Applications of Preimage Analysis

Abstract

XXX: todo

Categories and Subject Descriptors CR-number [*subcategory*]: third-level

Keywords keyword1, keyword2

1. Introduction

(Citation to make Kile happy with BibTeX: [1])

2. Finite Program Semantics

3. Set Representation

4. Preimages Under Primitives

XXX: lifts are generally uncomputable, but many specific lifts are computable or approximately computable

XXX: something about computing functions backward by computing inverses forward

XXX: obvious how to do this with invertible functions (though infinite endpoints are a little tricky); not obvious how to do the same with two-argument functions

4.1 Invertible Primitives

We consider only strictly monotone functions on $\mathbb{R}$. Further on, we recover more generality by using language conditionals to implement *piecewise* monotone functions.

One reason we consider only strictly monotone functions is that they are easy to make invertible. Recall that a function is invertible if and only if it is injective (one-to-one) and surjective (onto).

Lemma 4.1.1. *If* $\mathsf{f} : \mathsf{A} \to \mathsf{B}$ *is strictly montone,* f *is injective. If* f *is additionally surjective,* f *and its inverse are continuous.*

Preimages under invertible functions can be computed using their inverses. We are primarily interested in computing preimages under restricted functions.

Lemma 4.1.2. *Let* $\mathsf{A}' \subseteq \mathsf{A}$, $\mathsf{B}' \subseteq \mathsf{B}$, *and* $\mathsf{f} : \mathsf{A} \to \mathsf{B}$ *have inverse* $\mathsf{f}^{-1} : \mathsf{B} \to \mathsf{A}$. *Let* $\mathsf{f}' := \mathsf{restrict}\ \mathsf{f}\ \mathsf{A}'$. *Then*

$$\mathsf{preimage}\ \mathsf{f}'\ \mathsf{B}' \ = \ \mathsf{A}' \cap \mathsf{image}\ \mathsf{f}^{-1}\ \mathsf{B}' \tag{1}$$

These facts suggest that we can compute images (or preimages) of intervals under any strictly monotone, surjective f by applying f (or its inverse) to interval endpoints to yield an interval. This is evident for endpoints in A. Limit endpoints like $+\infty$ require a larger $\overline{\mathsf{f}}$ defined on a compact superset of A.

The next theorem is easier to state with interval notation in which the kind of interval is not baked into the syntax.

Definition 4.1.3 (interval)**.** $[\![\mathsf{a}_1, \mathsf{a}_2, \alpha_1, \alpha_2]\!]$ *denotes an interval, where* $\mathsf{a}_1, \mathsf{a}_2 \in \overline{\mathbb{R}}$ *are extended real endpoints, and* $\alpha_1, \alpha_2 \in \mathsf{Bool}$ *determine whether* a_1 *and* a_2 *are contained in the interval.*

Example 4.1.4. Examples of intervals using $[\![\cdots]\!]$ notation are

$$\begin{aligned} [\![0, 1, \mathsf{true}, \mathsf{false}]\!] &= [0, 1) \\ [\![-\infty, 0, \mathsf{false}, \mathsf{true}]\!] &= (-\infty, 0] \\ [\![-\infty, +\infty, \mathsf{false}, \mathsf{false}]\!] &= (-\infty, +\infty) = \mathbb{R} \\ [\![-\infty, +\infty, \mathsf{true}, \mathsf{true}]\!] &= [-\infty, +\infty] = \overline{\mathbb{R}} \end{aligned} \tag{2}$$

All but the last are subsets of $\mathbb{R}$. ♢

Theorem 4.1.5 (images of intervals by endpoints)**.** *Let* $\overline{\mathsf{A}}$ *and* $\overline{\mathsf{B}}$ *be compact subsets of* $\overline{\mathbb{R}}$, $\overline{\mathsf{f}} : \overline{\mathsf{A}} \to \overline{\mathsf{B}}$ *strictly monotone and surjective, and* f *the restriction of* $\overline{\mathsf{f}}$ *to some* $\mathsf{A} \subseteq \overline{\mathsf{A}}$. *For all nonempty* $[\![\mathsf{a}_1, \mathsf{a}_2, \alpha_1, \alpha_2]\!] \subseteq \mathsf{A}$,

- *If* $\overline{\mathsf{f}}$ *is increasing,* $\mathsf{image}\ \mathsf{f}\ [\![\mathsf{a}_1, \mathsf{a}_2, \alpha_1, \alpha_2]\!] = [\![\overline{\mathsf{f}}\ \mathsf{a}_1, \overline{\mathsf{f}}\ \mathsf{a}_2, \alpha_1, \alpha_2]\!]$.
- *If* $\overline{\mathsf{f}}$ *is decreasing,* $\mathsf{image}\ \mathsf{f}\ [\![\mathsf{a}_1, \mathsf{a}_2, \alpha_1, \alpha_2]\!] = [\![\overline{\mathsf{f}}\ \mathsf{a}_2, \overline{\mathsf{f}}\ \mathsf{a}_1, \alpha_2, \alpha_1]\!]$.

Proof. Because $\overline{\mathsf{A}}$ is compact and totally ordered, every subset of $\overline{\mathsf{A}}$ has a lower and an upper bound in $\overline{\mathsf{A}}$. Therefore, the endpoints of every interval subset of A are in $\overline{\mathsf{A}}$.

Let $(\mathsf{a}_1, \mathsf{a}_2] \subseteq \mathsf{A}$. Suppose $\overline{\mathsf{f}}$ is strictly increasing; thus $\mathsf{a}_1 < \mathsf{a} \leq \mathsf{a}_2$ if and only if $\overline{\mathsf{f}}\ \mathsf{a}_1 < \overline{\mathsf{f}}\ \mathsf{a} \leq \overline{\mathsf{f}}\ \mathsf{a}_2$, so $\mathsf{image}\ \mathsf{f}\ (\mathsf{a}_1, \mathsf{a}_2] = \mathsf{image}\ \overline{\mathsf{f}}\ (\mathsf{a}_1, \mathsf{a}_2] = (\overline{\mathsf{f}}\ \mathsf{a}_1, \overline{\mathsf{f}}\ \mathsf{a}_2]$. The remaining cases are similar. □

To use Theorem 4.1.5 to compute preimages under f by computing images under its inverse f^{-1}, we must know whether f^{-1} is increasing or decreasing. The following lemma can help.

Lemma 4.1.6. *If* $\mathsf{f} : \mathsf{A} \to \mathsf{B}$ *is strictly monotone and surjective with inverse* $\mathsf{f}^{-1} : \mathsf{B} \to \mathsf{A}$, *then* f *is increasing if and only if* f^{-1} *is increasing.*

Example 4.1.7. To extend $\mathsf{log} : (0, +\infty) \to \mathbb{R}$ to $[0, +\infty]$, define

$$\begin{aligned} \overline{\mathsf{log}}\ \mathsf{a} \ &:= \ \lim_{\mathsf{a}' \to \mathsf{a}} \mathsf{log}\ \mathsf{a}' \\ &= \ \mathsf{cond}\ \ \mathsf{a} = 0 \ \longrightarrow\ -\infty \\ &\qquad\quad\ \ \mathsf{a} = +\infty \ \longrightarrow\ +\infty \\ &\qquad\quad\ \ \mathsf{else} \ \longrightarrow\ \mathsf{log}\ \mathsf{a} \end{aligned} \tag{3}$$

The extension of its inverse $\exp$ is $\overline{\exp} : \overline{\mathbb{R}} \to [0, +\infty]$, defined similarly, which by Lemma 4.1.6 is also strictly increasing. Thus,

$$\begin{aligned} \text{image } \log\ (0, 1] &= (\overline{\log}\ 0, \overline{\log}\ 1] = (-\infty, 0] \\ \text{preimage } \log\ [0, +\infty) &= \text{image } \exp\ [0, +\infty) \\ &= [\overline{\exp}\ 0, \overline{\exp}\ +\infty) = [1, +\infty) \end{aligned} \tag{4}$$

by Theorem 4.1.5 and Lemma 4.1.2, ♢

4.2 Two-Argument Primitives

We do not expect to be able to compute preimages under $\mathbb{R} \times \mathbb{R} \rightharpoonup \mathbb{R}$ primitives by simply inverting them. Two-argument invertible real functions are difficult to define and are usually pathological.

