QUALITATIVE MATHEMATICAL REASONING

Elisha Sacks

Clinical Decision Making Group, Laboratory for Computer Science
Massachusetts Institute of Technology, Cambridge, MA. 02139

ABSTRACT

Qualitative analysis studies the mechanisms with which humans derive the abstract behavior of a complex system from a description of its components and their interconnections, while ignoring unimportant or unknown low-level details. I present a program that analyzes a system by forming and combining *mathematical* descriptions of its sub-systems. This approach can overcome significant shortcomings in the simulation method used by other qualitative reasoning systems.

1. Motivation

I contend that qualitative reasoners should produce explicit mathematical descriptions of behavior. I have developed a qualitative mathematical rcasoner. QMR, to test my contention. It derives closed-form expressions for each parameter in a system and determines the behavior of these expressions. This presupposes that real-world parameters can be modeled by real-valued piecewise differentiate functions of time, and systems by sets of functionally related parameters. Actually, I only discuss systems that can be solved using Laplace Transforms and other basic techniques. Extending QMR to cases with unwieldy or open-form solutions is discussed in the conclusions.

2. An Example

Before delving into the details of QMR, let us consider a simple example, the behavior of a ball after being thrown straight up from height 0 with velocity vo- This system can be represented by the ball's height *h[t),* velocity *v(t)* and acceleration *a(t).* The constraints

$$\left.\begin{array}{l} v(t) = h'(t) \\ a(t) = v'(t) \\ a(t)\ constant \end{array}\right\} \text{imply} \left\{\begin{array}{l} h(t) = \frac{g}{2}t^2 + v_0 t \\ v(t) = gt + v_0 \\ a(t) = g \end{array}\right. \quad \text{(i)}$$

QMR interprets these results and produces the qualitative description appearing in table 1. This trivial example shows only a small fraction of QMR's power. My thesis [4] contains more extensive examples including damped oscillation and heat dissipation.

3. Data Structures

QMR represents a function on $\Re^1$ by a collection of interval descriptors, each corresponding to a closed sub-interval of its domain. The function is continuous and strictly monotonic or constant within these intervals, 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.

$$[lb, p_1],\ [p_1, p_2], \ldots, [p_{k-1}, p_k],\ [p_k, ub] \quad \text{(2)}$$

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

Each fun-int contains the information that high school students use to sketch functions. It includes the function's direction, singularities, inflection points, end-point values and limits, convexity. Values may bo undefined or unknown. The inverse, for instance, does not exist on constant intervals, and may lack a closed form on monotone intervals Table 2 shows the fun-ints for the ball's height *h.* Convexity is represented as a list of triples *(lb sign ub)* in which *sign* is the second derivatives sign on *(lb ub)*. All other entries arc self-explanatory.

Constraints and initial values mix numbers and unknown symbols, so QMR implements generic arithmetic operations similar to those in other symbolic algebra systems. Inequality relations between arbitrary expressions can be asserted, but inconsistent assertions are rejected. This information is used by generalized inequality predicates, for example

$$\left.\begin{array}{l} e_1 < v_1 \\ e_2 < v_2 \end{array}\right\} \text{implies } e_1 + e_2 < v_1 + v_2. \quad \text{(3)}$$

Table 1: Qualitative Description of a Ball's Flight

Time point 1: $t = 0$
h's value is 0. *v*'s value is v_0
Between points 1 and 2:
h increases and decelerates. *v* decreases and is linear.
Time point 2: $t = -v_0/g$
h reaches a maximum of $-v_0^2/2g$. *v*'s value is 0
Between points 2 and 3:
h decreases and accelerates. *v* decreases and is linear.
Time point 3: $t = -2v_0/g$
h's value is 0. *v*'s value is $-v_0$.
The description continues...

Table 2: The Ball's Fun-ints

	Interval 1	Interval 2
direction	up	down
fun	$\lambda t.\ v_0 t + \frac{g}{2}t^2$	$\lambda t.\ v_0 t + \frac{g}{2}t^2$
lb-val	0	$-\frac{v_0^2}{2g}$
convexity	$((0\ -1\ -\frac{v_0}{g}))$	$((-\frac{v_0}{g}\ -1\ -\frac{2v_0}{g}))$
lb	0	$-v_0/g$
lb-val	0	$-v_0^2/2g$
ub	$-v_0/g$	$-2v_0/g$
ub-val	$-v_0^2/2g$	0

The current algorithms do not derive all valid inequality relations, but they never derive invalid ones. They have been adequate so far. If they prove insufficient, more sophisticated algebraic and analytic techniques can be utilized.

