On Envelopes and Backward Approximations

Eike Neumann

Max Planck Institute for Software Systems, Saarbrücken, EU

CCC 2021, Birmingham, UK
24 September 2021

Acknowledgements

This work is heavily inspired by discussions with Franz Brausse, Pieter Collins, Michal Konečný, Norbert Müller, Sewon Park, Florian Steinberg, and Martin Ziegler during my secondments within the CID project at KAIST in 2017 and 2019.

Folklore observations surrounding comparison of real numbers for inequality.

$$\frac{1}{\sqrt{2\pi}}\int_0^\infty x^{\sqrt{1729}}e^{-x}\,\mathrm{dx} \overset{?}{\leq} \sqrt{1729}^{\sqrt{1729}+\frac{1}{2}}e^{-\sqrt{1729}}e^{\frac{1}{12\sqrt{1729}}}$$

Let's compute some functions!

1. Suppose we have implemented algorithms for computing functions $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$.
2. Let's use these to compute a new function!
3. Why not take

$$h(x) = \begin{cases} f(x) & \text{if x } \geq 0, \\ g(x) & \text{if } x < 0. \end{cases}$$

4. Here's an algorithm that computes it:

if $(x \geq 0)$ then:
    return $f(x)$;
else:
    return $g(x)$;

Let's compute some functions!

1. Suppose we have implemented algorithms for computing functions $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$.
2. Let's use these to compute a new function!
3. Why not take

$$h(x) = \begin{cases} f(x) & \text{if } x \ge 0, \\ g(x) & \text{if } x < 0. \end{cases}$$

4. Here's an algorithm that computes it:

Undecidable!

if $\boxed{(x \ge 0)}$ then:
    return $f(x)$;
else:
    return $g(x)$;

Upper semantics

First solution: replace $\leq : \mathbb{R}^2 \to \{\text{True}, \text{False}\}$ with

$$\preceq : \mathbb{R}^2 \to \{\text{True}, \text{False}, \bot\},\ (x \preceq y) = \begin{cases} \text{True} & \text{if } x < y \\ \text{False} & \text{if } x > y \\ \bot & \text{if } x = y. \end{cases}$$

Equivalently,

$$\preceq : \mathbb{R}^2 \to \mathscr{K}(\{\text{True}, \text{False}\}),\ (x \preceq y) = \begin{cases} \{\text{True}\} & \text{if } x > y \\ \{\text{False}\} & \text{if } x < y \\ \{\text{True}, \text{False}\} & \text{if } x = y. \end{cases}$$

Upper semantics

Run the algorithm using upper powerspace semantics:

Upper semantics

Run the algorithm using upper powerspace semantics:

if $(x \succeq 0)$ then:
    return $f(x)$;
else:
    return $g(x)$;

Upper semantics

Run the algorithm using upper powerspace semantics:

$x = 0$

if $(x \geq 0)$ then:

  return $f(x)$;

else:

  return $g(x)$;

Upper semantics

Run the algorithm using upper powerspace semantics:

$$x = 0$$

if $(0 \geq 0)$ then:

  return $f(x)$;

else:

  return $g(x)$;

Upper semantics

Run the algorithm using upper powerspace semantics:

$x = 0$

if ($\{True, False\}$) then:

return $f(x)$;

else:

return $g(x)$;

Upper semantics

Run the algorithm using upper powerspace semantics:

$x = 0$

if $(\{True, False\})$ then:

return $f(0)$;

else:

return $g(x)$;

Upper semantics

Run the algorithm using upper powerspace semantics:

$x = 0$

if ($\{True, False\}$) then:

return $f(0)$;

else:

return $g(0)$;

Upper semantics

Run the algorithm using upper powerspace semantics:

$x = 0$

if $(\{True, False\})$ then:

$\quad$ **return** $f(0)$;

else:

$\quad$ **return** $g(0)$;

$\rightsquigarrow$ **value** $= \{f(0), g(0)\} \in \mathscr{K}(\{\mathbf{True}, \mathbf{False}\})$

Upper semantics

Run the algorithm using upper powerspace semantics:

$x = 0$

if ($\{True, False\}$) then:

return $f(0)$;

else:

return $g(0)$;

$\rightsquigarrow$ **value** $= \{f(0), g(0)\} \in \mathscr{K}(\{\mathbf{True}, \mathbf{False}\})$

$\Rightarrow$ Agrees with $h(x)$ if and only if $f(0) = g(0)$.

Lower semantics

Second solution: replace $\leq : \mathbb{R}^2 \to \{\text{True}, \text{False}\}$ with

$$^{\dagger}{\leq} : \mathbb{R}^2 \times \mathbb{Q}_{>0} \rightsquigarrow \{\text{True}, \text{False}\},\ (x \,{}^{\dagger}{\leq}_{\delta}\, y) = \begin{cases} \{\text{True}\} & \text{if } x \geq y + 2\delta \\ \{\text{False}\} & \text{if } x \leq y - 2\delta \\ \{\text{True}, \text{False}\} & \text{if } |x - y| < 2\delta. \end{cases}$$

Lower semantics

Fix $\delta > 0$. Run the algorithm using lower semantics:

Lower semantics

Fix $\delta > 0$. Run the algorithm using lower semantics:

if $(x^{\dagger} \geq_{\delta} 0)$ then:
    return $f(x)$;
else:
    return $g(x)$;

