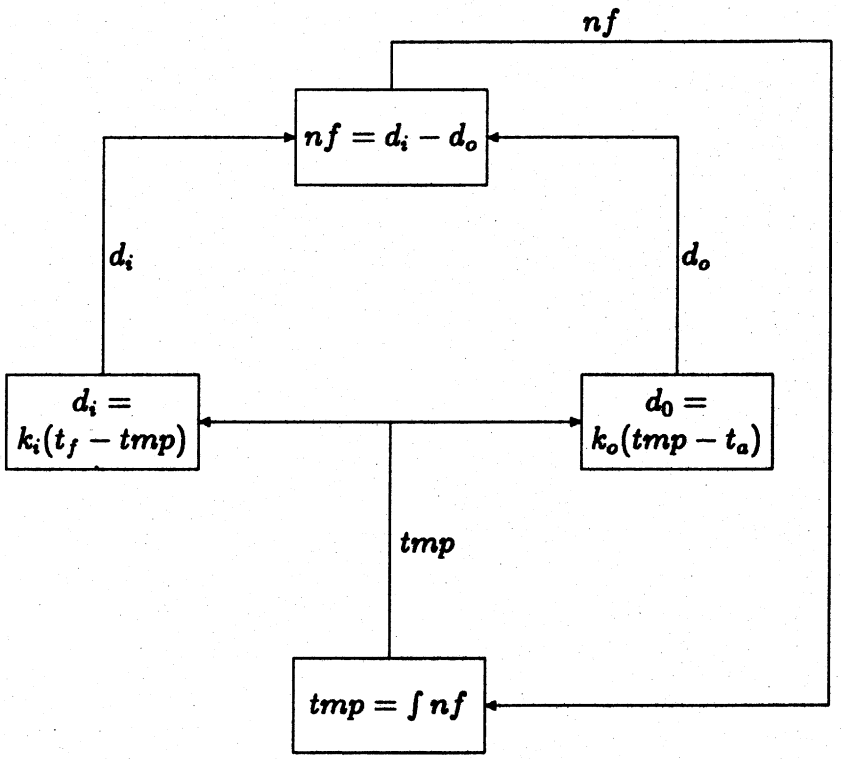

Figure 3.4: The Heat Flow Network

3.2.2 Heat Flow

This example demonstrates QMR's analysis of a heat flow network—shown in figure 3.4—equivalent to the one discussed by Kuipers [15]. A container is heated from temperature t_c by a flame whose temperature is t_f; air temperature is t_a and $t_a < t_c < t_f$. The inflow of heat, d_i, is directly proportional to the temperature gradient between the flame and the container. Similarly, the outflow, d_o, is directly proportional to the gradient between the container and the outside air. Finally, the temperature inside the container, $tmp(t)$, equals the integral of the net heat flow. QMR finds the closed-form solution

$$tmp(t) = e^{-(k_0+k_i)t}\,(t_c - t_{eql}) + t_{eql} \quad \text{with } t_{eql} = \frac{k_i t_f + k_o t_a}{k_i + k_o} \tag{3.1}$$

and produces three possible FD's, depending on whether t_c is greater than, equal to, or less than the equilibrium temperature, t_{eql}. The qualitative descriptions of the three possible FD's appear below.

```
There are 3 possible descriptions of TMP.
-----------------------------------------------
;;; Note that t0 is greater than the equilibrium value.
Time point 1: 0
H's value is TC

Between points 1 and 2:
H decreases and decelerates.
--------------------------
Time point 2: INFINITY
              KI TF + KO TA
H approaches -------------
                 KO + KI
-----------------------------------------------
;;; Note that t0 is equal to the equilibrium value.
Time point 1: 0
              KI TF + KO TA
H's value is -------------
                 KO + KI

Between points 1 and 2:
H is constant.
--------------------------
Time point 2: INFINITY
              KI TF + KO TA
H approaches -------------
                 KO + KI
-----------------------------------------------
;;; Note that t0 is smaller than the equilibrium value.
Time point 1: 0
H's value is TC

Between points 1 and 2:
H increases and decelerates.
--------------------------
Time point 2: INFINITY
              KI TF + KO TA
H approaches -------------
                 KO + KI
```