Instead, we compute approximate preimages only, using inverses with respect to one argument (with the other held constant).

Definition 4.2.1 (axial inverse)**.** *Let* $f_c : A \times B \to C$*. Functions* $f_a : B \times C \to A$ *and* $f_b : C \times A \to B$ *defined so that*

$$f_c\ \langle a, b\rangle = c \iff f_a\ \langle b, c\rangle = a \iff f_b\ \langle c, a\rangle = b \tag{5}$$

are ***axial inverses*** *with respect to* f_c*'s first and second arguments.*

We call f_c ***axis-invertible*** when it has axial inverses f_a and f_b. We call f_a the ***first axial inverse*** of f_c because it is the inverse of f_c along the first axis: f_a with only c varying (i.e. $\lambda c \in C.\ f_a\ \langle b, c\rangle$), is the inverse of f_c with only a varying (i.e. $\lambda a \in A.\ f_c\ \langle a, b\rangle$). Similarly, f_b is the ***second axial inverse***.

TODO: plot of f_c with b fixed?

Example 4.2.2. Let $\text{add}_c : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $\text{add}_c\ \langle a, b\rangle := a + b$. Its axial inverses are $\text{add}_a\ \langle b, c\rangle := c - b$ and $\text{add}_b\ \langle c, a\rangle := c - a$. ♢

We have chosen the axial inverse function types carefully: they are the only types for which f_c, f_a and f_b form a cyclic group.

Lemma 4.2.3. *The following statements are equivalent.*

- f_c *has axial inverses* f_a *and* f_b.
- f_a *has axial inverses* f_b *and* f_c.
- f_b *has axial inverses* f_c *and* f_a.

Equivalently, every axis-invertible function generates a cyclic group of order 3 by inversion in the first axis.

This fact is analogous to how mutual inverses f and f^{-1} also form a cyclic group (of order 2, generated by inversion). Similar to using mutual inversion to compute preimages under both $\log$ and $\exp$, Lemma 4.2.3 allows computing preimages under two-argument functions related by axial inversion.

Example 4.2.4. Define $\text{sub}_c : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ by $\text{sub}_c\ \langle a, b\rangle := a - b$. Because $\text{sub}_c = \text{add}_b$, $\text{sub}_a = \text{add}_c$ and $\text{sub}_b = \text{add}_a$. ♢

Unlike inverses, axial inverses do not provide a direct way to compute exact preimages. Instead, they provide a way to compute a preimage's smallest rectangular bounding set.

Theorem 4.2.5 (preimage bounds from axial inverse images)**.** *Let* $A' \subseteq A$, $B' \subseteq B$, $C' \subseteq C$, *and* $f_c : A \times B \to C$ *with axial inverses* f_a *and* f_b*. If* $f'_c = \text{restrict } f_c\ (A' \times B')$*, then*

$$\begin{aligned} \text{preimage } f'_c\ C' \subseteq\ & (A' \cap \text{image } f_a\ (B' \times C')) \times \\ & (B' \cap \text{image } f_b\ (C' \times A')) \end{aligned} \tag{6}$$

Further, the right-hand side is the smallest rectangular superset.

Proof. The smallest rectangle containing $\text{preimage } f'_c\ C'$ is

$$\begin{aligned} \text{preimage } f'_c\ C' \subseteq\ & (\text{image fst } (\text{preimage } f'_c\ C')) \times \\ & (\text{image snd } (\text{preimage } f'_c\ C')) \end{aligned} \tag{7}$$

Starting with the first set in the product, expand definitions, distribute fst, replace $f_c\ \langle a, b\rangle = c$ by $f_a\ \langle b, c\rangle = a$, and simplify:

$$\begin{aligned} &\text{image fst } (\text{preimage } f'_c\ C') \\ &= \text{image fst } \{\langle a, b\rangle \in A' \times B' \mid f_c\ \langle a, b\rangle \in C'\} \\ &= \{a \in A' \mid \exists\, b \in B'.\ f_c\ \langle a, b\rangle \in C'\} \\ &= \{a \in A' \mid \exists\, b \in B', c \in C'.\ f_c\ \langle a, b\rangle = c\} \\ &= \{a \in A' \mid \exists\, b \in B', c \in C'.\ f_a\ \langle b, c\rangle = a\} \\ &= \{f_a\ \langle b, c\rangle \mid b \in B', c \in C', f_a\ \langle b, c\rangle \in A'\} \\ &= A' \cap \{f_a\ \langle b, c\rangle \mid b \in B', c \in C'\} \\ &= A' \cap \text{image } f_a\ (B' \times C') \end{aligned}$$

The second set in the product is similar. □

Example 4.2.6. Let $\text{add}'_c := \text{restrict } \text{add}_c\ ([0, 1] \times [0, 2])$. By Theorem 4.2.5,

$$\begin{aligned} \text{preimage } \text{add}'_c\ [0, \tfrac{1}{2}] &\subseteq ([0, 1] \cap \text{image } \text{add}_a\ ([0, 2] \times [0, \tfrac{1}{2}])) \times \\ &\quad ([0, 2] \cap \text{image } \text{add}_b\ ([0, \tfrac{1}{2}] \times [0, 1])) \\ &= ([0, 1] \cap [-2, \tfrac{1}{2}]) \times ([0, 2] \cap [-1, \tfrac{1}{2}]) \\ &= [0, \tfrac{1}{2}] \times [0, \tfrac{1}{2}] \end{aligned}$$

is the smallest rectangular subset of $[0, 1] \times [0, 2]$ that contains the preimage of $[0, \frac{1}{2}]$ under add_c. ♢

At this point, we have an analogue of Lemma 4.1.2, in that we can compute (approximate) preimages by computing images under (axial) inverses. To compute images using interval endpoints, we need analogues of Lemma 4.1.1 (strictly monotone, surjective functions are invertible and continuous), Theorem 4.1.5 (images of intervals by endpoints), and Lemma 4.1.6 (inverse direction).

We first need a notion of properties that hold along an axis for every fixed value of the other argument.

Definition 4.2.7 (uniform axis property)**.** $f_c : A \times B \to C$ *has property* P ***uniformly*** *in its first axis when* $P\ (\text{flip } (\text{curry } f_c)\ b)$ *for all* $b \in B$*, and uniformly in its second axis when* $P\ (\text{curry } f_c\ a)$ *for all* $a \in A$*. If the axis is not specified,* P *holds uniformly for both.*

Now Lemma 4.1.1's analogue is an easy corollary.

Lemma 4.2.8. *Let* $f_c : A \times B \to C$ *for totally ordered* A, B *and* C*. If* f_c *is uniformly surjective and either uniformly strictly increasing or uniformly strictly decreasing in each axis, then* f_c *is axis-invertible; further, it and its axial inverses are continuous.*

From here on, assume axis monotonicity properties are uniform unless otherwise stated.

Example 4.2.9. add_c is uniformly surjective and strictly increasing. sub_c is uniformly surjective and strictly increasing/decreasing in its first/second axis. Therefore, both are axis-invertible. ♢

Restriction usually makes a function not uniformly surjective.

Example 4.2.10. Let $\text{add}'_c : [0, 1] \times [0, 1] \to [0, 2]$, defined by restricting add_c. It is strictly increasing, but not uniformly surjective: the range of $\text{curry } \text{add}'_c\ 0$ is $[0, 1]$, not $[0, 2]$. ♢

Fortunately, restriction sometimes does the opposite.

Example 4.2.11. Define $\text{mul}_c : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ by $\text{mul}_c\ \langle a, b\rangle := a \cdot b$. It is not uniformly surjective nor strictly monotone because $\text{mul}_c\ \langle 0, b\rangle = 0$ for all $b \in B$. But $\text{mul}^{++}_c : (0, +\infty) \times (0, +\infty) \to (0, +\infty)$, and mul_c restricted to the other quadrants, are uniformly surjective and strictly increasing or decreasing in each axis. ♢

Theorem 4.1.5 justifies computing images of intervals with infinite endpoints by applying an extended function to the endpoints.

Its two-argument analogue is more involved because extended, two-argument functions may not be defined at every point.