Lower semantics

Fix $\delta > 0$. Run the algorithm using lower semantics:

$\eta \in (-2\delta, 2\delta)$

if $(x^{\dagger} \geq_{\delta} 0)$ then:

return $f(x)$;

else:

return $g(x)$;

Lower semantics

Fix $\delta > 0$. Run the algorithm using lower semantics:

$\eta \in (-2\delta, 2\delta)$

if $(\eta^{\dagger} \geq_{\delta} 0)$ then:

return $f(x)$;

else:

return $g(x)$;

Lower semantics

Fix $\delta > 0$. Run the algorithm using lower semantics:

$\eta \in (-2\delta, 2\delta)$

if (*True*) then: Non-deterministic choice.

return $f(x)$;

else:

return $g(x)$;

Lower semantics

Fix $\delta > 0$. Run the algorithm using lower semantics:

$\eta \in (-2\delta, 2\delta)$

if (*True*) then: Non-deterministic choice.

return $f(\eta)$;

else:

return $g(x)$;

Lower semantics

Fix $\delta > 0$. Run the algorithm using lower semantics:

$\eta \in (-2\delta, 2\delta)$

if ($True$) then: Non-deterministic choice.

return $f(\eta)$;

else:

return $g(x)$;

$\rightsquigarrow$ **value** $\in \{f(\eta), g(\eta)\}$

Lower semantics

Fix $\delta > 0$. Run the algorithm using lower semantics:

$\eta \in (-2\delta, 2\delta)$

if (*True*) then: Non-deterministic choice.

return $f(\eta)$;

else:

return $g(x)$;

$\rightsquigarrow$ **value** $\in \{f(\eta), g(\eta)\}$

$\Rightarrow$ ε-close to $h(\eta)$ if $\delta \in \min\{\omega_f(0, \varepsilon/2), \omega_g(0, \varepsilon/2)\}$.

Lower semantics

Fix $\delta > 0$. Run the algorithm using lower semantics:

$\eta \in (-2\delta, 2\delta)$

if (*True*) then: Non-deterministic choice.

return $f(\eta)$;

else:

return $g(x)$;

$\rightsquigarrow$ **value** $\in \{f(\eta), g(\eta)\}$

$\Rightarrow$ ε-close to $h(\eta)$ if $\delta \in \min\{\omega_f(0,\varepsilon/2), \omega_g(0,\varepsilon/2)\}$.

moduli of continuity

Summary

Naïve semantics:

if $(x \geq 0)$ then:
    return $f(x)$;
else:
    return $g(x)$;

Obviously correct.
Easiest to understand.
Impossible to execute on a machine.

Upper semantics:

if $(x \succeq 0)$ then:
    return $f(x)$;
else:
    return $g(x)$;

$f(0) = g(0)$.

Slightly harder to understand.
Can be executed. Quite inefficient.

Lower semantics:

if $(x^{\dagger} \geq_{\delta} 0)$ then:
    return $f(x)$;
else:
    return $g(x)$;

$\delta \in \min\{\omega_f(0, \varepsilon/2), \omega_g(0, \varepsilon/2)\}$.

Hardest to understand.
Correctness requires "hard analysis".
Can be executed efficiently.

Let's compute more functions!

1. Goal: relate "upper" and "lower" semantics of "arbitrary" (naïve) programs.
2. For now, focus on chains of function compositions:

$$f_n \circ \cdots \circ f_1$$

   where $f_1, \ldots, f_n$ are arbitrary (!) functions between effective metric spaces.

Backward Approximations

Let $f : X \to Y$ be an arbitrary (!) function between effective metric spaces. Let

$$^{\dagger}f : X \times \mathbb{Q}_{>0} \rightsquigarrow Y, \; ^{\dagger}f(x, \delta) = \{f(\tilde{x}) \mid \tilde{x} \in B(x, \delta)\}$$

be its *backward approximation*.

1. Very common relaxation: equation solving, matrix diagonalisation, backwards stable algorithms...
2. Useful for computing functions that do depend continuously on the input.
3. $^{\dagger}f$ is always continuous.
4. $^{\dagger}f$ computable if f has a computable left inverse.

Backward Approximations

Idea: instead of computing

$$f_n \circ \cdots \circ f_1(x)$$

compute

$$^{\dagger}f_n(\cdot, \delta_n) \circ \cdots \circ {}^{\dagger}f_1(\cdot, \delta_1)$$

for "sufficiently small" δ_i's and hope that the result is close to the true result.

Backward Approximations

Idea: instead of computing

$$f_n \circ \dots \circ f_1(x)$$

compute

$$^\dagger f_n(\cdot, \delta_n) \circ \dots \circ {}^\dagger f_1(\cdot, \delta_1)$$

for "sufficiently small" δ_i's and ~~hope~~ prove that the result is close to the true result.