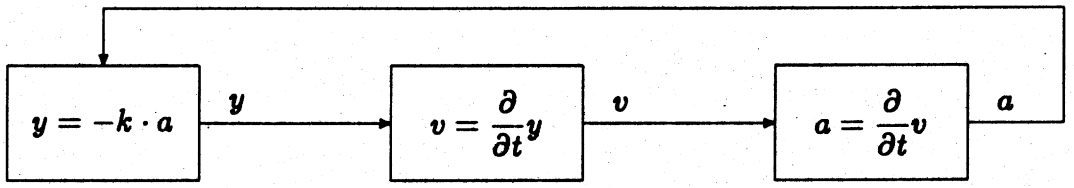

Figure 3.5: The Spring Network

field	fun-int 1	fun-int 2
direction	*down*	*up*
fun	$y_0 \cos \frac{t}{\sqrt{k}}$	$y_0 \cos \frac{t}{\sqrt{k}}$
inverse	$\sqrt{k}\left[2\pi n + \arccos \frac{t}{y_0}\right]$	$\sqrt{k}\left[2\pi(n+1) - \arccos \frac{t}{y_0}\right]$
der-fun	$-\frac{y_0}{\sqrt{k}} \sin \frac{t}{\sqrt{k}}$	$-\frac{y_0}{\sqrt{k}} \sin \frac{t}{\sqrt{k}}$
der2-fun	$-\frac{y_0}{k} \cos \frac{t}{\sqrt{k}}$	$-\frac{y_0}{k} \cos \frac{t}{\sqrt{k}}$
lb	$2\pi n\sqrt{k}$	$(2n+1)\pi\sqrt{k}$
lb-val	y_0	$-y_0$
lb-r-lim	y_0	$-y_0$
ub	$(2n+1)\pi\sqrt{k}$	$2(n+1)\pi\sqrt{k}$
ub-val	$-y_0$	y_0
ub-r-lim	$-y_0$	y_0
der-map	NIL	NIL
der2-map	$\left(\left(2\sqrt{k}\pi n\ {-1}\ (2n+\frac{1}{2})\sqrt{k}\pi\right)\right.$ $\left.\left((2n+\frac{1}{2})\sqrt{k}\pi\ 1\ (2n+1)\sqrt{k}\pi\right)\right)$	$\left(\left((2n+1)\sqrt{k}\pi\ 1\ (2n+\frac{3}{2})\sqrt{k}\pi\right)\right.$ $\left.\left((2n+\frac{3}{2})\sqrt{k}\pi\ {-1}\ 2(n+1)\sqrt{k}\pi\right)\right)$

Table 3.2: Parameterized *fun-ints*

3.2.3 Oscillation

This example, taken from Kuipers [13], demonstrates QMR's analysis of periodic functions, using par-fun-ints. A frictionless spring is extended to length y_0, relative to its natural length, and released with velocity 0. Acceleration and length obey the equation $y(t) = -ka(t)$ with $k > 0$. The complete functional network appears in figure 3.5. QMR represents the solution, $y(t) = y_0 \cos \frac{t}{\sqrt{k}}$, by the two parameterized fun-ints, shown in table 3.2; the parameter n varies from 0 to infinity. QDESCRIBE only describes the first period since the function fulfills the conditions stated in section 2.4. There are no singularities; inflection points and upper bounds are a fixed distance from lower bounds; and values are cyclic.

```
There is a unique qualitative description of Y.
-----------------------------------------------
Time point 1: 0
Y's value is Y0

Between points 1 and 2:
Y decreases.                    %PI SQRT(K)
It accelerates between 0 and ------------
                                     2
                          %PI SQRT(K)
It decelerates between ------------ and %PI SQRT(K)
                                2
%PI SQRT(K)
------------ is an inflection point
     2                                        %PI SQRT(K)
Y goes through the significant value 0 at ------------
--------------------------                         2
Time point 2: %PI SQRT(K)
It is significant because Y reaches a minimum.
Y's value is - Y0

Between points 2 and 3:
Y increases.                                  3 %PI SQRT(K)
It accelerates between %PI SQRT(K) and --------------
                                                2
                          3 %PI SQRT(K)
It decelerates between -------------- and 2 %PI SQRT(K)
                                2
3 %PI SQRT(K)
-------------- is an inflection point
      2
                                               3 %PI SQRT(K)
Y goes through the significant value 0 at --------------
--------------------------                         2
Time point 3: 2 %PI SQRT(K)
Y's value is Y0
-----------------------------------------------
Y repeats the qualitative behavior from [ 0 , 2 %pi SQRT(K) ]
  on [ 2 SQRT(K) N PI , 2 SQRT(K) (N + 1) PI ] for n= 1 to infinity.
It is periodic.
```