Example 4.2.12. $\mathsf{add_c}$ cannot be extended to $\overline{\mathsf{add_c}} : \overline{\mathbb{R}} \times \overline{\mathbb{R}} \to \overline{\mathbb{R}}$ in the same way log is extended to $\overline{\mathsf{log}}$ because

$$\lim_{\langle \mathsf{a}', \mathsf{b}' \rangle \to \langle \mathsf{a}, \mathsf{b} \rangle} \mathsf{add_c}\ \langle \mathsf{a}', \mathsf{b}' \rangle \tag{8}$$

does not exist when $\langle \mathsf{a}, \mathsf{b} \rangle$ is $\langle -\infty, +\infty \rangle$ or $\langle +\infty, -\infty \rangle$. ♢

The previous example suggests that extensions of increasing, two-argument functions are always well-defined except at off-diagonal corners. This is true, and similar statements hold for axes with other directions, and for more restricted domains.

Theorem 4.2.13. *Let* $\mathsf{A}, \mathsf{B}, \mathsf{C} \subseteq \mathbb{R}$ *and* $\mathsf{f_c} : \mathsf{A} \times \mathsf{B} \to \mathsf{C}$ *be uniformly surjective and strictly increasing or decreasing in each axis. Let* $\overline{\mathsf{A}}$, $\overline{\mathsf{B}}$ *and* $\overline{\mathsf{C}}$ *be the closures of* A, B *and* C *in* $\overline{\mathbb{R}}$*. The following extension is well-defined:*

$$\begin{aligned} &\overline{\mathsf{f_c}} : (\overline{\mathsf{A}} \times \overline{\mathsf{B}}) \backslash \mathsf{N} \to \overline{\mathsf{C}} \\ &\overline{\mathsf{f_c}}\ \langle \mathsf{a}, \mathsf{b} \rangle\ :=\ \lim_{\langle \mathsf{a}', \mathsf{b}' \rangle \to \langle \mathsf{a}, \mathsf{b} \rangle} \mathsf{f_c}\ \langle \mathsf{a}', \mathsf{b}' \rangle \end{aligned} \tag{9}$$

where $\mathsf{N} := \{\langle \min \overline{\mathsf{A}}, \max \overline{\mathsf{B}} \rangle, \langle \max \overline{\mathsf{A}}, \min \overline{\mathsf{B}} \rangle\}$ *if* $\mathsf{f_c}$ *is increasing or decreasing, and* $\mathsf{N} := \{\langle \min \overline{\mathsf{A}}, \min \overline{\mathsf{B}} \rangle, \langle \max \overline{\mathsf{A}}, \max \overline{\mathsf{B}} \rangle\}$ *if* $\mathsf{f_c}$ *is increasing/decreasing or decreasing/increasing.*

Proof. Suppose $\mathsf{f_c}$ is increasing, and let $\mathsf{g} : \mathbb{N} \to \mathsf{A} \times \mathsf{B}$ be a sequence of $\mathsf{f_c}$'s domain values.

Any g that converges to $\langle \max \overline{\mathsf{A}}, \max \overline{\mathsf{B}} \rangle$ has a strictly increasing subsequence. By monotonicity, $\mathsf{map}\ \mathsf{f_c}\ \mathsf{g}$ has a strictly increasing subsequence. It is bounded by $\max \overline{\mathsf{C}}$, so $\overline{\mathsf{f_c}}\ \langle \max \overline{\mathsf{A}}, \max \overline{\mathsf{B}} \rangle = \max \overline{\mathsf{C}}$. A similar argument proves $\overline{\mathsf{f}}\ \langle \min \overline{\mathsf{A}}, \min \overline{\mathsf{B}} \rangle = \min \overline{\mathsf{C}}$.

For any g that converges to $\langle \max \overline{\mathsf{A}}, \mathsf{b}' \rangle$ for some $\mathsf{b}' \in \mathsf{B}$, define

$$\mathsf{g}'\ :=\ \mathsf{map}\ (\lambda\, \langle \mathsf{a}, \mathsf{b} \rangle.\, \langle \mathsf{f_a}\ \langle \mathsf{b}', \mathsf{f_c}\ \langle \mathsf{a}, \mathsf{b} \rangle \rangle', \mathsf{b}' \rangle)\ \mathsf{g} \tag{10}$$

where $\mathsf{f_a}$ is $\mathsf{f_c}$'s first axial inverse, so that $\mathsf{map}\ \mathsf{f_c}\ \mathsf{g} = \mathsf{map}\ \mathsf{f_c}\ \mathsf{g}'$. Because g' has a subsequence that is strictly increasing in the first of each pair, and because the second of each pair is the constant b', by monotonicity, $\mathsf{map}\ \mathsf{f_c}\ \mathsf{g}'$ has a strictly increasing subsequence. It is bounded by $\max \overline{\mathsf{C}}$, so $\overline{\mathsf{f_c}}\ \langle \max \overline{\mathsf{A}}, \mathsf{b}' \rangle = \max \overline{\mathsf{C}}$. By similar arguments, $\overline{\mathsf{f}}\ \langle \min \overline{\mathsf{A}}, \mathsf{b} \rangle = \min \overline{\mathsf{C}}$ and so on.

Arguments for f decreasing, etc., are similar. □

Following the proof of Theorem 4.2.13, extensions of two-argument functions can be defined by two corner cases, four border cases, and an interior case.

Example 4.2.14. Define $\mathsf{pow_c} : (0,1) \times (0,+\infty) \to (0,1)$ by $\mathsf{pow_c}\ \langle \mathsf{a}, \mathsf{b} \rangle := \exp\ (\mathsf{b} \cdot \log\ \mathsf{a})$, which is increasing/decreasing. Its extension to a subset of $\overline{\mathbb{R}} \times \overline{\mathbb{R}}$ is

$$\begin{aligned} &\overline{\mathsf{pow_c}} : ([0,1] \times [0,+\infty]) \backslash \mathsf{N} \to [0,1] \\ &\overline{\mathsf{pow_c}}\ \langle \mathsf{a}, \mathsf{b} \rangle\ :=\ \mathsf{case}\ \langle \mathsf{a}, \mathsf{b} \rangle \\ &\qquad\qquad \langle 0, +\infty \rangle \longrightarrow 0 \\ &\qquad\qquad \langle 1, 0 \rangle \longrightarrow 1 \\ &\qquad\qquad \langle 0, \mathsf{b} \rangle \longrightarrow 0 \\ &\qquad\qquad \langle 1, \mathsf{b} \rangle \longrightarrow 1 \\ &\qquad\qquad \langle \mathsf{a}, 0 \rangle \longrightarrow 1 \\ &\qquad\qquad \langle \mathsf{a}, +\infty \rangle \longrightarrow 0 \\ &\qquad\qquad \mathsf{else} \longrightarrow \mathsf{pow_c}\ \langle \mathsf{a}, \mathsf{b} \rangle \end{aligned} \tag{11}$$

where $\mathsf{N} := \{\langle 0, 0 \rangle, \langle 1, +\infty \rangle\}$. ♢

The analogue of Theorem 4.1.5 is easiest to state if we have predicates that indicate a function's direction in each axis. Define $\mathsf{inc_1} : (\mathsf{A} \times \mathsf{B} \to \mathsf{C}) \Rightarrow \mathsf{Bool}$ so that $\mathsf{inc_1}\ \mathsf{f}$ if and only if f is strictly increasing in its first axis, and similarly $\mathsf{inc_2}$ so that $\mathsf{inc_2}\ \mathsf{f}$ if and only if f is strictly increasing in its second axis.