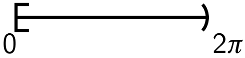
Main Question

Let $x \in X$. Under which assumptions on $f_1, \dots, f_n$ is it possible to find for all $\varepsilon > 0$ numbers $\delta_1 > 0, \dots, \delta_n > 0$ such that

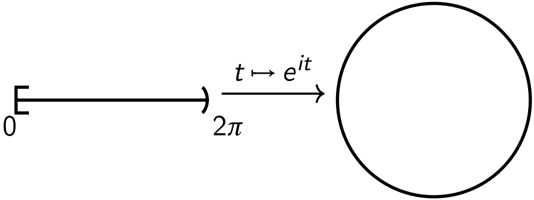
$$^{\dagger}f_n(\cdot, \delta_n) \circ \cdots \circ {}^{\dagger}f_1(\cdot, \delta_1)(x) \subseteq B\left(f_n \circ \cdots \circ f_1(x), \varepsilon\right)?$$

1. Obvious necessary condition: $f_n \circ \cdots \circ f_1$ continuous at x.
2. Obviously not sufficient.

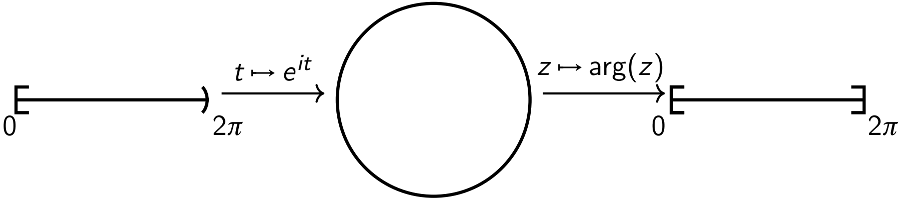
Example



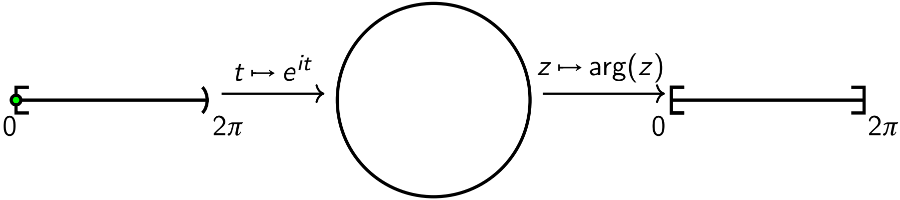
Example



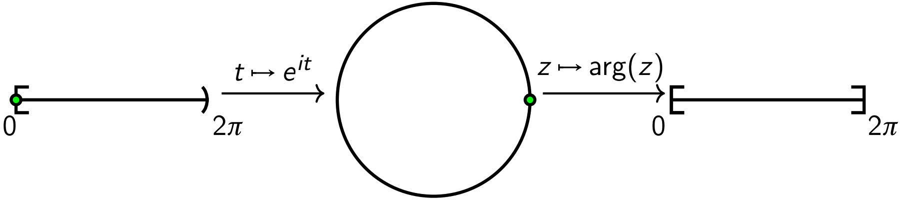
Example



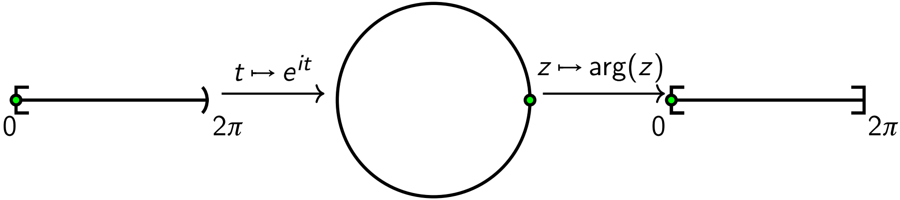
Example



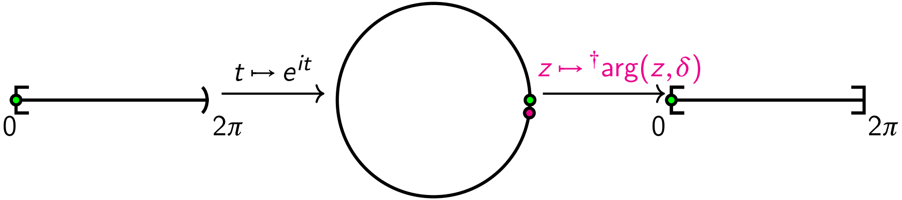
Example



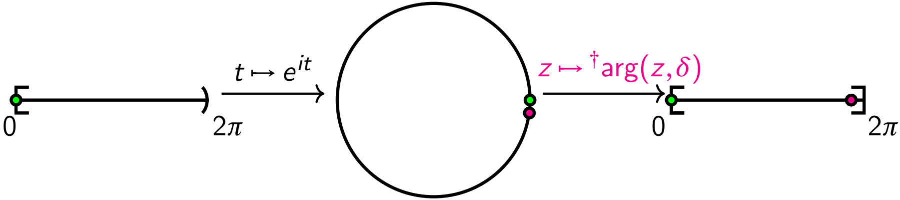
Example



Example



Example



The lattice of compacts

1. Let Y be an effective metric space.
2. Let $\mathscr{K}(Y)$ be the space of compact subsets of Y with the upper Vietoris topology.
3. Let $\mathscr{K}_\perp(Y)$ be the same space with a bottom element added.
4. Then $\mathscr{K}_\perp(Y)$ is a complete lattice. Effectively so:

$$\bigcup : \mathscr{K}(\mathscr{K}_\perp(Y)) \to \mathscr{K}_\perp(Y)$$

and

$$\bigcap : \mathscr{V}(\mathscr{K}_\perp(Y)) \to \mathscr{K}_\perp(Y)$$

are continuous.