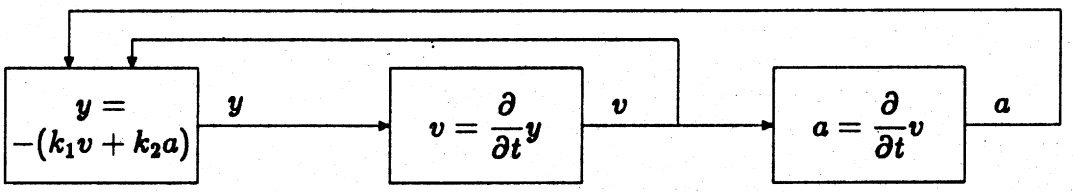

Figure 3.6: The Damped Spring Network

3.2.4 Damped Oscillation

This section describes a spring that is affected by friction; it includes the previous example as a special case. Once again, a spring is extended to length y_0, relative to its natural length, and released with velocity 0. It obeys the equation

$$y(t) = -k_1 v(t) - k_2 a(t) \text{ with } \begin{cases} k_1 > 0 \\ k_2 > 0 \end{cases} \tag{3.2}$$

where k_1 is the frictional coefficient. The functional network appears in figure 3.6; its solution is the damped cosine wave,

$$y(t) = 2y_0\sqrt{k_2}\Delta e^{-\frac{k_1}{k_2}t} \cos\left[\frac{t}{\beta} - \arctan(k_1\Delta)\right] \text{ with } \begin{cases} \Delta = \frac{1}{\sqrt{4k_2 - k_1^2}} \\ \beta = 2k_2\Delta \end{cases} \tag{3.3}$$

described below. Once again, QDESCRIBE summarizes the periodic behavior qualitatively, rather than resort to a tedious mathematical description. In order to improve legibility, I have typeset the description in a regular font and substituted the symbols Δ and β for their definitions.

There is a unique qualitative description of Y.
Time point 1: 0
Y's value is y_0
Between points 1 and 2:
Y decreases.
It accelerates between 0 and $\beta\,[\arctan k_1\Delta + \arctan(2k_2 - k_1)\Delta]$
It decelerates between $\beta\,[\arctan k_1\Delta + \arctan(2k_2 - k_1)\Delta]$ and $\pi\beta$
$\beta\,[\arctan k_1\Delta + \arctan(2k_2 - k_1)\Delta]$ is an inflection point
Y goes through the significant value 0
Time point 2: $\pi\beta$
It is significant because Y reaches a minimum.
Y's value is $-y_0 e^{-\pi k_1\Delta}$
Between points 2 and 3:
Y increases.

It decelerates between $\pi\beta$ and $\beta\,[\arctan k_1\Delta + \arctan(4k_2 - k_1)\Delta]$
It accelerates between $\beta\,[\arctan k_1\Delta + \arctan(4k_2 - k_1)\Delta]$ and $2\pi\beta$
$\beta\,[\arctan k_1\Delta + \arctan(4k_2 - k_1)\Delta]$ is an inflection point
Y goes through the significant value 0

Time point 3: $2\pi\beta$
Y's value is $y_0 e^{-2\pi k_1\Delta}$

Y repeats the qualitative behavior from $[0,\ 2\pi\beta]$
on $[2n\pi\beta,\ 2(n+1)\pi\beta]$ for $n = 1$ to ∞.
Its magnitude decreases by a factor of $e^{-2\pi k_1\Delta}$ on each interval.
The limit at ∞ is 0.

4. Comparison with Related Work

This chapter reviews and evaluates current research efforts in qualitative reasoning; in particular, it focuses on the *qualitative physics*, or QP, paradigm. Section 4.1 presents the QP philosophy and describes a generic QP system. It also cites several existing systems and explains how they fit into the generic theory; QP derives its data structures primarily from Forbus [11], but its algorithms are synthesized from several sources. Section 4.2 elaborates the argument (implicit in chapter 1) that QP systems must acquire a richer model of functions, while section 4.3 illustrates QMR's ability to fill that lacuna.

4.1 Overview of QP

Physics models interdependent, quantifiable, real-world phenomena by functional dependencies involving their quantified forms. For example, Newton's second law, $f = ma$, relates force, mass, and acceleration for any given object. A system's behavior can often be determined by finding a closed form solution to the equation which it obeys and examining the functional form of that solution. Even if a closed form is unobtainable, numerical analysis or simulation techniques yield approximate solutions, which generally suffice. Thus, the power and precision of mathematics is harnessed by modern day physics.

