Programming Techniques for Exact Real Arithmetic

Andrej Bauer
Faculty of Mathematics and Physics
University of Ljubljana, Slovenia

(joint work with Ivo List & Paul Taylor)

SCAN 2014, Würzburg, September 2014
In this talk

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

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

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

- Descriptive language – what to compute.
In this talk

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

- Descriptive language – what to compute.
- How to compute?
In this talk

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

- Descriptive language – what to compute.
- How to compute?
  - We present a simple-minded execution strategy
In this talk

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

- *Descriptive language – what to compute.*
- *How to compute?*
  - We present a simple-minded execution strategy
  - Many possibilities for optimization
In this talk

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

- Descriptive language – *what* to compute.
- *How* to compute?
  - We present a simple-minded execution strategy
  - Many possibilities for optimization
- Applications:
In this talk

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

- **Descriptive language – what to compute.**
- **How to compute?**
  - We present a simple-minded execution strategy
  - Many possibilities for optimization
- **Applications:**
  - specification of computational problems
In this talk

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

- Descriptive language – what to compute.
- How to compute?
  - We present a simple-minded execution strategy
  - Many possibilities for optimization
- Applications:
  - specification of computational problems
  - verification of safety and liveness properties
Our language is based on *Abstract Stone Duality* (ASD) by Paul Taylor.
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.
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.
Our language is based on *Abstract Stone Duality* (ASD) by Paul Taylor.

ASD is a variant of $\lambda$-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/](http://www.paultaylor.eu/ASD/)
Axioms for real numbers

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

- an ordered field ("can compute with reals")

- with Archimedean property ("can obtain approximations")

- Dedekind complete ("can use iterative methods")

- overt Hausdorff space ("can search for a witness")

- and $[a, b]$ is compact ("can verify something holds")
Axioms for real numbers

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

- an ordered field ("can compute with reals")
- with Archimedean property ("can obtain approximations")
Axioms for real numbers

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

- an ordered field ("can compute with reals")
- with Archimedean property ("can obtain approximations")
- Dedekind complete ("can use iterative methods")
Axioms for real numbers

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

- an ordered field (“can compute with reals”)
- with Archimedean property (“can obtain approximations”)
- Dedekind complete (“can use iterative methods”)
- overt Hausdorff space (“can search for a witness”)

Axioms for real numbers

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

- an ordered field (“can compute with reals”)
- with Archimedean property (“can obtain approximations”)
- Dedekind complete (“can use iterative methods”)
- overt Hausdorff space (“can search for a witness”)
- and $[a, b]$ is compact (“can verify something holds”)
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 $a$:

  $$a = (\text{cut } x \text{ left } x < a \text{ right } a < x)$$
Examples of cuts

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

$$a = (\text{cut } x \text{ left } x < a \text{ right } a < x)$$

- $\sqrt{a}$ is the cut

$$\text{cut } x \text{ left } (x < 0 \lor x^2 < a) \text{ right } (x > 0 \land x^2 > a)$$
Examples of cuts

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

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

- $\sqrt{a}$ is the cut

$$\text{cut } x \text{ left } (x < 0 \lor x^2 < a) \text{ right } (x > 0 \land x^2 > a)$$

- Exercise:

$$\text{cut } x \text{ left } (x < -a \lor x < a) \text{ right } (-a < x \land a < x)$$
Examples of cuts

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

  $$a = (\text{cut } x \text{ left } x < a \text{ right } a < x)$$

- $\sqrt{a}$ is the cut

  $$\text{cut } x \text{ left } (x < 0 \lor x^2 < a) \text{ right } (x > 0 \land x^2 > a)$$

- Exercise:

  $$\text{cut } x \text{ left } (x < -a \lor x < a) \text{ right } (-a < x \land a < x)$$

- The full notation for cuts is

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

  This means that the cut determines a number in $[a, b]$.
A language for real analysis

- Number types $\mathbb{N}, \mathbb{Q}, \mathbb{R}$
A language for real analysis

- Number types $\mathbb{N}, \mathbb{Q}, \mathbb{R}$
- Arithmetic $+, -, \times, /$
A language for real analysis

- Number types $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$
- Arithmetic $+,-,\times,/\,$
- Decidable equality $=$ and decidable order $<$ on $\mathbb{N}$ and $\mathbb{Q}$
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 $\mathbb{N}$
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 $\mathbb{R}$
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:
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 $\bot$
A language for real analysis