Theorem 4.2.15 (images of rectangles by interval endpoints)**.** *Let* $\mathsf{A}, \mathsf{B}, \mathsf{C}$ *be open subsets of* $\mathbb{R}$*, and* $\mathsf{f_c} : \mathsf{A} \times \mathsf{B} \to \mathsf{C}$ *be uniformly surjective and strictly increasing or decreasing in each axis, with extension* $\overline{\mathsf{f_c}}$ *as defined in Theorem 4.2.13. If* $\mathsf{A}' := \llbracket \mathsf{a}_1, \mathsf{a}_2, \alpha_1, \alpha_2 \rrbracket \subseteq \mathsf{A}$ *and* $\mathsf{B}' := \llbracket \mathsf{b}_1, \mathsf{b}_2, \beta_1, \beta_2 \rrbracket \subseteq \mathsf{B}$*, then the image* C' *under* $\mathsf{f_c}$ *is*

$$\begin{aligned} &\mathsf{image}\ \mathsf{f_c}\ (\llbracket \mathsf{a}_1, \mathsf{a}_2, \alpha_1, \alpha_2 \rrbracket \times \llbracket \mathsf{b}_1, \mathsf{b}_2, \beta_1, \beta_2 \rrbracket) \\ &\quad = \mathsf{let}\ \langle \mathsf{a}_1, \mathsf{a}_2, \alpha_1, \alpha_2 \rangle := \mathsf{cond}\ (\mathsf{inc_1}\ \mathsf{f_c}) \longrightarrow \langle \mathsf{a}_1, \mathsf{a}_2, \alpha_1, \alpha_2 \rangle \\ &\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathsf{else} \longrightarrow \langle \mathsf{a}_2, \mathsf{a}_1, \alpha_2, \alpha_1 \rangle \\ &\qquad\qquad \langle \mathsf{b}_1, \mathsf{b}_2, \beta_1, \beta_2 \rangle := \mathsf{cond}\ (\mathsf{inc_2}\ \mathsf{f_c}) \longrightarrow \langle \mathsf{b}_1, \mathsf{b}_2, \beta_1, \beta_2 \rangle \\ &\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathsf{else} \longrightarrow \langle \mathsf{b}_2, \mathsf{b}_1, \beta_2, \beta_1 \rangle \\ &\qquad\quad \mathsf{in}\ \llbracket \overline{\mathsf{f_c}}\ \langle \mathsf{a}_1, \mathsf{b}_1 \rangle, \overline{\mathsf{f_c}}\ \langle \mathsf{a}_2, \mathsf{b}_2 \rangle, \alpha_1\ \mathsf{and}\ \beta_1, \alpha_2\ \mathsf{and}\ \beta_2 \rrbracket \end{aligned} \tag{12}$$

Proof. Because $\mathsf{f_c}$ is continuous and $\mathsf{A}' \times \mathsf{B}'$ is a connected set, C' is a connected set, which in $\mathbb{R}$ is an interval. Thus, we need to determine only its bounds, and whether it contains each endpoint.

Suppose $\mathsf{f_c}$ is increasing. By monotonicity, C' is bounded by $\overline{\mathsf{f_c}}\ \langle \mathsf{a}_1, \mathsf{b}_1 \rangle$ on the bottom and $\overline{\mathsf{f_c}}\ \langle \mathsf{a}_2, \mathsf{b}_2 \rangle]$ on the top. It is therefore contained in $[\overline{\mathsf{f_c}}\ \langle \mathsf{a}_1, \mathsf{b}_1 \rangle, \overline{\mathsf{f_c}}\ \langle \mathsf{a}_2, \mathsf{b}_2 \rangle]$. If α_1 is false, C' cannot contain $\overline{\mathsf{f_c}}\ \langle \mathsf{a}_1, \mathsf{b}_1 \rangle$, and if β_1 is false, it cannot contain $\overline{\mathsf{f_c}}\ \langle \mathsf{a}_1, \mathsf{b}_1 \rangle$. Similarly, if either α_2 or β_2 is false, it cannot contain $\overline{\mathsf{f_c}}\ \langle \mathsf{a}_2, \mathsf{b}_2 \rangle$. Thus, $\mathsf{C}' = \llbracket \overline{\mathsf{f_c}}\ \langle \mathsf{a}_1, \mathsf{b}_1 \rangle, \overline{\mathsf{f_c}}\ \langle \mathsf{a}_2, \mathsf{b}_2 \rangle, \alpha_1\ \mathsf{and}\ \beta_1, \alpha_2\ \mathsf{and}\ \beta_2 \rrbracket$.

We still must prove $\langle \mathsf{a}_1, \mathsf{b}_1 \rangle$ and $\langle \mathsf{a}_2, \mathsf{b}_2 \rangle$ are in $\overline{\mathsf{f_c}}$'s domain. First, recall $\overline{\mathsf{f_c}} : (\overline{\mathsf{A}} \times \overline{\mathsf{B}}) \backslash \mathsf{N} \to \overline{\mathsf{C}}$, where $\overline{\mathsf{A}}$, $\overline{\mathsf{B}}$ and $\overline{\mathsf{C}}$ are the closures of A, B and C in $\overline{\mathbb{R}}$, and $\mathsf{N} = \{\langle \min \overline{\mathsf{A}}, \max \overline{\mathsf{B}} \rangle, \langle \max \overline{\mathsf{A}}, \min \overline{\mathsf{B}} \rangle\}$.

Because $\mathsf{A}' \subseteq \mathsf{A}$ and $\mathsf{B}' \subseteq \mathsf{A}$, $\mathsf{a}_1 \neq \max \overline{\mathsf{A}}$, $\mathsf{a}_2 \neq \min \overline{\mathsf{A}}$, $\mathsf{b}_1 \neq \max \overline{\mathsf{B}}$, and $\mathsf{b}_2 \neq \min \overline{\mathsf{B}}$, so

$$\begin{aligned} \langle \mathsf{a}_1, \mathsf{b}_1 \rangle &\neq \langle \max \overline{\mathsf{A}}, \mathsf{b} \rangle \quad \text{for all } \mathsf{b} \in \overline{\mathsf{B}} \\ \langle \mathsf{a}_1, \mathsf{b}_1 \rangle &\neq \langle \mathsf{a}, \max \overline{\mathsf{B}} \rangle \quad \text{for all } \mathsf{a} \in \overline{\mathsf{A}} \\ \langle \mathsf{a}_2, \mathsf{b}_2 \rangle &\neq \langle \min \overline{\mathsf{A}}, \mathsf{b} \rangle \quad \text{for all } \mathsf{b} \in \overline{\mathsf{B}} \\ \langle \mathsf{a}_2, \mathsf{b}_2 \rangle &\neq \langle \mathsf{a}, \min \overline{\mathsf{B}} \rangle \quad \text{for all } \mathsf{a} \in \overline{\mathsf{A}} \end{aligned} \tag{13}$$

Therefore, $\langle \mathsf{a}_1, \mathsf{b}_1 \rangle \notin \mathsf{N}$ and $\langle \mathsf{a}_2, \mathsf{b}_2 \rangle \notin \mathsf{N}$, as desired.

The remaining cases for $\mathsf{f_c}$ are similar. □

Example 4.2.16. Because $\mathsf{inc_1}\ \mathsf{pow_c}$ and $\mathsf{not}\ (\mathsf{inc_2}\ \mathsf{pow_c})$,

$$\begin{aligned} &\mathsf{image}\ \mathsf{pow_c}\ ((0, \tfrac{1}{2}] \times [2, +\infty)) \\ &= \mathsf{let}\ \langle \mathsf{a}_1, \mathsf{a}_2, \alpha_1, \alpha_2 \rangle := \langle 0, \tfrac{1}{2}, \mathsf{false}, \mathsf{true} \rangle \\ &\qquad\ \langle \mathsf{b}_1, \mathsf{b}_2, \beta_1, \beta_2 \rangle := \langle +\infty, 2, \mathsf{false}, \mathsf{true} \rangle \\ &\qquad \mathsf{in}\ \llbracket \overline{\mathsf{pow_c}}\ \langle \mathsf{a}_1, \mathsf{b}_1 \rangle, \overline{\mathsf{pow_c}}\ \langle \mathsf{a}_2, \mathsf{b}_2 \rangle, \alpha_1\ \mathsf{and}\ \beta_1, \alpha_2\ \mathsf{and}\ \beta_2 \rrbracket \\ &= \llbracket \overline{\mathsf{pow_c}}\ \langle 0, +\infty \rangle, \overline{\mathsf{pow_c}}\ \langle \tfrac{1}{2}, 2 \rangle, \mathsf{false}\ \mathsf{and}\ \mathsf{false}, \mathsf{true}\ \mathsf{and}\ \mathsf{true} \rrbracket \\ &= \llbracket 0, \tfrac{1}{4}, \mathsf{false}, \mathsf{true} \rrbracket\ =\ (0, \tfrac{1}{4}] \end{aligned}$$

♢

To use Theorem 4.2.15 to compute approximate preimages under $\mathsf{f_c}$ by computing images under its axial inverses, we must know whether each axis of $\mathsf{f_a}$ and $\mathsf{f_b}$ is increasing or decreasing. It helps to have an analogue of Lemma 4.1.6 (inverse direction).

