Non-Standard Analysis and Embedded Software

Richard Platek

Odyssey Research Associates
301 Dates Drive
Ithaca, NY 14850-1326
richard@oracorp.com

3rd NASA Formal Methods Workshop

May 12, 1995

Outline of Talk

- ❒ Embedded software
- ❒ Need for formal methods to include continuous mathematics
- ❒ A proposed research program based on non-standard analysis

Embedded Software

- Hidden ubiquitous computers -- The environment of the future
- Analogy with electric motors
  - Question: How many electric motors service you a day?
- Many of these computational engines will directly interact with the physical environment (conventional use of the term "Embedded Software").
- Basic specification and verification issues need to be expressed in terms of the physical environment.

"Real Space" Software

- ❒ Analogy with real time software

- ❒ Although real time software has a history going back to the earliest days of computer science it is still an active research area as to how to specify, implement and verify such software.

- ❒ Logics for reasoning about real time software use real time (i.e., the real numbers used to measure time) or at least a close approximation to it.

- ❒ Real space software needs analogous logics in order to be able to prove claims such as "This software moves this object to this place at this time".

Challenge to the Formal Methods Community

- Construct the necessary logical infrastructure to support the development and evaluation of such software

- More accurately:

  - Construct the necessary logical infrastructure to support the development and evaluation of such systems

Basic Stumbling Block

- The discrete/continuous dichotomy
- Traditional formal methods only exploited discrete mathematics (e.g., logic, set theory, algebra)
- This is also true of computer science which might add combinatorics, number theory, etc.
- Traditional physical and engineering disciplines rely on continuous math (e.g., calculus, analysis, topology).
- Very few technical people are comfortable in both cultures.

The Discrete/Continuous Dichotomy

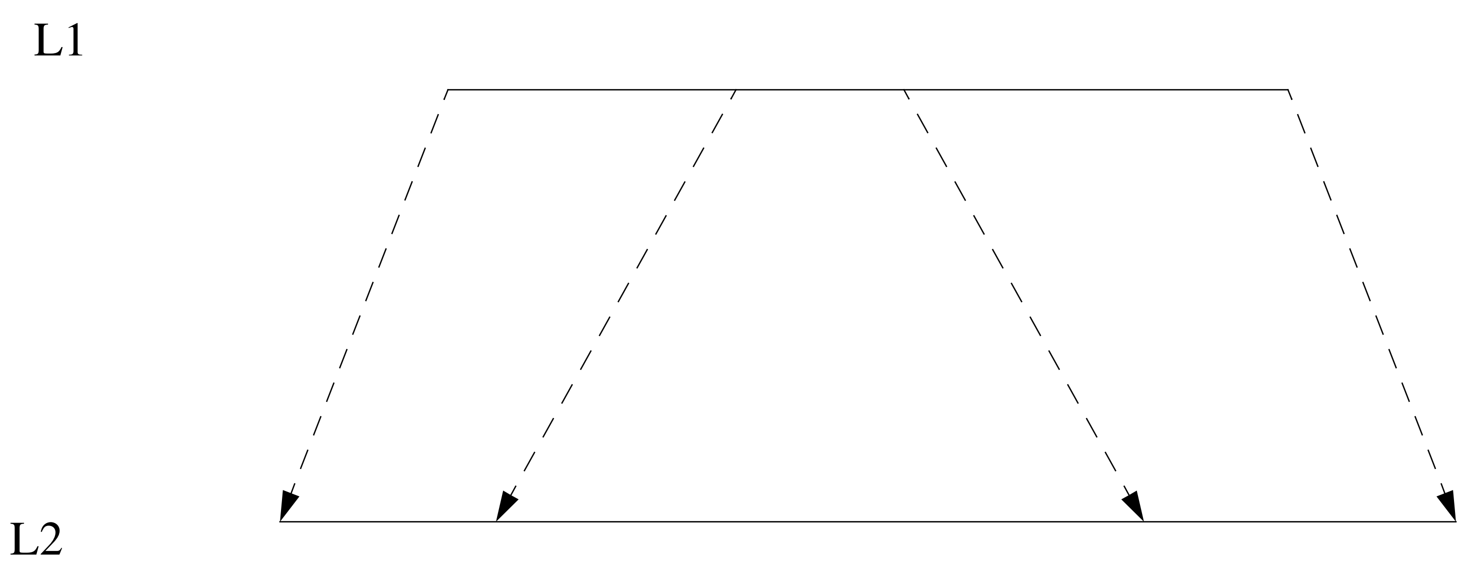
- ❒ In a discrete situation each individual has a distinct identity; wholes are simply collections of individuals.

- ❒ In a continuous situation there is no individual identity rather there is continuous flow or change.

The Paradoxes

- Classical attempts to discretize the continuous led to paradoxes
- When and How does a caterpillar become a butterfly?
- Zeno's paradoxes
  - The arrow
  - Achilles and the tortoise

The Paradoxes



There is a one to one correspondence between the points of L1 and L2. Each point has the same length. Where do the different lengths of the lines come from?