The lattice of compacts

Every function

$$f\colon X \to \mathscr{K}_\perp(Y)$$

has a best continuous approximation

$$F\colon X \to \mathscr{K}_\perp(Y),$$

i.e.,

1. $f(x) \in F(x)$ for all $x \in X$.
2. If $G\colon X \to \mathscr{K}_\perp(Y)$ is continuous with $f(x) \in G(x)$ for all $x \in X$ then $F(x) \subseteq G(x)$ for all $x \in X$.

$\mathscr{K}_\perp$ is a monad, yielding a natural notion of composition for functions $F\colon X \to \mathscr{K}_\perp(Y)$, $G\colon Y \to \mathscr{K}_\perp(Z)$. Explicitly:

$$G \circ F(x) = \bigcup_{y \in F(x)} G(y).$$

Overview

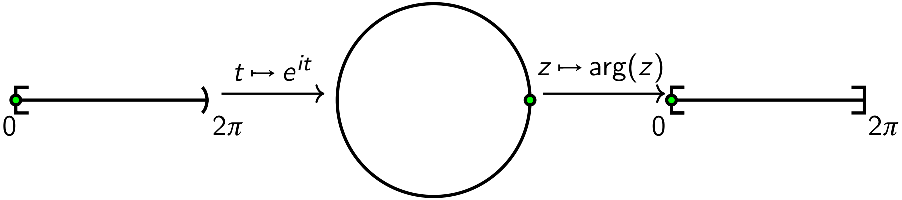
$$
\begin{tikzcd}
& \mathscr{K}_\perp(X_2) \arrow[r, "(F_2)_*"] & \mathscr{K}_\perp(X_3) \arrow[r, "(F_3)_*"] & \cdots \arrow[r, "(F_{n+1})_*"] & \mathscr{K}_\perp(X_{n+1}) \\
X_1 \arrow[ur, "F_1"] \arrow[r, "f_1"] & X_2 \arrow[u, "\{\cdot\}"] \arrow[ur, "F_2"] \arrow[r, "f_2"] & X_3 \arrow[u, "\{\cdot\}"] \arrow[ur, "F_3"] \arrow[r, "f_3"] & \cdots \arrow[ur, "F_n"] \arrow[r, "f_n"] & X_{n+1} \arrow[u, "\{\cdot\}"]
\end{tikzcd}
$$

Theorem

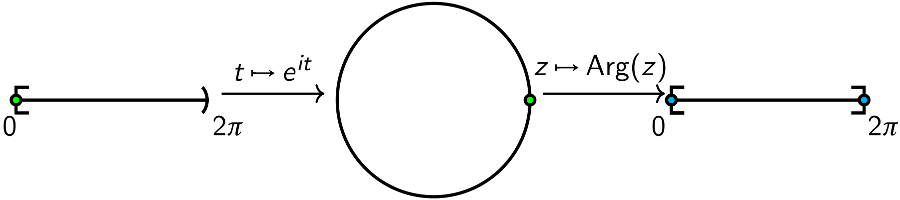
Let $F_i : X_i \to \mathscr{K}_\perp(X_{i+1})$ be the best continuous approximation of $f_i : X_i \to X_{i+1}$. Assume that $F_i(x) \neq \perp$ for all $x \in X_i$. Then the following are equivalent:

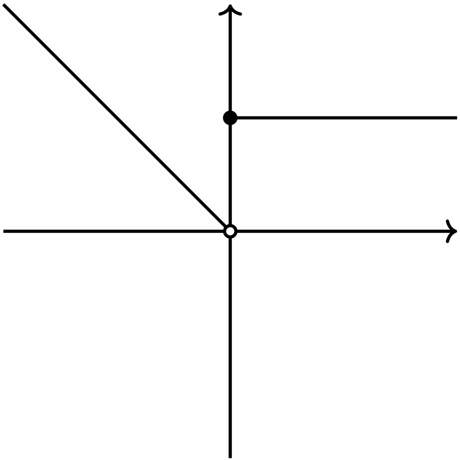
1. $$\forall \varepsilon > 0. \forall x \in X_1. \exists \delta > 0.$$ $$\left({}^{\dagger} f_n(\cdot,\delta) \circ \dots \circ {}^{\dagger} f_1(\cdot,\delta)(x) \subseteq B(f_n \circ \dots \circ f_1(x), \varepsilon).\right).$$
2. *There is a continuous witness $\omega : X_1 \times \mathbb{Q}_{>0} \rightsquigarrow \mathbb{Q}_{>0}$ for the above.*
3. *We have $F_n \circ \dots \circ F_1(x) = \{f_n \circ \dots \circ f_1(x)\}$ for all $x \in X_1$.*