Qualitative Physics attempts to generalize the methods of Mathematical Physics and apply them to partially specified systems in which certain parameters are unknown or unimportant. This would allow common sense informal reasoning—as performed by human experts—to be mechanized. Common sense reasoning is a powerful tool in many domains, so it would seem that a system with that capability could acquire previously unreached levels of expertise. It could model problems as constraint networks similar to the functional networks of Mathematical Physics (and of QMR), but drawn at a higher level of abstraction, one sufficient to express the relevant issues and suppress the trivia. These networks would degrade gracefully when faced with partial knowledge, producing less specific models rather than failing. The remainder of this section outlines QP's data structures and algorithms and illustrates their capabilities.

4.1.1 Data Structures

A QP system consists of *quantities* and *constraints*, representing properties and

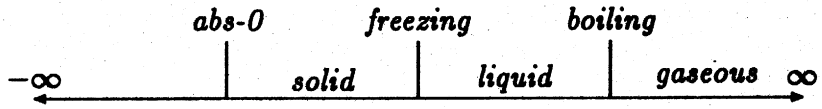

Figure 4.1: Value Space for Water

laws respectively. Quantities are time dependent functions which map into *value spaces* rather than real (or complex) numbers. A value space describes a subinterval of $\Re$ abstractly by breaking it into a finite number of open regions separated by boundary points. For example, a pot of water has quantity $temp(t)$ which represents its temperature at time t and lies in the value space depicted in Figure 4.1 with boundaries: *abs-0, freezing* and *boiling*, and regions: *solid, liquid* and *gaseous*. Quantities are best viewed as real valued functions whose exact values have been abstracted into *qualitative values*, regions or boundary points.

Derivatives of quantities with respect to time are first class quantities in their own right; in fact, any higher order derivative can be described by recursive application of this rule. All current programs assume that quantities are smooth within qualitative regions. As in standard calculus, a quantity increases, remains constant, or decreases depending on whether its *qualitative derivative* is positive, zero, or negative. Hence, these three values appear in every derivative's value space.

Constraints state functional equations that must hold between quantities, for example, $p(t)v(t) = k \cdot temp(t)$ constrains the pressure, temperature, and volume of the gas in a sealed container. Ordinary differential equations are obtained in a similar manner by relating quantities to their derivatives. Frictionless constant gravitation, for example, can be expressed as

$$v'(t) = a(t), \quad a(t) < 0, \quad a'(t) = 0 \tag{4.1}$$

with a representing acceleration and v, velocity. These *algebraic* constraints involve sums, products, equalities, and inequalities. A second type, *functional* constraints, state direct or inverse monotone (not necessarily linear) dependencies between quantities, such as $y = M^+(x)$, and provide a higher level of abstraction than algebraic ones.

The constraints on a quantity may vary when it moves from one qualitative region to another. For example, the relation between water's temperature and volume is different for each of the three regions in its value space. Quantities which obey different constraints in different regions model real world systems whose behavior depends on the operating range of their parameters. In summary, QP models physical systems with quantities and constraints; the next section describes the algorithms which it uses to analyze these models.

4.1.2 Algorithms

QP derives a system's behavior from its constraints and initial values by determining the successive qualitative states which it enters. Each state consists of the qualitative values and derivatives of the system's quantities. Thus, the water pot system could have

- $temp(t_1) \in$ *liquid* and *increasing*
- $temp(t_2) =$ *boiling* and *constant*

as consecutive states. Analysis proceeds by repeated execution of the steps:

1. Propagate initial values and
2. Derive the next transition and its initial values.

The first step utilizes arithmetic rules such as

$$\text{if } \begin{cases} a \geq 0 \\ b \geq 0 \end{cases} \text{ then } \begin{cases} a+b \geq 0 \\ ab \geq 0 \end{cases} \tag{4.2}$$

and analytic theorems such as

- The quantity a increases, remains constant and decreases when a' is positive, zero and negative respectively.
- if $a = b + c$, b is increasing and c is constant then a is increasing.

to derive all qualitative values and derivatives from the initial values. For example, if v and v' obey the constraint

$$v(t) = av'(t) + b \text{ with } \begin{cases} a < 0 \\ b < 0 \end{cases} \tag{4.3}$$

and $v_0 > 0$ then v' will initially be negative by the arithmetic inequality. Thus, v decreases by the first analytic constraint and so v' increases towards zero by the second. Ambiguities occur when purely qualitative information does not imply a single qualitative result, for instance $f - g$ may be positive or negative when f and g are positive. They must either be resolved by quantitative information—in our case, information which determines that $f > g$ or $f < g$—or treated as mutually exclusive alternatives.