- Number types $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$
- Arithmetic $+$, $-$, $\times$, $\div$
- 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 $\land$ and $\lor$
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 $\land$ and $\lor$
  - existential quantifiers:
    - $\exists x : \mathbb{R}$, $\exists x : [a, b]$, $\exists x : (a, b)$, $\exists n : \mathbb{N}$, $\exists q : \mathbb{Q}$
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 \(\land\) and \(\lor\)
  - 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: \(\forall x : [a, b]\)
A logical formula $\phi(x)$ where $x \in A$ has two readings:

- Logical: a predicate on $A$
- Topological: an open subset of $A$ (a closed subset of $A$ in the case of Sierpinski space $\mathcal{S} = \{?, >\}$).

We use this to express topological and analytic notions logically.
A logical formula $\phi(x)$ where $x \in A$ has two readings:

- *logical*: a predicate on $A$
- *topological*: an open subset of $A$
- *analytic*: a closed subset of $A$

In particular, a formula without parameters is logically, a truth value topologically, an element of Sierpinski space $\mathcal{S} = \{?, >\}$.

We use this to express topological and analytic notions logically.
A logical formula $\phi(x)$ where $x \in A$ has two readings:

- **logical**: a predicate on $A$
- **topological**:
“Topologic”

- A logical formula $\phi(x)$ where $x \in A$ has two readings:
  - *logical*: a predicate on $A$
  - *topological*:
    - an open subset of $A$: $\phi(x) \iff \top$
A logical formula $\phi(x)$ where $x \in A$ has two readings:

- **logical**: a predicate on $A$
- **topological**:
  - an open subset of $A$: $\phi(x) \iff \top$
  - a closed subset of $A$: $\phi(x) \iff \bot$
A logical formula $\phi(x)$ where $x \in A$ has two readings:

- **logical**: a predicate on $A$
- **topological**:
  - an open subset of $A$: $\phi(x) \iff \top$
  - a closed subset of $A$: $\phi(x) \iff \bot$

In particular, a formula $\phi$ without parameters is
A logical formula $\phi(x)$ where $x \in A$ has two readings:

- *logical*: a predicate on $A$
- *topological*:
  - an open subset of $A$: $\phi(x) \iff \top$
  - a closed subset of $A$: $\phi(x) \iff \bot$

In particular, a formula $\phi$ without parameters is

- *logically*, a truth value
A logical formula $\phi(x)$ where $x \in A$ has two readings:

- **logical**: a predicate on $A$
- **topological**:
  - an open subset of $A$: $\phi(x) \iff \top$
  - a closed subset of $A$: $\phi(x) \iff \bot$

In particular, a formula $\phi$ without parameters is

- **logically**, a truth value
- **topologically**, an element of Sierpinski space $\Sigma = \{\bot, \top\}$
A logical formula $\phi(x)$ where $x \in A$ has two readings:

- **logical**: a predicate on $A$
- **topological**:
  - an open subset of $A$: $\phi(x) \iff \top$
  - a closed subset of $A$: $\phi(x) \iff \bot$

In particular, a formula $\phi$ without parameters is

- **logically**, a truth value
- **topologically**, an element of Sierpinski space $\Sigma = \{\bot, \top\}$

We use this to express topological and analytic notions logically.
Example: $\mathbb{R}$ is locally compact

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

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

\[
\phi(x) \iff \exists d, u \in \mathbb{Q} . d < x < u \land \forall y \in [d, u] . \phi(y)
\]
Example: $[0, 1]$ is connected

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

  $$U \cap V = \emptyset \quad \text{and} \quad U \cup V = [0, 1]$$

  then $U = [0, 1]$ or $V = [0, 1]$. 
Example: $[0, 1]$ is connected

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

  $$U \cap V = \emptyset \quad \text{and} \quad U \cup V = [0, 1]$$

  then $U = [0, 1]$ or $V = [0, 1]$.

- Topologically: for $\phi, \psi : [0, 1] \to \Sigma$, if

  $$(\exists x \in [0, 1]. \phi(x) \land \psi(x)) \iff \bot \quad \text{and} \quad (\forall x \in [0, 1]. \phi(x) \lor \psi(x)) \iff \top$$

  then $(\forall x \in [0, 1]. \phi(x)) \lor (\forall x \in [0, 1]. \psi(x))$. 
For every parameter $x$ there is solution $y$.