Theorem 4.2.17. *Let* $\mathsf{f_c} : \mathsf{A} \times \mathsf{B} \to \mathsf{C}$ *be uniformly surjective and strictly increasing or decreasing in each axis, with axial inverses* $\mathsf{f_a}$ *and* $\mathsf{f_b}$*. The following statements hold:*

1. $\mathsf{inc_2}\ \mathsf{f_a}$ *if and only if* $\mathsf{inc_1}\ \mathsf{f_c}$
2. $\mathsf{inc_1}\ \mathsf{f_a}$ *if and only if* $(\mathsf{inc_1}\ \mathsf{f_c})\ \mathsf{xor}\ (\mathsf{inc_2}\ \mathsf{f_c})$

Proof. For statement 1, fix $\mathsf{b} \in \mathsf{B}$ and apply Lemma 4.1.6.

For statement 2, let $\mathsf{c} \in \mathsf{C}$, $\mathsf{b}_1, \mathsf{b}_2 \in \mathsf{B}$, $\mathsf{a}_1 := \mathsf{f_a}\ \langle \mathsf{b}_1, \mathsf{c} \rangle$ and $\mathsf{a}_2 := \mathsf{f_a}\ \langle \mathsf{b}_2, \mathsf{c} \rangle$. Let $\mathsf{c}' := \mathsf{f_c}\ \langle \mathsf{a}_1, \mathsf{b}_2 \rangle$; note $\mathsf{c} = \mathsf{f_c}\ \langle \mathsf{a}_1, \mathsf{b}_1 \rangle = \mathsf{f_c}\ \langle \mathsf{a}_2, \mathsf{b}_2 \rangle$. If $\mathsf{inc_1}\ \mathsf{f_c}$ and $\mathsf{inc_2}\ \mathsf{f_c}$, then $\mathsf{a}_1 > \mathsf{a}_2 \iff \mathsf{c} < \mathsf{c}'$ and $\mathsf{b}_1 < \mathsf{b}_2 \iff \mathsf{c} < \mathsf{c}'$; therefore $\mathsf{b}_1 < \mathsf{b}_2 \iff \mathsf{a}_1 > \mathsf{a}_2$. The remaining cases are similar. □

By Lemma 4.2.3, $\mathsf{inc}_1\ \mathsf{f_b}$ if and only if $\mathsf{inc}_2\ \mathsf{f_c}$, and $\mathsf{inc}_2\ \mathsf{f_b}$ if and only if $\mathsf{inc}_1\ \mathsf{f_a}$. We can therefore determine the uniform directions of $\mathsf{f_a}$'s and $\mathsf{f_b}$'s axes solely from the uniform directions of $\mathsf{f_c}$'s axes.

4.3 Discontinuous Primitives

4.4 Primitive Implementation

Because floating-point functions are defined on subsets of $\overline{\mathbb{R}}$, it would seem we could compute preimages under strictly monotone, real functions by applying their floating-point counterparts to interval endpoints. This is mostly true, but we must take care to round in the right directions and to account for floating-point negative zero and not-a-number results.

$$\begin{aligned}
&\overline{\mathsf{pos\text{-}recip}} : [0, +\infty] \to [0, +\infty] \\
&\overline{\mathsf{pos\text{-}recip}}\ \mathsf{a} \;=\; \mathsf{cond}\ \begin{array}{ll} 0 < \mathsf{a} < +\infty & \longrightarrow\ 1/\mathsf{a} \\ \mathsf{a} = 0 & \longrightarrow\ +\infty \\ \mathsf{a} = +\infty & \longrightarrow\ 0 \end{array}
\end{aligned} \tag{14}$$

5. Application: Floating-Point Error Analysis

6. Recursive, Probabilistic Program Semantics

7. Application: Sampling With Constraints

8. Related Work

9. Conclusions and Future Work

References

[1] I. Sergey, D. Devriese, M. Might, J. Midtgaard, D. Darais, D. Clarke, and F. Piessens. Monadic abstract interpreters. In *Programming Language Design and Implementation*, pages 399–410, 2013.

$$
\begin{aligned}
p &::\equiv x := e;\ \ldots\ ; e \\
e &::\equiv x\ e \mid \mathsf{let}\ e\ e \mid \mathsf{env}\ n \mid \langle e, e\rangle \mid \mathsf{fst}\ e \mid \mathsf{snd}\ e \mid \mathsf{if}\ e\ e\ e \mid v \\
v &::\equiv \text{[first-order constants]}
\end{aligned}
$$

$$
[\![x := e;\ \ldots\ ; e_b]\!]_{\mathsf{a}} :\equiv x := [\![e]\!]_{\mathsf{a}}\ ;\ \ldots\ ;\ [\![e_b]\!]_{\mathsf{a}}
$$

$$
\begin{aligned}
[\![x\ e]\!]_{\mathsf{a}} &:\equiv [\![\langle e, \langle\rangle\rangle]\!]_{\mathsf{a}} \ggg_{\mathsf{a}} x \\
[\![\langle e_1, e_2\rangle]\!]_{\mathsf{a}} &:\equiv [\![e_1]\!]_{\mathsf{a}}\ \&\&\&_{\mathsf{a}}\ [\![e_2]\!]_{\mathsf{a}} \\
[\![\mathsf{fst}\ e]\!]_{\mathsf{a}} &:\equiv [\![e]\!]_{\mathsf{a}} \ggg_{\mathsf{a}} \mathsf{arr}_{\mathsf{a}}\ \mathsf{fst} \\
[\![\mathsf{snd}\ e]\!]_{\mathsf{a}} &:\equiv [\![e]\!]_{\mathsf{a}} \ggg_{\mathsf{a}} \mathsf{arr}_{\mathsf{a}}\ \mathsf{snd} \\
[\![v]\!]_{\mathsf{a}} &:\equiv \mathsf{arr}_{\mathsf{a}}\ (\mathsf{const}\ v)
\end{aligned}
\qquad
\begin{aligned}
[\![\mathsf{let}\ e\ e_b]\!]_{\mathsf{a}} &:\equiv ([\![e]\!]_{\mathsf{a}}\ \&\&\&_{\mathsf{a}}\ \mathsf{arr}_{\mathsf{a}}\ \mathsf{id}) \ggg_{\mathsf{a}} [\![e_b]\!]_{\mathsf{a}} \\
[\![\mathsf{env}\ 0]\!]_{\mathsf{a}} &:\equiv \mathsf{arr}_{\mathsf{a}}\ \mathsf{fst} \\
[\![\mathsf{env}\ (n+1)]\!]_{\mathsf{a}} &:\equiv \mathsf{arr}_{\mathsf{a}}\ \mathsf{snd} \ggg_{\mathsf{a}} [\![\mathsf{env}\ n]\!]_{\mathsf{a}} \\
[\![\mathsf{if}\ e_c\ e_t\ e_f]\!]_{\mathsf{a}} &:\equiv \mathsf{ifte}_{\mathsf{a}}\ [\![e_c]\!]_{\mathsf{a}}\ [\![\mathsf{lazy}\ e_t]\!]_{\mathsf{a}}\ [\![\mathsf{lazy}\ e_f]\!]_{\mathsf{a}} \\
[\![\mathsf{lazy}\ e]\!]_{\mathsf{a}} &:\equiv \mathsf{lazy}_{\mathsf{a}}\ \lambda 0.\ [\![e]\!]_{\mathsf{a}}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{id} &:= \lambda \mathsf{a}.\ \mathsf{a} \\
\mathsf{const}\ \mathsf{b} &:= \lambda \mathsf{a}.\ \mathsf{b}
\end{aligned}
\qquad
\text{subject to } [\![p]\!]_{\mathsf{a}} : \langle\rangle \rightsquigarrow_{\mathsf{a}} \mathsf{y} \text{ for some y}
$$

Figure 1: Interpretation of a let-calculus with first-order definitions and De-Bruijn-indexed bindings as arrow a computations.