Confirming what we already know

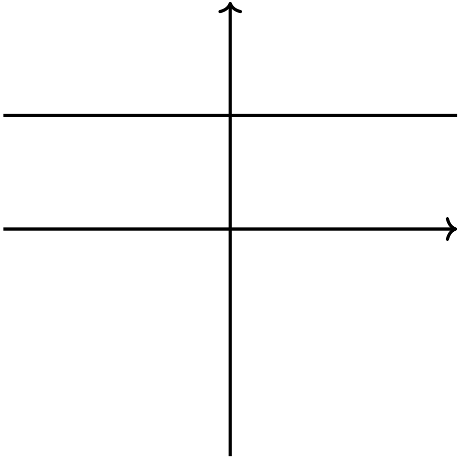


Confirming what we already know

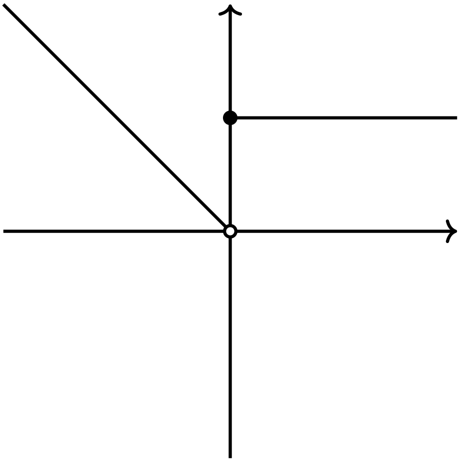




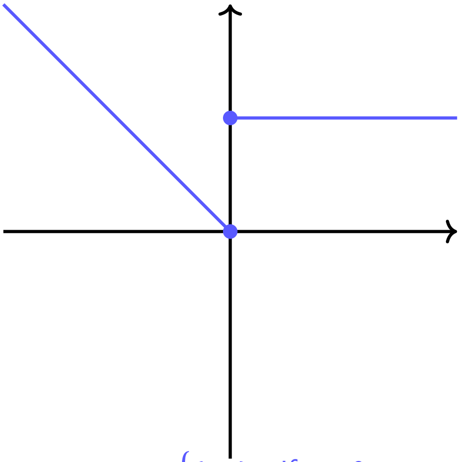
$$f(x) = \begin{cases} -x & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases}$$



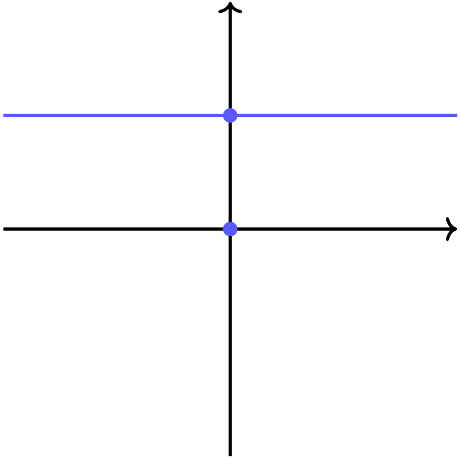
$$f \circ f(x) = 1.$$



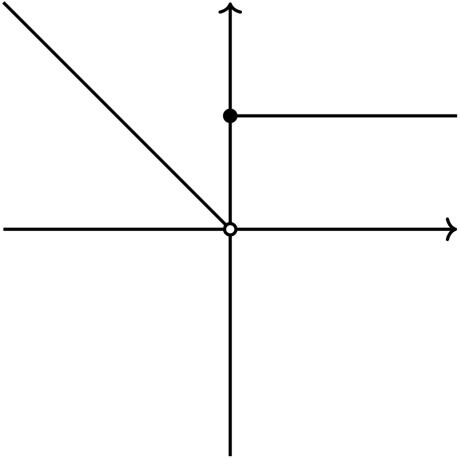
$$f(x) = \begin{cases} -x & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases}$$



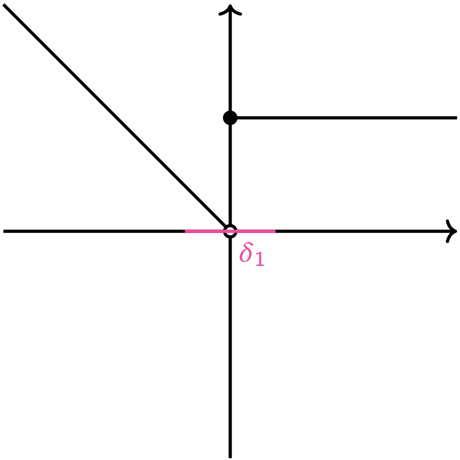
$$F(x) = \begin{cases} \{-x\} & \text{if } x < 0, \\ \{1\} & \text{if } x > 0, \\ \{0,1\} & \text{if } x = 0. \end{cases}$$



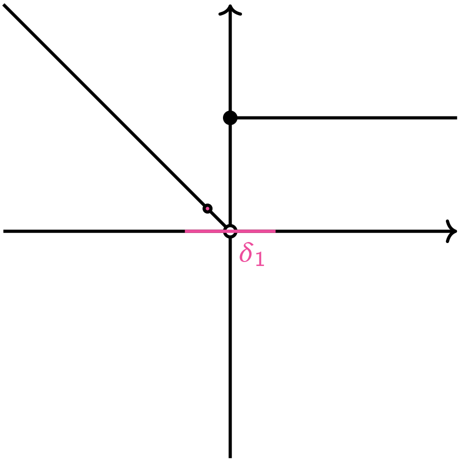
$$F \circ F(x) = \begin{cases} \{1\} & \text{if } x \neq 0, \\ \{0,1\} & \text{if } x = 0. \end{cases}$$



