Efficient Computation with Dedekind Reals

Andrej Bauer
(joint work with Paul Taylor)

Department of Mathematics and Physics
University of Ljubljana, Slovenia

MAP, Trieste, August 2008

In this talk

We present a mathematical language which is powerful enough to let us talk about real analysis, but also simple enough to be an efficient programming language.

Foundations: Abstract Stone Duality

- Our language is based on *Abstract Stone Duality* (ASD) by Paul Taylor.
- ASD is a variant of λ-calculus which directly axiomatizes spaces and continuous maps.
- We use a fragment of ASD which can be understood on its own.
- Further material: `http://www.paultaylor.eu/ASD/`

A language for real analysis

- Number types $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$
- Arithmetic $+$, $-$, $\times$, $/$
- Decidable equality $=$ and decidable order $<$ on $\mathbb{N}$ and $\mathbb{Q}$
- General recursion on $\mathbb{N}$
- Semidecidable order relation $<$ on $\mathbb{R}$
- Logic:
  - truth $\top$ and falsehood $\bot$
  - connectives $\wedge$ and $\vee$
  - existential quantifiers:

    $$\exists x : \mathbb{R}, \quad \exists x : [a, b], \quad \exists x : (a, b), \quad \exists n : \mathbb{N}, \quad \exists q : \mathbb{Q}$$

  - universal quantifier: $\forall x : [a, b]$

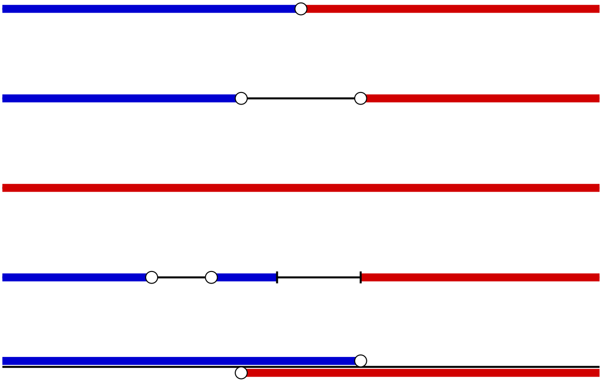
Axioms for real numbers

The real numbers $\mathbb{R}$ are:

- an ordered field,
- with Archimedean property,
- Dedekind complete,
- overt Hausdorff space,
- and $[0, 1]$ is compact.

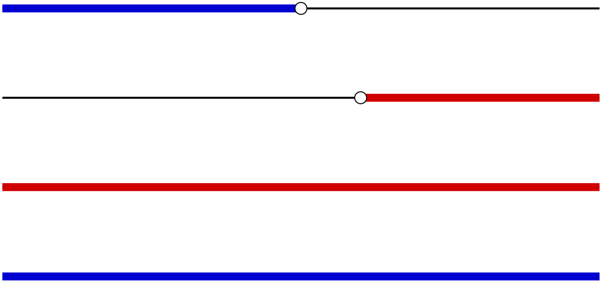
Dedekind cuts

A *cut* is a pair of *rounded*, *bounded*, *disjoint*, and *located* open sets.



Lower and upper reals

By taking the lower rounded sets we obtain the *lower reals*, and similarly for *upper reals*. These are more fundamental than reals.



Examples of cuts

- A number a determines a cut, which determines a:

$$a = \mathsf{cut}\ x\ \mathsf{left}\ x < a\ \mathsf{right}\ a < x$$

- $\sqrt{a}$ is the cut

$$\mathsf{cut}\ x\ \mathsf{left}\ (x < 0 \vee x^2 < a)\ \mathsf{right}\ (x > 0 \wedge x^2 > a)$$

- Exercise:

$$\mathsf{cut}\ x\ \mathsf{left}\ (x < -a \vee x < a)\ \mathsf{right}\ (-a < x \wedge a < x)$$

- The full notation for cuts is

$$\mathsf{cut}\ x : [a, b]\ \mathsf{left}\ \phi(x)\ \mathsf{right}\ \psi(x)$$

  This means that the cut determines a number in $[a, b]$.

“Topologic”

- A logical formula $\phi(x)$ where $x : A$ has two readings:
  - *logical*: a predicate on A
  - *topological*: an open subset of A
- In particular, a closed formula ϕ is
  - *logically*, a truth value
  - *topologically*, an element of Sierpinski space Σ
- We use this to express topological and analytic notions *logically*.