$$
\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y} ::= \mathsf{X} \Rightarrow \mathsf{Y}_{\perp}
$$

$$
\begin{aligned}
&\mathsf{arr}_{\perp} : (\mathsf{X} \Rightarrow \mathsf{Y}) \Rightarrow (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y}) \\
&\mathsf{arr}_{\perp}\ \mathsf{f} := \mathsf{f}
\end{aligned}
$$

$$
\begin{aligned}
&(\ggg_{\perp}) : (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y}) \Rightarrow (\mathsf{Y} \rightsquigarrow_{\perp} \mathsf{Z}) \Rightarrow (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Z}) \\
&(\mathsf{f}_1 \ggg_{\perp} \mathsf{f}_2)\ \mathsf{a} := \mathsf{if}\ (\mathsf{f}_1\ \mathsf{a} = \perp)\ \perp\ (\mathsf{f}_2\ (\mathsf{f}_1\ \mathsf{a}))
\end{aligned}
$$

$$
\begin{aligned}
&(\&\&\&_{\perp}) : (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y}_1) \Rightarrow (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y}_2) \Rightarrow (\mathsf{X} \rightsquigarrow_{\perp} \langle \mathsf{Y}_1, \mathsf{Y}_2\rangle) \\
&(\mathsf{f}_1\ \&\&\&_{\perp}\ \mathsf{f}_2)\ \mathsf{a} := \mathsf{if}\ (\mathsf{f}_1\ \mathsf{a} = \perp \text{ or } \mathsf{f}_2\ \mathsf{a} = \perp)\ \perp\ \langle \mathsf{f}_1\ \mathsf{a}, \mathsf{f}_2\ \mathsf{a}\rangle
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{ifte}_{\perp} : (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Bool}) \Rightarrow (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y}) \Rightarrow \\
&\qquad (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y}) \Rightarrow (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y}) \\
&\mathsf{ifte}_{\perp}\ \mathsf{f}_1\ \mathsf{f}_2\ \mathsf{f}_3\ \mathsf{a} := \\
&\quad \mathsf{case}\ \mathsf{f}_1\ \mathsf{a} \\
&\qquad \mathsf{true} \longrightarrow \mathsf{f}_2\ \mathsf{a} \\
&\qquad \mathsf{false} \longrightarrow \mathsf{f}_3\ \mathsf{a} \\
&\qquad \perp \longrightarrow \perp
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{lazy}_{\perp} : (1 \Rightarrow (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y})) \Rightarrow (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y}) \\
&\mathsf{lazy}_{\perp}\ \mathsf{f}\ \mathsf{a} := \mathsf{f}\ 0\ \mathsf{a}
\end{aligned}
$$

Figure 2: Bottom arrow definitions.

$$
\mathsf{X} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Y} ::= \langle \mathsf{Set}\ \mathsf{Y}, \mathsf{Set}\ \mathsf{Y} \Rightarrow \mathsf{Set}\ \mathsf{X}\rangle
$$

$$
\begin{aligned}
&\mathsf{pre} : (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Y}) \\
&\mathsf{pre}\ \mathsf{f}\ \mathsf{A} := \\
&\quad \langle \mathsf{image}_{\perp}\ \mathsf{f}\ \mathsf{A}, \lambda \mathsf{B}.\ \mathsf{preimage}_{\perp}\ \mathsf{f}\ \mathsf{A}\ \mathsf{B}\rangle
\end{aligned}
$$

$$
\varnothing_{\mathsf{pre}} := \langle \varnothing, \lambda \mathsf{B}.\ \varnothing\rangle
$$

$$
\begin{aligned}
&\mathsf{ap}_{\mathsf{pre}} : (\mathsf{X} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Y}) \Rightarrow \mathsf{Set}\ \mathsf{Y} \Rightarrow \mathsf{Set}\ \mathsf{X} \\
&\mathsf{ap}_{\mathsf{pre}}\ \langle \mathsf{Y}', \mathsf{p}\rangle\ \mathsf{B} := \mathsf{p}\ (\mathsf{B} \cap \mathsf{Y}')
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{range}_{\mathsf{pre}} : (\mathsf{X} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Y}) \Rightarrow \mathsf{Set}\ \mathsf{Y} \\
&\mathsf{range}_{\mathsf{pre}}\ \langle \mathsf{Y}', \mathsf{p}\rangle := \mathsf{Y}'
\end{aligned}
$$

$$
\begin{aligned}
&\langle \cdot, \cdot\rangle_{\mathsf{pre}} : (\mathsf{X} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Y}_1) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Y}_2) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Y}_1 \times \mathsf{Y}_2) \\
&\langle \langle \mathsf{Y}_1', \mathsf{p}_1\rangle, \langle \mathsf{Y}_2', \mathsf{p}_2\rangle\rangle_{\mathsf{pre}} := \\
&\quad \mathsf{let}\ \mathsf{Y}' := \mathsf{Y}_1' \times \mathsf{Y}_2' \\
&\qquad \mathsf{p} := \lambda \mathsf{B}.\ \bigcup_{\langle \mathsf{b}_1, \mathsf{b}_2\rangle \in \mathsf{B}} (\mathsf{p}_1\ \{\mathsf{b}_1\}) \cap (\mathsf{p}_2\ \{\mathsf{b}_2\}) \\
&\quad \mathsf{in}\ \langle \mathsf{Y}', \mathsf{p}\rangle
\end{aligned}
$$

$$
\begin{aligned}
&(\circ_{\mathsf{pre}}) : (\mathsf{Y} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Z}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Y}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Z}) \\
&\langle \mathsf{Z}', \mathsf{p}_2\rangle \circ_{\mathsf{pre}} \mathsf{h}_1 := \langle \mathsf{Z}', \lambda \mathsf{C}.\ \mathsf{ap}_{\mathsf{pre}}\ \mathsf{h}_1\ (\mathsf{p}_2\ \mathsf{C})\rangle
\end{aligned}
$$

$$
\begin{aligned}
&(\uplus_{\mathsf{pre}}) : (\mathsf{X} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Y}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Y}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightarrow} \mathsf{Y}) \\
&\mathsf{h}_1 \uplus_{\mathsf{pre}} \mathsf{h}_2 := \mathsf{let}\ \mathsf{Y}' := (\mathsf{range}_{\mathsf{pre}}\ \mathsf{h}_1) \cup (\mathsf{range}_{\mathsf{pre}}\ \mathsf{h}_2) \\
&\qquad\qquad \mathsf{p} := \lambda \mathsf{B}.\ (\mathsf{ap}_{\mathsf{pre}}\ \mathsf{h}_1\ \mathsf{B}) \cup (\mathsf{ap}_{\mathsf{pre}}\ \mathsf{h}_2\ \mathsf{B}) \\
&\qquad\qquad \mathsf{in}\ \langle \mathsf{Y}', \mathsf{p}\rangle
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{image}_{\perp} : (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y}) \Rightarrow \mathsf{Set}\ \mathsf{X} \Rightarrow \mathsf{Set}\ \mathsf{Y} \\
&\mathsf{image}_{\perp}\ \mathsf{f}\ \mathsf{A} := (\mathsf{image}\ \mathsf{f}\ \mathsf{A}) \backslash \{\perp\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{preimage}_{\perp} : (\mathsf{X} \rightsquigarrow_{\perp} \mathsf{Y}) \Rightarrow \mathsf{Set}\ \mathsf{X} \Rightarrow \mathsf{Set}\ \mathsf{Y} \Rightarrow \mathsf{Set}\ \mathsf{X} \\
&\mathsf{preimage}_{\perp}\ \mathsf{f}\ \mathsf{A}\ \mathsf{B} := \{\mathsf{a} \in \mathsf{A} \mid \mathsf{f}\ \mathsf{a} \in \mathsf{B}\}
\end{aligned}
$$

Figure 3: Lazy preimage mappings and operations.