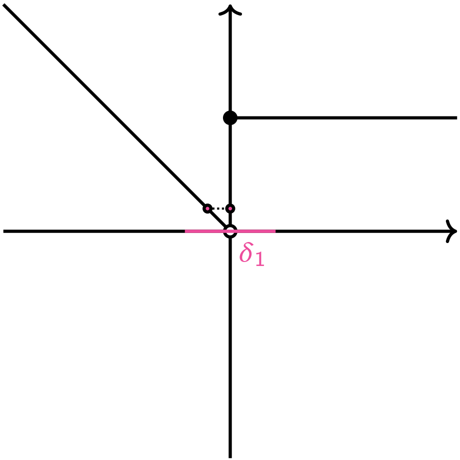
$$f(x) = \begin{cases} -x & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases}$$



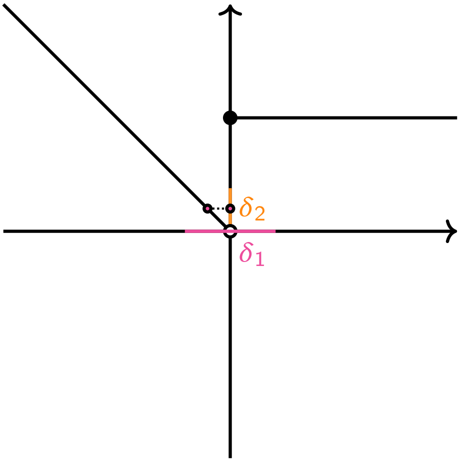
$$f(x) = \begin{cases} -x & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases}$$



$$f(x) = \begin{cases} -x & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases}$$



$$f(x) = \begin{cases} -x & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases}$$



$$f(x) = \begin{cases} -x & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases}$$

Possible notions of convergence

1. A-priori uniform δ:

$$\forall \varepsilon > 0. \forall x \in X_1. \exists \delta > 0.$$
$$\left({}^{\dagger} f_n(\cdot, \delta) \circ \cdots \circ {}^{\dagger} f_1(\cdot, \delta)(x) \subseteq B(f_n \circ \cdots \circ f_1(x), \varepsilon). \right).$$

2. Adaptive scheme:

$$\begin{aligned}
&\forall \varepsilon > 0. \\
&\forall x_1 \in X_1. \exists \delta_1 > 0. \\
&\forall x_2 \in f_1(x_1, \delta_1). \exists \delta_2 > 0. \\
&\ldots. \\
&\forall x_n \in f_{n-1}(x_{n-1}, \delta_{n-1}). \exists \delta_n > 0. \\
&\forall x_{n+1} \in f_n(x_n, \delta_n). \\
&\qquad (x_{n+1} \in B(f_n \circ \cdots \circ f_1(x_1), \varepsilon)).
\end{aligned}$$

Universal Envelopes

$$
\begin{tikzcd}
 & & \mathscr{K}_\perp(Y) \\
X \arrow[r, "f"'] \arrow[urr, bend left, "F"] & Y \arrow[ur, "\{\cdot\}"'] \arrow[u, phantom, "\leq"] &
\end{tikzcd}
$$

Universal Envelopes

$$\begin{tikzcd} & & \mathscr{K}_\perp(Y) \text{ Complete Lattice} \\ X \arrow[r, "f"'] \arrow[urr, bend left, "F"] & Y \arrow[ur, "\{\cdot\}"'] \arrow[u, phantom, "\leq"] & \end{tikzcd}$$

Universal Envelopes

$$\begin{tikzcd}
& & L \\
X \arrow[urr, bend left, "F"] \arrow[r, "f"] & Y \arrow[ur, "\xi_L"'] \arrow[u, phantom, "\leq"] &
\end{tikzcd}$$

Universal Envelopes

$$
\begin{tikzcd}
 & & L \\
X \arrow[r, "f"] \arrow[urr, bend left, "F"] \arrow[drr, bend right, "G"'] \arrow[ur, phantom, "\leq"] \arrow[dr, phantom, "\leq"] & Y \arrow[ur, "\xi_L"'] \arrow[dr, "\xi_M"] & \\
 & & M
\end{tikzcd}
$$

Universal Envelopes

$$
\begin{tikzcd}
 & & L \arrow[dd, "\exists\Phi"] \\
X \arrow[r, "f"] \arrow[urr, bend left, "F"] \arrow[drr, bend right, "\forall G"'] \arrow[r, phantom, shift left=6ex, "\leq"] \arrow[r, phantom, shift right=6ex, "\leq"] & Y \arrow[ur, "\xi_L"'] \arrow[dr, "\forall\xi_M"] & \\
 & & M
\end{tikzcd}
$$

1. $\Phi \circ \xi_L = \xi_M$
2. $\Phi \circ F \geq G$

Universal Envelopes

$$\begin{tikzcd}
 & & L \arrow[dd, "\exists\Phi"] \\
X \arrow[r, "f"] \arrow[urr, bend left, "F"] \arrow[drr, bend right, "\forall G"'] \arrow[ur, phantom, "\leq"] \arrow[dr, phantom, "\geq"] & Y \arrow[ur, "\xi_L"'] \arrow[dr, "\forall\xi_M"] & \\
 & & M
\end{tikzcd}$$

1. $\Phi \circ \xi_L = \xi_M$
2. $\Phi \circ F \geq G$

$\rightsquigarrow$ Existence of Φ requires injectivity.