In every state $x$ good thing $y$ happens.

Note: $A$ must be overt and $B$ compact.
\begin{align*}
\forall x \in A \ . \ \exists y \in B \ . \ \phi(x, y)
\end{align*}

- “For every parameter $x$ there is solution $y$."

\textit{Note:} A must be overt and B compact.
∀∃ statements

∀x ∈ A . ∃y ∈ B . \( \phi(x, y) \)

- “For every parameter \( x \) there is solution \( y \).”
- “In every state \( x \) good thing \( y \) happens.”
$\forall x \in A . \exists y \in B . \phi(x, y)$

- “For every parameter $x$ there is solution $y$.”
- “In every state $x$ good thing $y$ happens.”
- Note: $A$ must be *overt* and $B$ *compact*. 
The maximum of $f : [0, 1] \rightarrow \mathbb{R}$

\[
\begin{align*}
\text{cut } x \text{ left } & (\exists y \in [0, 1]. x < f(y)) \\
\text{right } & (\forall z \in [0, 1]. f(z) < x)
\end{align*}
\]
Cauchy completeness

- A rapid Cauchy sequence \((a_n)_n\) satisfies

\[ |a_{n+1} - a_n| < 2^{-n}. \]
A rapid Cauchy sequence \((a_n)_n\) satisfies

\[|a_{n+1} - a_n| < 2^{-n}.\]

Its limit is the cut

\[\text{cut } x \text{ left } (\exists n \in \mathbb{N}. x < a_n - 2^{-n+1})\]

\[\text{right } (\exists n \in \mathbb{N}. a_n + 2^{-n+1} < x)\]
From mathematics to programming

- We would like to *compute* with our language.
We would like to *compute* with our language.

- We limit attention to logic and \( \mathbb{R} \).
From mathematics to programming

- We would like to compute with our language.
- We limit attention to logic and $\mathbb{R}$.
- Not surprisingly, we compute with (improper) intervals.
The interval lattice $L$

- The lattice of **pairs** $[a, b]$, where $a$ is upper and $b$ lower real.
The interval lattice $L$

- The lattice of pairs $[a, b]$, where $a$ is upper and $b$ lower real.
- Ordered by $[a, b] \sqsubseteq [c, d] \iff a \leq c \land d \leq b$. 

[Diagram of the interval lattice $L$ with the intervals $[\infty, -\infty]$ and $[-\infty, \infty]$ shown.]
The interval lattice $L$

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

![Diagram of the interval lattice $L$]

The interval lattice $L$ is a lattice that consists of pairs of real numbers $[a, b]$, where $a$ is the upper and $b$ is the lower bound. The lattice is ordered by $[a, b] \subseteq [c, d] \iff a \leq c \land d \leq b$. The lattice contains the real numbers as the interval $[a, a]$.
Extending arithmetic to $L$

- Extend operations from $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ to $L \times L \rightarrow L$.
Extending arithmetic to $L$

- Extend operations from $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ to $L \times L \rightarrow L$:
  - $L$ is equipped with the Scott topology
Extending arithmetic to $L$

- Extend operations from $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$ to $L \times L \to L$:
  - $L$ is equipped with the Scott topology
  - *any* continuous extension is acceptable
Extending arithmetic to $L$

- Extend operations from $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ to $L \times L \rightarrow L$:
  - $L$ is equipped with the Scott topology
  - *any* continuous extension is acceptable
  - (improper) intervals are understood *order-theoretically*
Extending arithmetic to $L$

- Extend operations from $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$ to $L \times L \to L$:
  - $L$ is equipped with the Scott topology
  - *any* continuous extension is acceptable
  - (improper) intervals are understood *order-theoretically*
- The interesting case is *Kaucher multiplication*. 
Extending arithmetic to $L$

- Extend operations from $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$ to $L \times L \to L$:
  - $L$ is equipped with the Scott topology
  - *any* continuous extension is acceptable
  - (improper) intervals are understood order-theoretically
- The interesting case is *Kaucher multiplication*.
- Given an arithmetical expression $e$ we compute its *lower* and *upper* approximants $e^-$ and $e^+$ in $L$:

  $$ e^- \sqsubseteq e \sqsubseteq e^+ $$
Extending arithmetic to $L$

- Extend operations from $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ to $L \times L \rightarrow L$:
  - $L$ is equipped with the Scott topology
  - *any* continuous extension is acceptable
  - (improper) intervals are understood *order-theoretically*

- The interesting case is *Kaucher multiplication*.