The second step uses continuity theorems such as

if $f(0) > 0$ then f will remain positive in a neighborhood of zero

and intuitions such as

if f approaches a point then it will reach that point

to deduce which quantities, if any, will leave their current qualitative regions first and what their new qualitative values will be. For example, if the quantity g starts out positive and g' obeys the constraint $g'(t) = at$ with a negative then g will reach zero after some open time interval. At this point, a transition occurs and the system enters a new qualitative state, possibly involving new constraints. The propagation/transition cycle begins anew with initial value $g = 0$ and *decreasing*. Once again, ambiguities may arise due to lack of quantitative detail. A quantity may either leave a region or approach its boundary asymptotically; similarly, either of two quantities which approach boundaries might cross first or both might cross simultaneously. The result in each case depends on the magnitude of derivatives, not just their signs.

The analysis algorithm, described above, parallels a numerical simulation at a qualitative level of abstraction. Hence, it suffers from simulation's major weakness; it produces a trace of a system's incremental behavior but lacks global perspective. Periodic behavior causes the analysis program to run on forever, repeating a chain of states over and over and ambiguous behavior forces it to branch and produce multiple descriptions (some of which may never terminate if they branch or loop themselves). For these reasons, QP produces a *history*, a graph whose nodes represent states and links, possible transitions. Periodicity shows up as cycles and ambiguity as multiple out-links. In summary, analysis derives a history for a system by determining all possible states reachable from its initial state.

4.1.3 Current Systems

Several QP systems have been proposed, built, and demonstrated over the past few years. All define quantities, constraints, and propagation similarly though no two have the same transition rules or algorithms. De Kleer's QUAL[1] [2] derives the small signal behavior of circuits; hence, it need not analyze transitions from state to state or consider devices with multiple operating regions. It works by applying qualitative constraint propagation (the first step of QP's analysis algorithm) to a network model. This model includes qualitative versions of resistors, capacitors and inductors, along with Kirchoff's laws.

De Kleer and Brown's ENVISION [4–7] extends QUAL to a general system dynamics model in which nodes process "material" and pipes transfer material from node to node. Behavior is categorized in terms of "flow" and "pressure" which generalize current and voltage; generalized versions of Kirchoff's laws also apply. Unlike QUAL, it performs a complete qualitative analysis and produces a history.

[1]QUAL evolved from De Kleer's NEWTON [1], a prototypical qualitative reasoner that analyzed frictionless motion of point masses along two dimensional tracks.

ENVISION also allows devices to have multiple operating regions, each governed by its own constraints. Williams's Temporal Qualitative Analysis [22,23] extends QUAL in a manner similar to ENVISION and also describes a feedback analysis algorithm. It restricts itself to circuit theory but could be extended to the general system dynamic model. Williams provides a clean precise semantics for all of his algorithms and clearly states the continuity and differentiability assumptions that others often neglect. De Kleer and Bobrow [3] extend ENVISION to reason about higher order derivatives rather than just quantities and their first derivatives. This gives derivatives the first class status ascribed to them in QP and allows inflection and other higher order effects to be expressed.

Forbus's Qualitative Process Theory[2] [11,12] and Kuipers's ENV [13–15] define quantities, regions and constraints in the same way as the generic QP theory. Forbus stresses construction of networks for given problem domains and attempts to formulate a theory thereof. In contrast, Kuipers focuses on generalizing the theory of differential equations to qualitative networks[3] but places little emphasis on their construction. Finally, De Kleer and Brown [8] and Doyle [9] review the QP theory and compare current QP systems.

4.2 Problems with QP

QP's qualitative simulation paradigm offers insight into *naive* reasoning techniques but fails to model *expert* behavior. Indeed, it largely ignores three widely accepted features of expertise

1. large bodies of compiled knowledge,
2. hierarchical abstractions and
3. domain specific representations and algorithms.

First, experts summarize important recurrent systems as *clichés*, concise descriptions of their behavior and appearance. Future problems that match clichés need not be analyzed since the expert remembers how they behave. Second, even totally new systems often decompose into a few interconnected sub-systems. The expert ignores the sub-systems' inner workings and treats them as black boxes which implement specified functions. He analyzes the entire system in terms of the interactions between its components, reducing complexity by abstraction. The sub-systems, in turn, may be matched against clichés or decomposed further, leading to a hierarchy

[2]Forbus, like De Kleer, based his work on a prototype, FROB [10], that determined the ways in which a point mass could bounce on a polygonal surface.