4. Instantiation Algorithms

QMIt derives the behavior of expressions by constructing and describing their FD's. It matches expressions syntactically with a large library of basic patterns that includes cubic, exponential, logarithmic, hyperbolic and trigonometric functions. If a match succeeds the corresponding *instantiation algorithm* creates an FD based on the matched arguments. The qualitative region in which each argument lies determines the FD's form, whereas the actual arguments determine its entries. For example, $ax^2 + bx$ matches the quadratic pattern $?ux^2 + ?vx + ?w$ with $?u = a$ $?v = b$ and ?w - 0. The quadratic instantiator chooses a quadratic or a linear form, depending on whether *a* is nonzero or zero, and substitutes a. 6 and 0 into that form.

Instantiation algorithms use three transformations: restriction, linear substitution and linear composition, to produce a wide range of FD's from a few basic ones. All quadratics, for instance, come from the x^2 FD. Linear substitution produces an FD for $f(ax + b)$ from that of $f(x)$ by mapping each fun-int *[lb, ub* onto

$$\begin{cases} \left[\frac{lb}{a}, \frac{ub}{a}\right] & \text{if } a > 0 \\ \left[\frac{ub}{a}, \frac{lb}{a}\right] & \text{if } a < 0 \end{cases} \tag{4}$$

and appropriately scaling the derivatives, singularities and convexities. The first and second derivatives are $af'(ax + b)$ and $a^2f''(ax + b)$ by the chain rule. Singular points are scaled and their values multiplied by *a,* but convexity regions are just scaled since multiplying by a^2 leaves their qualitative 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, $f(x) = 6$. Finally, the restriction operator derives an FD for $f(x)$ restricted to a sub-interval of its original domain.

All three instantiation functions perform perfectly when certain qualitative information is available, for example *a's* sign for substitution and composition. If ambiguity exists, e.g. $a \leq 0$, they create an FD for every possible case and record the appropriate assumption in each one. The next section describes combination algorithms that are more powerful and general than the instantiation algorithms. However, unlike the instantiation functions, they do not always produce complete FD's due to the limitations of their current methods. As with symbolic arithmetic, more sophisticated methods arc unnecessary at the moment.

5. Combination Algorithms

Combination algorithms implement functional composition, addition and multiplication of arbitrary FD's. They build other operators, such as exponentiation, division and subtraction, from these three. They enable QMR to decompose complicated expressions that do not match any stored pattern into sums, products, exponentiations or compositions of two or more sub-expressions, to analyze recursively the sub-expressions and to combine the results.

The composition algorithm 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. 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. Each singularity of g's transfers to $f \circ g$ and each f singularity d maps to $g^{-1}(d)$. In both cases, left and right derivatives 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, in the worst case, by falling back on the derivative's definition and taking limits.

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. 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 their definitions when the sums are undefined. For example, $\lim_{x\to\infty} [e^{ax} - bx]$ with a and b positive cannot 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 $f + g$ cannot be determined by any general algorithm, so QMR uses special-purpose techniques. It determines the sum's direction directly when f and g have the same *direction*, one function is constant, or $(f + g)'$ is non-positive or non-negative on the interval. Otherwise, the derivative must change sign at some point p. QMR attempts to find p and analyze the two sub-intervals (lb, p) and (p, ub) recursively. If this fails—zeros of symbolic expressions don't always exist in closed form and sometimes cannot 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 heuristically that "very small" intervals have no turning points and 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 $(fg)' = f'g + fg'$ 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 is determined by an algorithm analogous to that used for sums.

6. Description Algorithms

Description algorithms derive FD's functional behavior, including directionality, convexity, extrema, discontinuities, limits, singularities and asymptotes, by straightforward mathematical means.

Higher-level descriptions can be constructed from the basic queries. One program summarizes an FD's *mathematical* behavior by listing directionality, discontinuities, convexity, singularities and turning points. Another generates the joint *qualitative* behavior of an FD set demonstrated in table 1. A graphics program that draws qualitative sketches of FD's also exists.