Example: $\mathbb{R}$ is locally compact

- Classically: for open $U \subseteq \mathbb{R}$ and $x \in \mathbb{R}$,

$$x \in U \iff \exists\, d, u \in \mathbb{Q} \,.\, x \in (d, u) \subseteq [d, u] \subseteq U$$

- Topologically: for $\phi : \mathbb{R} \to \Sigma$ and $x : \mathbb{R}$,

$$\phi(x) \iff \exists\, d, u \in \mathbb{Q} \,.\, d < x < u \wedge \forall\, y \in [d, u] \,.\, \phi(y)$$

Example: $[0,1]$ is connected

- Classically: for open $U, V \subseteq [0,1]$,

$$U \cap V = \emptyset \wedge U \cup V = [0,1] \implies U = [0,1] \vee V = [0,1]$$

- (Topo)logically: for $\phi, \psi : [0,1] \to \Sigma$, if

$$\perp \iff \phi(x) \wedge \psi(x)$$

then

$$\forall\, x \in [0,1] \,.\, (\phi(x) \vee \psi(x)) \implies \\ (\forall\, x \in [0,1] \,.\, \phi(x)) \vee (\forall\, x \in [0,1] \,.\, \psi(x))$$

Example: $\mathbb{R}$ is connected

- Classically: for open $U, V \subseteq \mathbb{R}$,

$$U \cup V = \mathbb{R} \wedge U \neq \emptyset \wedge V \neq \emptyset \implies U \cap V \neq \emptyset$$

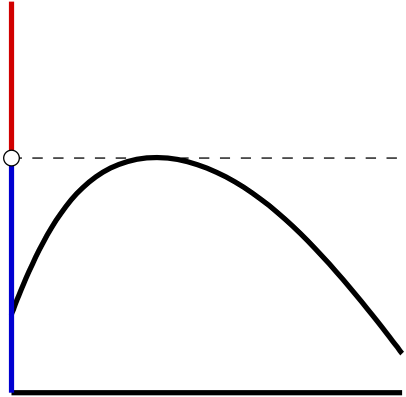
- (Topo)logically: for $\phi, \psi : \mathbb{R} \to \Sigma$, if

$$\top \iff \phi(x) \vee \psi(x)$$

  then

$$(\exists\, x \in \mathbb{R} \,.\, \phi(x)) \wedge (\exists\, x \in \mathbb{R} \,.\, \psi(x)) \implies \exists\, x \in \mathbb{R} \,.\, \phi(x) \wedge \psi(x).$$

The maximum of $f : [0,1] \to \mathbb{R}$



$$\text{cut } x \;\; \text{left } (\exists\, y \in [0,1]\,.\, x < f(y)) \\ \text{right } (\forall\, z \in [0,1]\,.\, f(z) < x)$$

Cauchy completeness

- A *rapid* Cauchy sequence $(a_n)_n$ satisfies

$$|a_{n+1} - a_n| < 2^{-n}.$$

- Its limit is the cut

$$\begin{aligned} \mathsf{cut}\ x\ \ \mathsf{left}\ (\exists\, n \in \mathbb{N}\,.\, x < a_n - 2^{-n+1}) \\ \mathsf{right}\ (\exists\, n \in \mathbb{N}\,.\, a_n + 2^{-n+1} < x) \end{aligned}$$

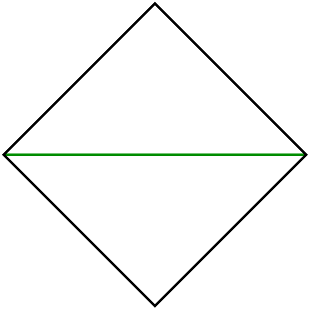
From mathematics to programming

- We would like to *compute* with our language.
- We limit attention to logic and $\mathbb{R}$, and leave recursion and $\mathbb{N}$ for future work.
- Not surprisingly, we compute with intervals.
- The prototype is written in OCaml and uses the MPFR library for fast dyadic rationals.

The interval lattice L

- The lattice of pairs $[a, b]$, where a is *upper* and b *lower real*.
- Ordered by $[a, b] \sqsubseteq [c, d] \iff a \leq c \wedge d \leq b$.
- The lattice contains $\mathbb{R}$.

$[\infty, -\infty]$



$[-\infty, \infty]$

Extending arithmetic to L

- We extend arithmetic operations from $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$ to $L \times L \to L$.
- The interesting case is *Kaucher multiplication*.
- Given an arithmetical expression e we compute its *lower* and *upper* approximants e^- and e^+ in L:

$$e^- \sqsubseteq e \sqsubseteq e^+.$$

- We also extend $<$ to $L \times L \to \Sigma$:

$$[a, b] < [c, d] \iff b < c$$

Lower and upper approximants

- For each sentence ϕ we define a *lower* and *upper approximants* $\phi^-, \phi^+ \in \{\top, \bot\}$ such that