$$X \underset{pre}{\rightsquigarrow} Y ::= Set\ X \Rightarrow (X \underset{pre}{\rightarrow} Y)$$

$$arr_{pre} : (X \Rightarrow Y) \Rightarrow (X \underset{pre}{\rightsquigarrow} Y)$$
$$arr_{pre} := lift_{pre} \circ arr_{\perp}$$

$$(\ggg_{pre}) : (X \underset{pre}{\rightsquigarrow} Y) \Rightarrow (Y \underset{pre}{\rightsquigarrow} Z) \Rightarrow (X \underset{pre}{\rightsquigarrow} Z)$$
$$(h_1 \ggg_{pre} h_2)\ A := \text{let } h_1' := h_1\ A \quad h_2' := h_2\ (range_{pre}\ h_1') \text{ in } h_2' \circ_{pre} h_1'$$

$$(\&\&\&_{pre}) : (X \underset{pre}{\rightsquigarrow} Y) \Rightarrow (X \underset{pre}{\rightsquigarrow} Z) \Rightarrow (X \underset{pre}{\rightsquigarrow} Y \times Z)$$
$$(h_1 \&\&\&_{pre} h_2)\ A := \langle h_1\ A, h_2\ A \rangle_{pre}$$

$$ifte_{pre} : (X \underset{pre}{\rightsquigarrow} Bool) \Rightarrow (X \underset{pre}{\rightsquigarrow} Y) \Rightarrow (X \underset{pre}{\rightsquigarrow} Y) \Rightarrow (X \underset{pre}{\rightsquigarrow} Y)$$
$$ifte_{pre}\ h_1\ h_2\ h_3\ A := \text{let } h_1' := h_1\ A \quad h_2' := h_2\ (ap_{pre}\ h_1'\ \{true\}) \quad h_3' := h_3\ (ap_{pre}\ h_1'\ \{false\}) \text{ in } h_2' \uplus_{pre} h_3'$$

$$lazy_{pre} : (1 \Rightarrow (X \underset{pre}{\rightsquigarrow} Y)) \Rightarrow (X \underset{pre}{\rightsquigarrow} Y)$$
$$lazy_{pre}\ h\ A := \text{if } (A = \varnothing)\ \varnothing_{pre}\ (h\ 0\ A)$$

$$lift_{pre} := pre$$

Figure 4: Preimage arrow definitions.

$$x \rightsquigarrow_{a^*} y ::= AStore\ s\ (x \rightsquigarrow_a y) ::= J \Rightarrow (\langle s, x \rangle \rightsquigarrow_a y)$$

$$arr_{a^*} : (x \Rightarrow y) \Rightarrow (x \rightsquigarrow_{a^*} y)$$
$$arr_{a^*} := \eta_{a^*} \circ arr_a$$

$$(\ggg_{a^*}) : (x \rightsquigarrow_{a^*} y) \Rightarrow (y \rightsquigarrow_{a^*} z) \Rightarrow (x \rightsquigarrow_{a^*} z)$$
$$(k_1 \ggg_{a^*} k_2)\ j := (arr_a\ fst \&\&\&_a k_1\ (left\ j)) \ggg_a k_2\ (right\ j)$$

$$(\&\&\&_{a^*}) : (x \rightsquigarrow_{a^*} y_1) \Rightarrow (x \rightsquigarrow_{a^*} y_2) \Rightarrow (x \rightsquigarrow_{a^*} \langle y_1, y_2 \rangle)$$
$$(k_1 \&\&\&_{a^*} k_2)\ j := k_1\ (left\ j) \&\&\&_a k_2\ (right\ j)$$

$$ifte_{a^*} : (x \rightsquigarrow_{a^*} Bool) \Rightarrow (x \rightsquigarrow_{a^*} y) \Rightarrow (x \rightsquigarrow_{a^*} y) \Rightarrow (x \rightsquigarrow_{a^*} y)$$
$$ifte_{a^*}\ k_1\ k_2\ k_3\ j := ifte_a\ (k_1\ (left\ j))\ (k_2\ (left\ (right\ j)))\ (k_3\ (right\ (right\ j)))$$

$$lazy_{a^*} : (1 \Rightarrow (x \rightsquigarrow_{a^*} y)) \Rightarrow (x \rightsquigarrow_{a^*} y)$$
$$lazy_{a^*}\ k\ j := lazy_a\ \lambda 0.\ k\ 0\ j$$

$$\eta_{a^*} : (x \rightsquigarrow_a y) \Rightarrow (x \rightsquigarrow_{a^*} y)$$
$$\eta_{a^*}\ f\ j := arr_a\ snd \ggg_a f$$

Figure 5: AStore (associative store) arrow transformer definitions.

$$id_{pre}\ A := \langle A, \lambda B.\ B \rangle$$
$$fst_{pre}\ A := \langle proj_1\ A, unproj_1\ A \rangle$$
$$snd_{pre}\ A := \langle proj_2\ A, unproj_2\ A \rangle$$

$$proj_1 := image\ fst$$
$$proj_2 := image\ snd$$
$$unproj_1\ A\ B := A \cap (B \times proj_2\ A)$$
$$unproj_2\ A\ B := A \cap (proj_1\ A \times B)$$

$$const_{pre}\ b\ A := \langle \{b\}, \lambda B.\ \text{if } (B = \varnothing)\ \varnothing\ A \rangle$$
$$\pi_{pre}\ j\ A := \langle proj\ j\ A, unproj\ j\ A \rangle$$

$$proj : J \Rightarrow Set\ (J \to X) \Rightarrow Set\ X$$
$$proj\ j\ A := image\ (\pi\ j)\ A$$

$$unproj : J \Rightarrow Set\ (J \to X) \Rightarrow Set\ X \Rightarrow Set\ (J \to X)$$
$$unproj\ j\ A\ B := A \cap \prod_{i \in J} \text{if } (j = i)\ B\ (proj\ j\ A)$$

Figure 6: Preimage arrow lifts needed to interpret probabilistic programs.

$$\mathsf{X} \underset{\mathsf{pre}}{\rightharpoonup}' \mathsf{Y} ::= \langle \mathsf{Rect\ Y}, \mathsf{Rect\ Y} \Rightarrow \mathsf{Rect\ X} \rangle$$

$$\varnothing'_{\mathsf{pre}} := \langle \varnothing, \lambda \mathsf{B}.\, \varnothing \rangle$$

$$\mathsf{ap}'_{\mathsf{pre}} : (\mathsf{X} \underset{\mathsf{pre}}{\rightharpoonup}' \mathsf{Y}) \Rightarrow \mathsf{Rect\ Y} \Rightarrow \mathsf{Rect\ X}$$
$$\mathsf{ap}'_{\mathsf{pre}}\ \langle \mathsf{Y}', \mathsf{p} \rangle\ \mathsf{B} := \mathsf{p}\ (\mathsf{B} \cap \mathsf{Y}')$$

$$(\circ'_{\mathsf{pre}}) : (\mathsf{Y} \underset{\mathsf{pre}}{\rightharpoonup}' \mathsf{Z}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightharpoonup}' \mathsf{Y}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightharpoonup}' \mathsf{Z})$$
$$\langle \mathsf{Z}', \mathsf{p}_2 \rangle \circ'_{\mathsf{pre}} \mathsf{h}_1 := \langle \mathsf{Z}', \lambda \mathsf{C}.\, \mathsf{ap}'_{\mathsf{pre}}\ \mathsf{h}_1\ (\mathsf{p}_2\ \mathsf{C}) \rangle$$

$$\langle \cdot, \cdot \rangle'_{\mathsf{pre}} : (\mathsf{X} \underset{\mathsf{pre}}{\rightharpoonup}' \mathsf{Y}_1) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightharpoonup}' \mathsf{Y}_2) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightharpoonup}' \mathsf{Y}_1 \times \mathsf{Y}_2)$$
$$\langle \langle \mathsf{Y}'_1, \mathsf{p}_1 \rangle, \langle \mathsf{Y}'_2, \mathsf{p}_2 \rangle \rangle'_{\mathsf{pre}} :=$$
$$\quad \langle \mathsf{Y}'_1 \times \mathsf{Y}'_2, \lambda \mathsf{B}.\, \mathsf{p}_1\ (\mathsf{proj}_1\ \mathsf{B}) \cap \mathsf{p}_2\ (\mathsf{proj}_2\ \mathsf{B}) \rangle$$

$$(\uplus'_{\mathsf{pre}}) : (\mathsf{X} \underset{\mathsf{pre}}{\rightharpoonup}' \mathsf{Y}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightharpoonup}' \mathsf{Y}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightharpoonup}' \mathsf{Y})$$
$$\langle \mathsf{Y}'_1, \mathsf{p}_1 \rangle \uplus'_{\mathsf{pre}} \langle \mathsf{Y}'_2, \mathsf{p}_2 \rangle :=$$
$$\quad \langle \mathsf{Y}'_1 \vee \mathsf{Y}'_2, \lambda \mathsf{B}.\, \mathsf{ap}'_{\mathsf{pre}}\ \langle \mathsf{Y}'_1, \mathsf{p}_1 \rangle\ \mathsf{B} \vee \mathsf{ap}'_{\mathsf{pre}}\ \langle \mathsf{Y}'_2, \mathsf{p}_2 \rangle\ \mathsf{B} \rangle$$