7. Comparison with Other Work

This section compares QMR with the qualitative simulation (QS) approach taken by Forbus [2], de Kleer and Brown [1], Kuipers [3] and Williams [5]. In QS. each parameter's instantaneous state consists of its qualitative value and the qualitative value of its derivative: increasing, constant or decreasing. A system's state, in turn, consists of the states of its parameters. QS derives a system's behavior from its constraints and initial values by determining the successive qualitative states which it might enter. States end when parameters reach boundary points, so QS must derive which, if any, do so from the constraints and initial values. It summarizes these results in a *history,* a graph whose nodes represent states and links, transitions. Periodic behavior shows up as cycles and ambiguity as multiple out-links.

Simulation offers insight into *naive* reasoning but ignores several aspects of *expert* behavior: large bodies of compiled knowledge, hierarchical abstraction and sophisticated mathematical models. Experts summarize important recurrent systems as *cliches,* concise descriptions of their behavior and appearance. Future problems that match cliches need not be analyzed since the expert remembers how they behave. Experts decompose a large system into a few sub-systems and treat them as black boxes that implement specified input/output behavior. These sub-systems, in turn, may be recognized as cliches or decomposed further, leading to a hierarchy of abstractions. In order to treat sub-systems abstractly, QS must replace each one by a high-level constraint that relates its inputs and outputs but ignores internal structure. However, sub-system behavior is described by histories, so QS must deduce constraints from histories. That difficult learning problem has not been mentioned—much less solved—by current QS researchers, so they must always use combinatorially explosive linear simulation. QS systems could learn and record cliches—though none actually do—but only as histories, not as constraints. Once again, these would not be very useful in problem decomposition.

Experts often derive and analyze mathematical models such as exponential decay, linear growth, and damped oscillation. QS lacks the necessary details for this analysis. For example, it cannot represent the functions

$$f(t) = e^{at} \text{ with } a > 0 \text{ or } g(t) = \sin(t) \qquad (5)$$

or deduce that $f(t) > g(t)$ for $t > 1/a$. Similarly, asymptotic behavior lies outside QS's ken. since it assumes, simplemindedly, that quantities eventually reach the boundaries that they approach. This heuristic can predict qualitatively incorrect behavior since it confuses bounds with limits. In general, QS cannot incorporate qualitative mathematical information into the existing constraint/simulation formalism without a detailed model of continuous functions. De Kleer and Brown point out that many expert systems fail when given simplified versions of problems which they have already solved. QS suffers from the dual of this weakness; it cannot produce better solutions from more precise problem specifications.

QMR's functional model removes the limitations that prevent QS 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 mathematical models. QMR encodes common functions as FD's, and families of functions as instantiation algorithms, such as the exponential model $ae^{bx} + c$ of decay and growth. It 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 QS. 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 QMR to apply numerical techniques to the first type, symbolic ones to the second, and general functional ones to the third. It takes advantage of powerful calculus methods whenever possible, but uses general ones when all else fails.

8. Conclusions

I have described a qualitative mathematical reasoner. QMR. and compared it with existing QS reasoners. QMR consists of a well-developed qualitative mathematics system that manipulates piecewise continuous parameterized functions, and a rudimentary qualitative reasoner that deduces the behavior of functional networks. Its mathematical sophistication allows expert reasoning about systems of known functional form, while its uniform representation, the FD. facilitates hierarchical decomposition of compound systems. In contrast, the QS paradigm attempts to build a sophisticated qualitative reasoner while relying on an extremely simple uniform model of functions. Existing QS programs use qualitative simulation as their reasoning algorithm. This leads to combinatorial explosion in complex networks since components cannot be treated as compound quantities; the former have history descriptions and the latter, constraints. Even if this limitation could be surmounted, the QS functional model would remain too weak for sophisticated mathematical reasoning techniques.

Though QMR solves many interesting problems 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, or where the exact functional relation between nodes is unknown. Extending QMR 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. A second goal is to prove QMR's worth by solving significant problems, not just simple examples. This goal will also guide QMR's development by setting a standard that it must meet.

9. References

[1] J. De Kleer, J. Brown, "A Qualitative Physics Based on Confluences." *Artificial Intelligence* 24(1-3). December 1984.

[2] K. Forbus. "Qualitative Process Theory." A.I. TR 789, M.I.T., July 1984.

|3] B. Kuipers. "Commonsense Reasoning about Causality," *Artificial Intelligence* 24(1-3). December 1984.

[4] E. Sacks. "Qualitative Mathematical Reasoning." LCS/TR-329. M.I.T.. 1985.