❒ Both lines have the same number of points; each point has the same length (viz., 0). Where does the length come from?

Quotations

- "No continuum can not be made up out of indivisibles, as for instance a line out of points, granting that the line is continuous and the point indivisible. " Aristotle
- "A point may not be a constituent part of a line"   Leibniz
- "A true continuum has no points"   Rene Thom

Non-Standard Analysis

- A modern, logically sound discretization of the continuous
- Based on Leibniz's Theory of Infinitesimals
- latter was used to develop all analysis during the 17th, 18th, and first half of the 19th century.
- It continued to be used in applied math and engineering until mid 20th century.
- Highly intuitive, very algebraic -- there are no limit arguments.

Leibniz' Infinitesimals

- Positive quantities smaller than all positive reals.
- They are not indivisible! If dx is an infinitesimal so is $\frac{dx}{2}$.
- The extended real numbers satisfy "all" the algebraic laws of the ordinary reals.
- Solves the paradoxes.
- Basis for an explosive development of new mathematics.

Examples

- Differentiation

$$\frac{(x+dx)^2 - x^2}{dx} = \frac{x^2 + 2 \times dx + (dx)^2 - x^2}{dx}$$

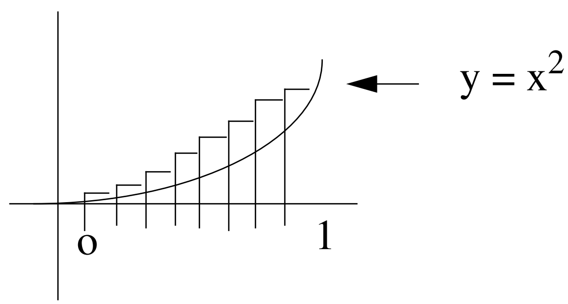
$$= 2x + dx$$

$$\approx 2x$$

No matter what choice of infintesimals dx

Examples

❒ Integration



Examples (cont'd)

$$\sum_{k=1}^{n} \left(\frac{k}{n}\right)^2 \cdot \frac{1}{n} = \frac{1}{n^3} \sum_{k=1}^{n} k^2$$

$$= \frac{1}{n^3} \left(\frac{n(n+1)(2n+1)}{6}\right)$$

$$= \frac{1}{n^3} \left(\frac{2n^3 + 3n^2 + n}{6}\right)$$

$$= \frac{1}{n^3} \left(\frac{\left(n^2 + n\right)(2n+1)}{6}\right)$$

$$= \frac{1}{3} + \frac{1}{2n} + \frac{1}{6n^2}$$

Let n be infinite $\approx \frac{1}{3}$ independent of n.

Examples

- Series

$$cos\ \ n\theta + i\ \ \sin\ \ n\theta\ =\ (cos\theta + i\ \ sin\theta)^n$$

$$= \sum_{k=0}^{n} \binom{n}{k} i^k sin^k\theta\ \ cos^{n-k}\theta$$

$$\sum_{k=0}^{n} \frac{n!}{k!\,(n-k)\,!} i^k sin^k\theta\ \ cos^{n-k}\theta$$

Examples (cont'd)

Given a finite angle x let n be infinite and $\theta = \frac{x}{n}$ be infinitesimal

Then

$$\frac{n!}{k!\,(n-k)\,!} \approx \frac{n^k}{k!}$$

$$\sin\theta \approx \theta$$

$$\sin^k\theta \approx \theta^k$$

$$cos\theta \approx 1$$

$$cos^{n-k}\theta \approx 1$$

$$cos x + i \sin\ x \approx \sum_{k=0}^{\infty} i^k \left(\frac{x^k}{k!}\right)$$

Applications to Embedded Software

- ❐ Add extended reals *R and extended integers *Z as types to any given program language together with the usual arithmetic and logical operators over these types.
- ❐ Extend the notion of computation to include possibly infinite computations to compute integrals and series.
- ❐ Prove that a program satisfies a given spec or generate (synthesize) the program from the spec.
- ❐ Note that computed results are not equal to the desired number they are only infinitesimally close (as in the above examples).

Applications to Embedded Software (con't)

- ❒ Analyze the problem as presented in physical terms to determine a finite epsilon such that epsilon approximate solution would be adequate.
- ❒ Using this epsilon transform the program over *R which is infinitesmally close to the correct solution to an ordinary program which is epsilon close. How to do this is still a research topic.
- ❒
- ❒ The resulting program will be Real Space correct but not Real Time correct since no attention was paid to efficiency of computation. It remains to investigate how to integrate these two requirements.

Basic Computer Science Question

- ❐ In the extended programming language described above what functions over *R and *Z are computable? Can one give a description of these functions independent of the starting programming language? Is there a natural recursion theory over *R? Currently being worked on using results of generalized recursion theory introduced 30 years ago by Kleene, Gandy, and Platek,