- Given an arithmetical expression $e$ we compute its *lower* and *upper* approximants $e^-$ and $e^+$ in $L$:

  $$e^- \subset e \subset e^+.$$ 

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

  $$[a, b] < [c, d] \iff b < c$$
For each sentence $\phi$ we define a lower and upper approximants $\phi^-, \phi^+ \in \{\top, \bot\}$ such that

$$\phi^- \quad \Rightarrow \quad \phi \quad \Rightarrow \quad \phi^+.$$
Lower and upper approximants

- For each sentence \( \phi \) 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.
Lower and upper approximants

For each sentence $\phi$ 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 $\phi = \bot$. 

18 / 24
Lower and upper approximants

- For each sentence $\phi$ 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:

\[
\begin{align*}
\bot^- &= \bot & \bot^+ &= \bot \\
\top^- &= \top & \top^+ &= \top \\
(\phi \land \psi)^- &= \phi^- \land \psi^- & (\phi \land \psi)^+ &= \phi^+ \land \psi^+ \\
(\phi \lor \psi)^- &= \phi^- \lor \psi^- & (\phi \lor \psi)^+ &= \phi^+ \lor \psi^+ \\
(e_1 < e_2)^- &= (e_1^- < e_2^-) & (e_1 < e_2)^+ &= (e_1^+ < e_2^+).
\end{align*}
\]
Approximants for cuts and quantifiers

- Cuts:

\[(\text{cut } x : [a, b] \text{ left } \phi(x) \text{ right } \psi(x))^- = [a, b]\]
\[(\text{cut } x : [a, b] \text{ left } \phi(x) \text{ right } \psi(x))^+ = [b, a]\]
Approximants for cuts and quantifiers

- **Cuts:**

\[
\begin{align*}
\text{(cut } x : [a, b] \text{ left } \phi(x) \text{ right } \psi(x))^- &= [a, b] \\
\text{(cut } x : [a, b] \text{ left } \phi(x) \text{ right } \psi(x))^+ &= [b, a]
\end{align*}
\]

- **Quantifiers:**

\[
\begin{align*}
\phi([a, b]) &\implies \forall x \in [a, b]. \phi(x) &\implies &\phi\left(\frac{a+b}{2}\right) \\
\phi\left(\frac{a+b}{2}\right) &\implies \exists x \in [a, b]. \phi(x) &\implies &\phi([b, a])
\end{align*}
\]
Refinement

\[ \phi^- \implies \phi \implies \phi^+ \]

- If \( \phi^- = \bot \) and \( \phi^+ = \top \) we cannot say much about \( \phi \).
Refinement

\[ \phi^- \implies \phi \implies \phi^+ \]

- If \( \phi^- = \bot \) and \( \phi^+ = \top \), we cannot say much about \( \phi \).
- To make progress, we refine \( \phi \) to an equivalent formula in which quantifiers range over smaller intervals.
Refinement

\[ \phi^- \implies \phi \implies \phi^+ \]

- If \( \phi^- = \bot \) and \( \phi^+ = \top \) we cannot say much about \( \phi \).
- To make progress, we refine \( \phi \) to an equivalent formula in which quantifiers range over smaller intervals.
- A simple strategy is to split quantified intervals in halves:
Refinement

\[ \phi^- \implies \phi \implies \phi^+ \]

- If \( \phi^- = \perp \) and \( \phi^+ = \top \) we cannot say much about \( \phi \).
- To make progress, we refine \( \phi \) to an equivalent formula in which quantifiers range over smaller intervals.
- A simple strategy is to split quantified intervals in halves:
  - \( \forall x \in [a, b] . \phi(x) \) is refined to
    \[ (\forall x \in [a, \frac{a+b}{2}] . \phi(x)) \land (\forall x \in [\frac{a+b}{2}, b] . \phi(x)) \]
Refinement

\[ \phi^- \implies \phi \implies \phi^+ \]