(a) Definitions for preimage mappings that compute rectangular covers.

$$\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Y} ::= \mathsf{Rect\ X} \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightharpoonup}' \mathsf{Y})$$

$$(\ggg'_{\mathsf{pre}}) : (\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Y}) \Rightarrow (\mathsf{Y} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Z}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Z})$$
$$(\mathsf{h}_1 \ggg'_{\mathsf{pre}} \mathsf{h}_2)\ \mathsf{A} := \mathsf{let}\ \mathsf{h}'_1 := \mathsf{h}_1\ \mathsf{A}$$
$$\qquad\qquad \mathsf{h}'_2 := \mathsf{h}_2\ (\mathsf{range}'_{\mathsf{pre}}\ \mathsf{h}'_1)$$
$$\qquad \mathsf{in}\ \ \mathsf{h}'_2 \circ'_{\mathsf{pre}} \mathsf{h}'_1$$

$$(\&\&\&'_{\mathsf{pre}}) : (\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Y}_1) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Y}_2) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \langle \mathsf{Y}_1, \mathsf{Y}_2 \rangle)$$
$$(\mathsf{h}_1 \&\&\&'_{\mathsf{pre}} \mathsf{h}_2)\ \mathsf{A} := \langle \mathsf{h}_1\ \mathsf{A}, \mathsf{h}_2\ \mathsf{A} \rangle'_{\mathsf{pre}}$$

$$\mathsf{ifte}'_{\mathsf{pre}} : (\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Bool}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Y}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Y}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Y})$$
$$\mathsf{ifte}'_{\mathsf{pre}}\ \mathsf{h}_1\ \mathsf{h}_2\ \mathsf{h}_3\ \mathsf{A} :=$$
$$\quad \mathsf{let}\ \mathsf{h}'_1 := \mathsf{h}_1\ \mathsf{A}$$
$$\qquad \mathsf{h}'_2 := \mathsf{h}_2\ (\mathsf{ap}'_{\mathsf{pre}}\ \mathsf{h}'_1\ \{\mathsf{true}\})$$
$$\qquad \mathsf{h}'_3 := \mathsf{h}_3\ (\mathsf{ap}'_{\mathsf{pre}}\ \mathsf{h}'_1\ \{\mathsf{false}\})$$
$$\quad \mathsf{in}\ \ \mathsf{h}'_2 \uplus'_{\mathsf{pre}} \mathsf{h}'_3$$

$$\mathsf{lazy}'_{\mathsf{pre}} : (1 \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Y})) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Y})$$
$$\mathsf{lazy}'_{\mathsf{pre}}\ \mathsf{h}\ \mathsf{A} := \mathsf{if}\ (\mathsf{A} = \varnothing)\ \varnothing'_{\mathsf{pre}}\ (\mathsf{h}\ 0\ \mathsf{A})$$

(b) Approximating preimage arrow, defined using approximating preimage mappings.

$$\mathsf{X} \underset{\mathsf{pre}^*}{\rightsquigarrow}' \mathsf{Y} ::= \mathsf{AStore}\ (\mathsf{R} \times \mathsf{T})\ (\mathsf{X} \underset{\mathsf{pre}}{\rightsquigarrow}' \mathsf{Y})$$

$$\mathsf{random}'_{\mathsf{pre}^*} : \mathsf{X} \underset{\mathsf{pre}^*}{\rightsquigarrow}' [0, 1]$$
$$\mathsf{random}'_{\mathsf{pre}^*}\ \mathsf{j} :=$$
$$\quad \mathsf{fst}_{\mathsf{pre}} \ggg'_{\mathsf{pre}} \mathsf{fst}_{\mathsf{pre}} \ggg'_{\mathsf{pre}} \pi_{\mathsf{pre}}\ \mathsf{j}$$

$$\mathsf{branch}'_{\mathsf{pre}^*} : \mathsf{X} \underset{\mathsf{pre}^*}{\rightsquigarrow}' \mathsf{Bool}$$
$$\mathsf{branch}'_{\mathsf{pre}^*}\ \mathsf{j} :=$$
$$\quad \mathsf{fst}_{\mathsf{pre}} \ggg'_{\mathsf{pre}} \mathsf{snd}_{\mathsf{pre}} \ggg'_{\mathsf{pre}} \pi_{\mathsf{pre}}\ \mathsf{j}$$

$$\mathsf{fst}'_{\mathsf{pre}^*} := \eta'_{\mathsf{pre}^*}\ \mathsf{fst}_{\mathsf{pre}}; \cdots$$

$$\mathsf{ifte}^{\Downarrow\prime}_{\mathsf{pre}^*} : (\mathsf{X} \underset{\mathsf{pre}^*}{\rightsquigarrow}' \mathsf{Bool}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}^*}{\rightsquigarrow}' \mathsf{Y}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}^*}{\rightsquigarrow}' \mathsf{Y}) \Rightarrow (\mathsf{X} \underset{\mathsf{pre}^*}{\rightsquigarrow}' \mathsf{Y})$$
$$\mathsf{ifte}^{\Downarrow\prime}_{\mathsf{pre}^*}\ \mathsf{k}_1\ \mathsf{k}_2\ \mathsf{k}_3\ \mathsf{j} :=$$
$$\quad \mathsf{let}\ \langle \mathsf{C}_\mathsf{k}, \mathsf{p}_\mathsf{k} \rangle := \mathsf{k}_1\ (\mathsf{left\ j})\ \mathsf{A}$$
$$\qquad \langle \mathsf{C}_\mathsf{b}, \mathsf{p}_\mathsf{b} \rangle := \mathsf{branch}_{\mathsf{pre}^*}\ \mathsf{j}\ \mathsf{A}$$
$$\qquad\qquad \mathsf{C}_2 := \mathsf{C}_\mathsf{k} \cap \mathsf{C}_\mathsf{b} \cap \{\mathsf{true}\}$$
$$\qquad\qquad \mathsf{C}_3 := \mathsf{C}_\mathsf{k} \cap \mathsf{C}_\mathsf{b} \cap \{\mathsf{false}\}$$
$$\qquad\qquad \mathsf{A}_2 := \mathsf{p}_\mathsf{k}\ \mathsf{C}_2 \cap \mathsf{p}_\mathsf{b}\ \mathsf{C}_2$$
$$\qquad\qquad \mathsf{A}_3 := \mathsf{p}_\mathsf{k}\ \mathsf{C}_3 \cap \mathsf{p}_\mathsf{b}\ \mathsf{C}_3$$
$$\quad \mathsf{in}\ \ \mathsf{if}\ (\mathsf{C}_\mathsf{b} = \{\mathsf{true}, \mathsf{false}\})$$
$$\qquad \langle \top, \lambda \mathsf{B}.\, \mathsf{A}_2 \vee \mathsf{A}_3 \rangle$$
$$\qquad (\mathsf{k}_2\ (\mathsf{left}\ (\mathsf{right\ j}))\ \mathsf{A}_2 \uplus'_{\mathsf{pre}}\ \mathsf{k}_3\ (\mathsf{right}\ (\mathsf{right\ j}))\ \mathsf{A}_3)$$

(c) Preimage* arrow combinators for probabilistic choice and guaranteed termination. Fig. 5 (AStore arrow transformer) defines η'_{pre^*}, $(\ggg'_{\mathsf{pre}^*})$, $(\&\&\&'_{\mathsf{pre}^*})$, $\mathsf{ifte}'_{\mathsf{pre}^*}$ and $\mathsf{lazy}'_{\mathsf{pre}^*}$.

Figure 7: Implementable arrows that approximate preimage arrows. Specific lifts such as $\mathsf{fst}_{\mathsf{pre}} := \mathsf{arr}_{\mathsf{pre}}\ \mathsf{fst}$ are computable (see Fig. 6), but $\mathsf{arr}'_{\mathsf{pre}}$ is not.