[3]See his examples in [16–21].

of abstractions. For example, an idealized amplifier transforms its input $f(t)$ into $kf(t)$ with $k > 1$ so two amplifiers in series transform $f(t)$ into $k^2 f(t)$. Ignoring the amplifiers' electronic components and concentrating on their external behavior allows this simple analysis. Finally, special purpose methods for representing and solving domain specific problems form a crucial component of expertise. They provide quick accurate solutions to many problems and should be used when applicable. The premier example is mathematical modeling of physical systems; any system that reduces to a simple set of differential equations can be analyzed easily and precisely using elementary calculus.

QP systems certainly match problems against stored cliché networks and decompose them into simpler sub-systems. QUAL, for instance, uses a hierarchical set of patterns to aggregate circuit elements into larger and larger components. However, in order to treat sub-systems abstractly, QP must replace each one by a high-level constraint that relates its inputs and outputs, but ignores internal structure. These constraints can be included in user supplied clichés, but I see no easy way for QP to derive them from *histories*. It can not learn new clichés or perform multi-level decompositions unless it develops abstraction facilities capable of deducing constraints from histories, a research project in its own right.

A third—and, in my eyes, more fundamental—limitation is that QP's data structures do not contain the information needed for precise mathematical analysis. They do not allow constraints to contain explicit time dependencies and so preclude closed form descriptions such as

$$f(t) = t^2 \text{ or } g(t) = at \text{ with } a > 0. \tag{4.4}$$

Even if such dependencies were permitted, QP's analysis algorithm could not determine that

$$f(t) > g(t) \text{ for } t > a \tag{4.5}$$

since its model of functions does not include relative growth rates. Similarly, asymptotic behavior lies outside QP's ken; it assumes, simplemindedly, that quantities eventually reach the boundaries that they approach. This heuristic predicts qualitatively incorrect behavior since it confuses bounds with limits. For example, the hyperbolic function, $\frac{1}{x}$, decreases and is greater than -10 for $x > 0$—yet it never passes 0, let alone approaches -10. Even in cases where QP's heuristic yields qualitatively correct results, such as the heat flow problem of section 3.2.2, it fails to predict how close to its limit a quantity will be at any given time. All in all, QP can not incorporate qualitative mathematical information into the existing constraint/simulation formalism without a detailed model of continuous functions. De Kleer and Brown [7] point out that many expert systems fail when given simplified versions of problems which they have already solved. QP suffers from the dual

of this weakness; it can not produce better solutions from more precise problem specifications.

4.3 Advantages of QM

QM's functional model removes the limitations that prevent QP from becoming an expert network analyst, without sacrificing its flexibility and generality, by stressing the aspects of expertise that it ignores: compiled knowledge, hierarchical abstraction and domain specific methods. QM encodes recurrent functions as FD's, and families of functions as instantiation algorithms, such as *i-exponential* for the exponential model, $ae^{bx}+c$, of decay and growth. ANALYZE[4] divides a network into sub-systems, connected by functional composition, addition or multiplication links, creates an FD for each sub-system, and uses composition algorithms to derive an overall FD. Sub-systems can be analyzed by recursive decomposition or by instantiation algorithms; unlike QP, this algorithm is fully hierarchical, since it uses a uniform representation, the FD, for all inputs and outputs. Finally, the FD model can record a wide range of information: numerical functions such as $\log x$, parameterized ones such as $\sin ax$, and purely qualitative ones such as "an increasing function". This allows QM to apply numerical techniques to the first type, symbolic ones to the second, and general functional ones to the third. QM takes advantage of powerful calculus methods whenever possible, but uses QP-style ones when all else fails.

As stated in chapter 2, QM is not a complete qualitative reasoner—just the mathematical model for one; it can describe, analyze and combine FD's, not derive them from networks. For the moment, no truly qualitative QM based reasoner exists. The only approximation thereof, QR, is purely algebraic and limited to networks with closed-form solutions. However, regardless of this restriction, which is discussed in chapter 5 in detail, almost all examples from current QP systems—and many that exceed their capabilities—can be analyzed. QM augments Kuipers's [13,16] description of a ball's flight with precise values for heights, velocities and accelerations. It does the same for his heat flow example [13,18] and *recognizes* the asymptote at infinity. QP can not determine that the first spring example will oscillate without damping and the second, with damping. In fact, it does not even realize that oscillation must occur, only that it *might*. Here too, QM produces a qualitative description of the spring's behavior, augmented with precise symbolic values for interesting quantities.