$$\phi^- \implies \phi \implies \phi^+.$$

- The approximants should be easy to compute.
- If $\phi^- = \top$ then $\phi = \top$, and if $\phi^+ = \bot$ then $\phi = \bot$.
- Easy cases:

$$\begin{aligned}
\bot^- &= \bot & \bot^+ &= \bot \\
\top^- &= \top & \top^+ &= \top \\
(\phi \wedge \psi)^- &= \phi^- \wedge \psi^- & (\phi \wedge \psi)^+ &= \phi^+ \wedge \psi^+ \\
(\phi \vee \psi)^- &= \phi^- \vee \psi^- & (\phi \vee \psi)^+ &= \phi^+ \vee \psi^+ \\
(e_1 < e_2)^- &= (e_1^- < e_2^-) & (e_1 < e_2)^+ &= (e_1^+ < e_2^+).
\end{aligned}$$

Approximants for cuts and quantifiers

- Cuts:

$$(\mathsf{cut}\ x : [a,b]\ \mathsf{left}\ \phi(x)\ \mathsf{right}\ \psi(x))^- = [a,b]$$
$$(\mathsf{cut}\ x : [a,b]\ \mathsf{left}\ \phi(x)\ \mathsf{right}\ \psi(x))^+ = [b,a]$$

- Quantifiers:

$$\phi([a,b]) \quad\Longrightarrow\quad \forall\, x \in [a,b]\,.\,\phi(x) \quad\Longrightarrow\quad \phi(\tfrac{a+b}{2})$$

$$\phi(\tfrac{a+b}{2}) \quad\Longrightarrow\quad \exists\, x \in [a,b]\,.\,\phi(x) \quad\Longrightarrow\quad \phi([b,a])$$

Refinement

- If $\phi^- = \bot$ and $\phi^+ = \top$ we cannot say much about ϕ.
- To make progress, we *refine* ϕ to an equivalent formula in which quantifers range over smaller intervals.
- A simple strategy is to split quantified intervals in halves:
  - $\forall\, x \in [a, b] \,.\, \phi(x)$ is refined to

$$(\forall\, x \in [a, \tfrac{a+b}{2}] \,.\, \phi(x)) \wedge (\forall\, x \in [\tfrac{a+b}{2}, b] \,.\, \phi(x))$$

  - $\exists\, x \in [a, b] \,.\, \phi(x)$ is refined to

$$(\exists\, x \in [a, \tfrac{a+b}{2}] \,.\, \phi(x)) \vee (\exists\, x \in [\tfrac{a+b}{2}, b] \,.\, \phi(x))$$

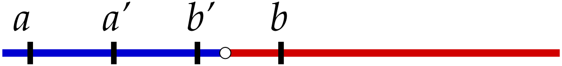
- This amounts to searching with *bisection*.

Refinement of cuts

- To refine a cut

$$\text{cut } x : [a, b] \text{ left } \phi(x) \text{ right } \psi(x)$$

  we try to move $a \mapsto a'$ and $b \mapsto b'$.



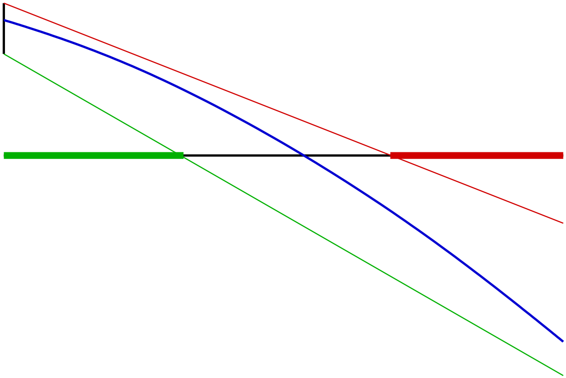
- If $\phi^-(a') = \top$ then move $a \mapsto a'$.
- If $\psi^-(b') = \top$ then move $b \mapsto b'$.
- One or the other endpoint moves eventually because cuts are located.

Evaluation

- To evaluate a sentence ϕ:
  - if $\phi^- = \top$ then output $\top$,
  - if $\phi^+ = \bot$ then output $\bot$,
  - otherwise refine ϕ and repeat.
- Evaluation may not terminate, but this is expected, as ϕ is only *semi*decidable.
- Is the procedure *semicomplete*, i.e., if ASD proves ϕ then ϕ evaluates to $\top$?

Speeding up the computation

Estimate an inequality $f(x) < 0$ on $[a, b]$ by approximating f with a linear map from above and below.



This is essentially Newton's interval method.

Future

- Incorporate $\mathbb{N}$ and recursion.
- Extend Newton's method to multivariate case.
- Write a more efficient interpreter.
- Can we do higher-type computations?
- Can this be implemented as a *library* for a standard language, rather than a specialized language?