Short comment on injective spaces

1. Injective topological spaces w.r.t. subspace embeddings = continuous lattices.
2. Subspaces in QCB_0 carry *sequentialisation* of relative topology.
3. $\Rightarrow \Sigma$ is not injective w.r.t. subspace embeddings.
4. Solution: consider injectivity with respect to Σ-split embeddings.
5. $i\colon X \to Y$ is Σ-split if $i^*\colon \mathscr{O}(Y) \to \mathscr{O}(X)$ admits a continuous section $s\colon \mathscr{O}(X) \to \mathscr{O}(Y)$.
6. Injective spaces are closed under retracts, finite products, and form an exponential ideal in the category of represented spaces. $\Rightarrow$ For all represented spaces X the space $\mathscr{O}^2(X)$ is injective.
7. Injective spaces are simultaneously $\mathscr{K}$-algebras and $\mathscr{V}$-algebras, and in particular complete lattices.

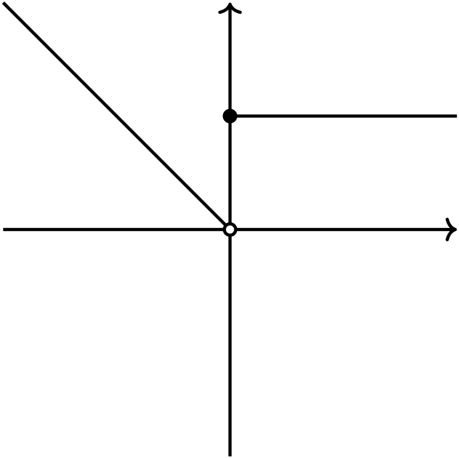
Universal Envelopes

1. Any function $f: X \to Y$ between represented spaces has a universal envelope.
2. There exists an injective space $\mathfrak{A}_f$ (unique up to unique isomorphism) and a continuous surjection

$$\pi: \mathfrak{A}_f \to \mathscr{O}(Y)$$

   such that π preserves arbitrary joins and the fibres $\pi^{-1}(U)$ are injective spaces for all $U \in \mathscr{O}(Y)$.
3. The fibres $\pi^{-1}(U)$ encode extra information that is needed to verify for a given $x \in X$ if $f(x) \in U$.
4. The universal envelope is given by the best continuous approximation of f of type $X \to \mathscr{O}(\mathfrak{A}_f)$.
5. Simplest case $\mathfrak{A}_f = \mathscr{O}(Y)$ and $\pi = \mathrm{id}_{\mathscr{O}(Y)}$. For Hausdorff Y we recover best continuous approximations in $\mathscr{K}_\perp(Y)$.

Universal Envelopes



$$f(x) = \begin{cases} -x & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases}$$

Fibre of $U \in \mathscr{O}(\mathbb{R})$ non-trivial
$\Leftrightarrow$
$1 \in U$, $0 \notin U$,
$(0,\delta) \subseteq U$ for some δ.

$\pi^{-1}(U) = [0,\delta)_{<} / \sim$,
where $\delta = \sup_{(0,\eta)\subseteq U} \eta$,
and $\sim$ identifies all numbers > 0.

$F(x)(U,[\delta]) = \top$
$\Leftrightarrow$
$(x \neq 0 \wedge f(x) \in U)$
$\vee (1 \in U \wedge \exists \eta > 0. B(0,\eta) \subseteq U \wedge |x| < \eta)$
$\vee (1 \in U \wedge |x| < \delta)$

Composition

$$F\colon X \to \mathscr{O}\left(\sum_{U\in\mathscr{O}(Y)} \mathscr{O}(\pi_f^{-1}(U))\right),\ G\colon Y \to \mathscr{O}\left(\sum_{V\in\mathscr{O}(Z)} \mathscr{O}(\pi_g^{-1}(V))\right)$$

$$F \circ G\,(x \in X)\left(V \in \mathscr{O}(Z), \alpha \in \pi_g^{-1}(V)\right)$$
$$= ???$$

Composition

$$F\colon X \to \mathscr{O}\left(\sum_{U\in\mathscr{O}(Y)} \mathscr{O}(\pi_f^{-1}(U))\right),\ G\colon Y \to \mathscr{O}\left(\sum_{V\in\mathscr{O}(Z)} \mathscr{O}(\pi_g^{-1}(V))\right)$$

$$\begin{aligned} &F \circ G\,(x \in X)\left(V \in \mathscr{O}(Z), \alpha \in \pi_g^{-1}(V)\right) \\ &\quad = F\,(x \in X)\,(???, ???) \end{aligned}$$

Composition

$$F\colon X \to \mathscr{O}\left(\sum_{U\in\mathscr{O}(Y)} \mathscr{O}(\pi_f^{-1}(U))\right),\ G\colon Y \to \mathscr{O}\left(\sum_{V\in\mathscr{O}(Z)} \mathscr{O}(\pi_g^{-1}(V))\right)$$

$$\begin{aligned}&F \circ G\,(x \in X)\left(V \in \mathscr{O}(Z), \alpha \in \pi_g^{-1}(V)\right)\\&= F\,(x \in X)\left(U = \{y \in Y \mid G(y, V, \alpha) = \top\} \in \mathscr{O}(Y), ???\right)\end{aligned}$$