- If \( \phi^- = \bot \) and \( \phi^+ = \top \) we cannot say much about \( \phi \).
- To make progress, we refine \( \phi \) to an equivalent formula in which quantifiers range over smaller intervals.
- A simple strategy is to split quantified intervals in halves:
  - \( \forall x \in [a, b] \cdot \phi(x) \) is refined to
    \[ (\forall x \in [a, \frac{a+b}{2}] \cdot \phi(x)) \land (\forall x \in [\frac{a+b}{2}, b] \cdot \phi(x)) \]
  - \( \exists x \in [a, b] \cdot \phi(x) \) is refined to
    \[ (\exists x \in [a, \frac{a+b}{2}] \cdot \phi(x)) \lor (\exists x \in [\frac{a+b}{2}, b] \cdot \phi(x)) \]
Refinement

\[ \phi^- \implies \phi \implies \phi^+ \]

- If \( \phi^- = \bot \) and \( \phi^+ = \top \) we cannot say much about \( \phi \).
- To make progress, we refine \( \phi \) to an equivalent formula in which quantifiers range over smaller intervals.
- A simple strategy is to split quantified intervals in halves:
  - \( \forall x \in [a, b] . \phi(x) \) is refined to
  \[ (\forall x \in [a, \frac{a+b}{2}] . \phi(x)) \land (\forall x \in [\frac{a+b}{2}, b] . \phi(x)) \]
  - \( \exists x \in [a, b] . \phi(x) \) is refined to
  \[ (\exists x \in [a, \frac{a+b}{2}] . \phi(x)) \lor (\exists x \in [\frac{a+b}{2}, b] . \phi(x)) \]
- This amounts to searching with bisection.
Refinement of cuts

- To refine a cut

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

we try to move \( a \leftrightarrow a' \) and \( b \leftrightarrow b' \).

\[
\begin{array}{cccc}
  a & a' & b' & b \\
  \text{ [ } & \text{ [ } & \text{ [ } & \text{ [ }
\end{array}
\]
Refinement of cuts

- To refine a cut

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

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

- If $\phi^-(a') = \top$ then move $a \leftrightarrow a'$.
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 \(b \mapsto b'\).
Refinement of cuts

- To refine a cut

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

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

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

- To evaluate a sentence $\phi$: 

  - if $\cdot =$ $\cdot$ then output $\cdot$, 
  - if $\cdot +$ $? =$ $\cdot$ then output $\cdot$, 
  - otherwise refine $\cdot$ and repeat.

Evaluation may not terminate, but this is expected, as $\cdot$ is only semi-decidable.
Evaluation

To evaluate a sentence $\phi$:
- if $\phi^- = \top$ then output $\top$, 
- otherwise refine and repeat.

Evaluation may not terminate, but this is expected, as $\phi$ is only semi-decidable.
To evaluate a sentence $\phi$:
- if $\phi^- = \top$ then output $\top$,
- if $\phi^+ = \bot$ then output $\bot$,
Evaluation

- To evaluate a sentence $\phi$:
  - if $\phi^- = \top$ then output $\top$,
  - if $\phi^+ = \bot$ then output $\bot$,
  - otherwise refine $\phi$ and repeat.
Evaluation

- To evaluate a sentence $\phi$:
  - if $\phi^- = \top$ then output $\top$,
  - if $\phi^+ = \bot$ then output $\bot$,
  - otherwise refine $\phi$ and repeat.

- Evaluation may not terminate, but this is expected, as $\phi$ is only semidecidable.
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.
Questions

- How do we incorporate $\mathbb{N}$ and recursion?
Questions

▷ How do we incorporate \( \mathbb{N} \) and recursion?
▷ How to extend Newton’s method to improper intervals?
Questions

- How do we incorporate $\mathbb{N}$ and recursion?
- How to extend Newton’s method to improper intervals?
- How to extend Newton’s method to the multivariate case?
Questions

- How do we incorporate $\mathbb{N}$ and recursion?
- How to extend Newton’s method to improper intervals?
- How to extend Newton’s method to the multivariate case?
- Can we do higher-type computations $\int$ and $\frac{d}{dx}$?
Questions

- How do we incorporate \( \mathbb{N} \) and recursion?
- How to extend Newton’s method to improper intervals?
- How to extend Newton’s method to the multivariate case?
- Can we do higher-type computations \( \int \) and \( \frac{d}{dx} \)?
- Can this lead to a useful domain-specific language?