[4] See figure 2.5 on page 13.

5. Summary and Future Work

This thesis describes my qualitative mathematical reasoner, QMR, and compares it with existing QP style reasoners. QMR consists of a well developed qualitative mathematics system, QM, that manipulates piecewise continuous parameterized functions, and a rudimentary qualitative reasoner, QR, that deduces the behavior of functional networks. QM's mathematical sophistication allows expert style reasoning about systems of known functional form; also, its uniform representation, the FD, facilitates hierarchical decomposition of compound systems. In contrast, the QP paradigm attempts to build a sophisticated qualitative reasoner while relying on an extremely simple uniform model of functions. Existing QP programs use qualitative simulation, also called perturbation analysis, as their reasoning algorithm. This leads to combinatorial explosion in complex networks since components can not be treated as compound quantities; the former have history descriptions and the latter, constraints. Even were this limitation to be surmounted, QP's model of functions would remain too weak for precise mathematical reasoning.

Though QR solves many interesting problems, including most examples from current QP systems, quickly and precisely, its current capabilities are inadequate for expert reasoning about realistic systems. Experts reason about large systems whose closed-form solutions are nonexistent or unwieldy; in addition, the exact functional relation between nodes may be unknown. As two QP examples reveal, QR fails under those conditions. Kuipers [15,19] analyzes Starling's Equilibrium, a physiological model containing functional constraints (explained in section 4.1.1) along with algebraic ones; also, De Kleer and Brown [7] analyze a fluid flow model that has no closed-form solution. Extending QR to complex and partially specified networks is a goal for future research. One possible approach would use approximation techniques, such as power series expansions, when closed-form solutions fail. It would apply theorems about differential equations to analyze partially specified systems; for example, a function satisfying $y' = M_0^-(y)$ approaches zero monotonically. In both cases, QM would serve as a function expert, creating and analyzing FD's for the expressions derived by QR. These analysis methods would formalize and justify the intuitions behind QP's algorithms by grounding them in mathematical theory. A second goal is to prove QR's worth by solving significant real-world problems, as opposed to contrived examples; this goal will also guide QR's development by setting a standard that it must meet.

References

[1]

Johan de Kleer.
Qualitative and Quantitative Knowledge in Classical Mechanics.
A.I. TR 352, M.I.T., 1979.

[2]

Johan de Kleer.
Causal and Teleological Reasoning in Circuit Recogintion.
PhD thesis, M.I.T, september, 1979.

[3]

Johan de Kleer and Daniel G. Bobrow.
Qualitative Reasoning with Higher-Order Derivatives.
In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence.*
IJCAI, august, 1984.

[4]

Johan de Kleer and John Seely Brown.
Mental Models of Physical Mechanisms and their Acquisition.
In J.R. Anderson, editor, *Cognitive Skills an their Acquisition.* Erlbaum, 1981.

[5]

Johan de Kleer and John Seely Brown.
Foundations of Envisioning.
Technical Report, XEROX PARC, august, 1982.
revised version of 1982 AAAI article

[6]

Johan de Kleer and John Seely Brown.
Assumptions and Ambiguities in Mechanistic Mental Models.
Technical Report, XEROX PARC, march, 1982.

[7]

Johan de Kleer and John Seely Brown.
A Qualitative Physics Based on Confluences.
to appear in *Artificial Intelligence*

[8]

Johan de Kleer and John Seely Brown.
The Origin, Form and Logic of Qualitative Physical Laws.
In *Proceedings of the Eigth International Joint Conference on Artificial Intelligence,*
pages 1158-1169. IJCAI, august, 1983.

[9]
Richard J. Doyle.
Representing Change for Common-Sense Physical Reasoning.
Working Paper 243, M.I.T., january, 1983.

[10]
Kenneth D. Forbus.
A Study of Qualitative and Geometric Knowledge in Reasoning about Motion.
A.I. TR 615, M.I.T., february, 1981.

[11]
Kenneth D. Forbus.
Qualitative Process Theory.
A.I. Memo 664, M.I.T., february, 1982.
A similar paper will appear in *Artificial Intelligence*

[12]
Kenneth D. Forbus.
Measurement Interpretation in Qualitative Process Theory.
In *Proceedings of the Eigth International Joint Conference on Artificial Intelligence*, pages 315-320. IJCAI, august, 1983.