Composition

$$F\colon X \to \mathscr{O}\left(\textstyle\sum_{U\in\mathscr{O}(Y)} \mathscr{O}(\pi_f^{-1}(U))\right),\ G\colon Y \to \mathscr{O}\left(\textstyle\sum_{V\in\mathscr{O}(Z)} \mathscr{O}(\pi_g^{-1}(V))\right)$$

$$\begin{aligned}&F\circ G\,(x\in X)\left(V\in\mathscr{O}(Z),\alpha\in\pi_g^{-1}(V)\right)\\ &=F\,(x\in X)\left(U=\{y\in Y \mid G(y,V,\alpha)=\top\}\in\mathscr{O}(Y),\top\in\pi_f^{-1}(U)\right)\end{aligned}$$

Composition

$$F\colon X \to \mathscr{O}\left(\textstyle\sum_{U\in\mathscr{O}(Y)} \mathscr{O}(\pi_f^{-1}(U))\right),\ G\colon Y \to \mathscr{O}\left(\textstyle\sum_{V\in\mathscr{O}(Z)} \mathscr{O}(\pi_g^{-1}(V))\right)$$

$$F\circ G\,(x\in X)\left(V\in\mathscr{O}(Z), \alpha\in\pi_g^{-1}(V)\right)$$
$$= F\,(x\in X)\left(U = \{y\in Y \mid G(y,V,\alpha) = \top\}\in\mathscr{O}(Y), \top\in\pi_f^{-1}(U)\right)$$

No longer continuous in the open set argument.

Composition

$$F\colon X \to \mathscr{O}\left(\sum_{U\in\mathscr{O}(Y)} \mathscr{O}(\pi_f^{-1}(U))\right),\ G\colon Y \to \mathscr{O}\left(\sum_{V\in\mathscr{O}(Z)} \mathscr{O}(\pi_g^{-1}(V))\right)$$

$$F \circ G\,(x \in X)\left(V \in \mathscr{O}(Z), \alpha \in \pi_g^{-1}(V)\right)$$
$$= F\,(x \in X)\left(U = \{y \in Y \mid G(y, V, \alpha) = \top\} \in \mathscr{O}(Y), \top \in \pi_f^{-1}(U)\right)$$

No longer continuous in the open set argument.

This yields a function

$$G \circ F\colon X \to \prod_{V\in\mathscr{O}(Z)} \mathscr{O}\left(\pi_g^{-1}(V)\right)$$

Overview

$$\begin{tikzcd}
 & \prod_U \mathscr{O}\left(\pi_{f_1}^{-1}(U)\right) & \cdots & \prod_U \mathscr{O}\left(\pi_{f_{n-1}}^{-1}(U)\right) & \prod_U \mathscr{O}\left(\pi_{f_n}^{-1}(U)\right) \\
X_1 \arrow[r, "f_1"] \arrow[ur, "F_1"] & X_2 \arrow[r, "f_2"] \arrow[u, "\xi_1"] \arrow[ur, "F_2"] & \ldots \arrow[r, "f_{n-1}"] \arrow[ur, "F_{n-1}"] & X_n \arrow[r, "f_n"] \arrow[u, "\xi_{n-1}"] \arrow[ur, "F_n"] & X_{n+1} \arrow[u, "\xi_n"]
\end{tikzcd}$$

Theorem

The following are equivalent:

1.
$$
\begin{aligned}
&\forall \varepsilon > 0. \\
&\forall x_1 \in X_1. \exists \delta_1 > 0. \\
&\forall x_2 \in {}^{\dagger} f_1(x_1, \delta_1). \exists \delta_2 > 0. \\
&\ldots. \\
&\forall x_n \in {}^{\dagger} f_{n-1}(x_{n-1}, \delta_{n-1}). \exists \delta_n > 0. \\
&\forall x_{n+1} \in {}^{\dagger} f_n(x_n, \delta_n). \\
&\qquad (x_{n+1} \in B(f_n \circ \cdots \circ f_1(x_1), \varepsilon)).
\end{aligned}
$$

2. *We can find continuous witnesses* $\omega_i : X_i \times \mathbb{Q}_{>0} \rightsquigarrow X_{i+1}$ *for the above.*
3. $F_n \circ \cdots \circ F_1(x) = \xi_n \circ f_n \circ \cdots \circ f_1(x)$.

Conclusion/Future Work

1. Connected "upper" and "lower" relaxations of "exact" computational problems.
2. Replaces "hard analysis" questions on ε's and δ's by "soft analysis" questions on equality.
3. Envelopes can serve as a foothold for proving quantitative ε-δ-results.
4. Backward approximations can yield efficient implementations of programs using envelopes.

TODOs:

1. Extend results to WHILE-programs (mostly done).
2. Extend to non-deterministic functions (partially done).
3. Pursue extraction of bounds for δ's from equality proofs (promising preliminary results).
4. Pipe dream: write programs with envelope semantics, prove them correct, compile program + proof into efficient program using backward approximations (entirely delusional).