[13]
Benjamin Kuipers.
Commonsense Reasoning about Causality: Deriving Behavior from Structure.
TUWPICS 18, Tufts University, may, 1982.

[14]
Benjamin Kuipers.
Getting the Envisionment Right.
In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 209-212. IJCAI, august, 1982.

[15]
Benjamin Kuipers.
Causal Reasoning in Medicine: Analysis of a Protocol.
TUWPICS 20, Tufts University, february, 1983.

[16]
Benjamin Kuipers.
Envisionment example #1: The Ball Envisionment.
Memo 9, Tufts University, october, 1983.
Draft

[17]
Benjamin Kuipers.
Envisionment example #2: The Spring Envisionment.
Memo 10, Tufts University, october, 1983.
Draft

[18]
Benjamin Kuipers.
Envisionment example #3: The Summary Heat Flow Envisionment.
Memo 11, Tufts University, october, 1983.
Draft

[19]
Benjamin Kuipers.
Envisionment example #4: The Starling Envisionment.
Memo 12, Tufts University, october, 1983.
Draft

[20]
Benjamin Kuipers.
Envisionment example #5: The Simple Heat Flow Envisionment.
Memo 13, Tufts University, october, 1983.
Draft

[21]
Benjamin Kuipers.
Envisionment example #6: The Double Heat Flow Envisionment and Simplification.
Memo 14, Tufts University, october, 1983.
Draft

[22]
Brian C. Williams.
Qualitative Analysis of MOS Circuits.
Master's thesis, M.I.T, 1984.
also to appear in *Artificial Intelligence*

[23]
Brian C. Williams.
The Use of Continuity in a Qualitative Physics.
In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence.*
IJCAI, august, 1984.

A. Instantiation Functions

These are the functions (of x) for which QM has instantiation functions.

1. $ax^3 + bx^2 + cx + d$
2. $ax^2 + bx + c$
3. $ax + b$
4. a
5. $|ax + b|$
6. $(bx + c)^p$
7. $(ax + b)c^{dx}$
8. b^{cx} $b \geq 0$
9. $a_1 b_1^{c_1 x} + a_2 b_2^{c_2 x}$
10. $\log_u(cx + d)$
11. $\dfrac{ax + b}{cx + d}$
12. $\sin(bx + c)$, $\cos(bx + c)$, $\tan(bx + c)$
13. $\arcsin(bx + c)$, $\arccos(bx + c)$, $\arctan(bx + c)$
14. $\sinh(bx + c)$, $\cosh(bx + c)$
15. $base^{bx}[k_1 \sin cx + k_2 \cos cx]$
16. $base^{bx} \sin(cx + d)$, $base^{bx} \cos(cx + d)$
17. $floor(bx + c)$, $ceiling(bx + c)$
18. The impulse function with low l, high h, and width w.
19. The impulse functions with linear/cubic rise time r, start t_0, width w, low value l, and high value h.
20. The triangle wave with period 2w, low value l, and peak h starting at t_0.

B. Description Queries

Description queries fall into two classes, *point* queries describe functional behavior at a point and *interval* queries describe behavior on an open, half open, or closed interval.

Point Descriptions

1. evaluate
2. evaluate inverse
3. left and right limits
4. limit
5. defined?
6. bounded?
7. continuous?
8. direction from the left and right
9. left and right derivatives
10. differentiable?
11. convexity

Interval Descriptions

1. extrema
2. maxima and minima
3. discontinuities
4. singularities
5. smooth?
6. asymptotes
7. *point* descriptions 5–7, 10 and 11 extended to intervals.

This blank page was inserted to preserve pagination.

CS-TR Scanning Project
Document Control Form

Date : 5/19/95

Report # LCS-TR-329

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 48 (56-IMAGES)
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☒ Unknown ☐ Other:_____

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☒ Cover Page
☒ Spine ☒ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP (1) UNN'ED TITLE PAGE	
(1-48) PAGES #'ED 1-48	
(50-57) SCANCONTROL, COVER, SPINE, PRINTERS NOTES	
(54-56) TRGTS (3)	

Scanning Agent Signoff:

Date Received: 5/19/95 Date Scanned: 6/1/95 Date Returned: 6/1/95

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 6/1/1995

M.I.T. Libraries
Document Services