Analysing Functions by Projection-Based Backward Abstraction

Kei Davis
Department of Computing Science, University of Glasgow
Glasgow G12 8QQ, UK

Abstract

Various techniques for strictness analysis by non-standard interpretation have been proposed in which the basic non-standard values are projections. We show that continuous functions can be completely characterised by appropriate forward and backward abstractions.

1 Introduction

Strictness analysis by non-standard (or abstract) interpretation allows significant optimisation by compilers for lazy functional languages ([1], [2], [3]). Research continues in both the development of strictness analysis techniques and their practical exploitation. In projection-based *backward* analysis, abstract values in the form of projections are propogated from the leaves of an expression to the root, and such analyses are able to determine not just ordinary strictness but more elusive properties such as head strictness, even when working in finite domains of abstract values ([4]). We are concerned with the theoretical power of such analyses, without much consideration of such practical considerations restriction to finite abstract domains; in particular we show that continuous functions can be completely determined by both forward and backward abstract values from appropriate domains.

This works brings together and expands on ideas from several sources, most notably [4], [5], and [6].

2 Projections

A *projection* is a continuous idempotent function that approximates the identity. The set of all projections on a given domain, ordered by the usual function ordering, forms an ω-algebraic complete lattice, with the identity ID as the greatest element, and the constant bottom function BOT as the least. Since the glb of a set of projections in the domain of continuous functions is not necessarily a projection, the glb in the lattice of projections is defined to be the greatest projection approximating every element of the set. This projection necessarily approximates the glb in the continuous function space. A projection is *finitary* if its image is a domain. The set of finitary projections on any domain U also forms an ω-algebraic complete lattice, and will be denoted by $|\,U\,|$. All projections herein are finitary, though no essential use is made of this fact. The symbols α, β, γ, and δ will always denote projections.

3 Projections and Strictness

Projections may be used to specify upper and lower bounds on the definedness of values—a semantic interpretation, and upper and lower bounds on the degree of evaluation of expressions—an operational interpretation. Though it is possible to formalise the operational interpretation [3], here it is used only as an informal source of intuition. We give three examples. Let `f` denote $f \in U \to V$ such that $f = f \circ BOT$. This equation makes clear that f requires no information from its argument, that is, the argument may be completely undefined; operationally this says that any argument of `f` need never be evaluated: if evaluation of an application of `f` requires evaluation of the argument, evaluation of the argument may safely diverge or return a dummy value. Here we say that f is BOT strict.

As another example, let `swap` denote *swap*, a function on pairs, such that $swap(x, y) = (y, x)$. Define projections F and S by $F = ID \times BOT$, and $S = BOT \times ID$. Then $S \circ swap = swap \circ F$, indicating that if the second component of the result of *swap* need not be defined, then the first component of its argument need not be defined. Operationally, if the second component of the result of `swap` will not be evaluated, then the first component of any argument of `swap` need never be evaluated. Here we say that *swap* is F strict in an S-strict context. In the previous example, we could have said that f was BOT strict in an ID-strict context. In both examples, projections only specified upper bounds on required definedness (by discarding unnecessary information) and therefore only upper bounds on evaluation.

Finally, consider the projection H on lists defined by

$$\begin{array}{lcl} H\ \bot & = & \bot\ , \\ H\ [] & = & []\ , \\ H\ (\bot : xs) & = & \bot\ , \\ H\ (x : xs) & = & x : (H\ xs),\ x \neq \bot\ , \end{array}$$

where $[]$ denotes the empty list and infix : denotes the cons operation. Then H is the identity on finite, partial, and infinite lists not containing bottom elements, but truncates other lists at their first bottom element. For example,

$$H\ (1:2:3:\bot:4:[])\ =\ 1:2:3:\bot\ .$$

If function f has the property $f = f \circ H$ then f is head strict; we take this as the *definition* of (semantic) head strictness. Operationally, if `f` denotes f, and f is head strict, then whenever evaluation of an application of `f` demands evaluaton of a cons cell of its argument it is safe to evaluate the head of the cons cell as well. Again, the projection H has specified only an upper bound on required definedness, though in a conditional manner: if the head of any cons cell is not defined, then the tail need not be defined either.

In projection-based backward strictness analysis, the central problem is, given γ and f, to find δ such that $\gamma \circ f = \gamma \circ f \circ \delta$, or equivalently, $\gamma \circ f \sqsubseteq f \circ \delta$. More generally, given f we seek to determine a *projection transformer* τ, a function from projections to projections, such that for all γ we have $\gamma \circ f \sqsubseteq f \circ (\tau\ \gamma)$. This inequality is the *safety condition* for τ, and any τ satisfying the safety condition will be called a *backward abstraction* of f. Taking τ to be $\lambda\alpha.ID$ is always safe, but tells nothing about f; smaller τ is more informative, and greater τ is always safe.

As the following example shows, we cannot in general hope to choose τ small enough that we get equality instead of inequality in the safety condition. Let $f \in \mathbf{1}_{\bot\bot} \rightarrow \mathbf{1}_{\bot\bot}$, and $\gamma,\ \delta \in |\,\mathbf{1}_{\bot\bot}\,|$, where $\mathbf{1}_{\bot\bot} = \{\bot,\ lift\ \bot,\ lift^2\ \bot\}$, and

$$\begin{array}{lll} f\ \bot = \bot\ , & \gamma\ \bot = \bot\ , & \delta\ \bot = lift\ \bot\ , \\ f\ (lift\ \bot) = \bot\ , & \gamma\ (lift\ \bot) = lift\ \bot\ , & \delta\ (lift\ \bot) = lift\ \bot\ , \\ f\ (lift^2\ \bot) = lift^2\ \bot\ , & \gamma\ (lift^2\ \bot) = lift\ \bot\ , & \delta\ (lift^2\ \bot) = lift^2\ \bot\ . \end{array}$$

Then δ is the least projection such that $\gamma \circ f \sqsubseteq f \circ \delta$, but

$$\begin{array}{ll} (\gamma \circ f)\ \bot = \bot\ , & (f \circ \delta)\ \bot = \bot\ , \\ (\gamma \circ f)\ (lift\ \bot) = \bot\ , & (f \circ \delta)\ (lift\ \bot) = \bot\ , \\ (\gamma \circ f)\ (lift^2\ \bot) = lift\ \bot\ , & (f \circ \delta)\ (lift^2\ \bot) = lift^2\ \bot\ , \end{array}$$

that is, $\gamma \circ f \neq f \circ \delta$.

A continuous function may not even have a *least* backward abstraction. For example, there is no least backward abstraction of *parallel or*, defined by

$$\begin{array}{lcl}
\mathit{Bool} & = & \{\mathit{true},\ \mathit{false}\}_\perp \\
\mathit{por} & \in & \mathit{Bool} \rightarrow \mathit{Bool}\ , \\
\mathit{por}\ (\perp,\ \mathit{lift\ false}) & = & \perp\ , \\
\mathit{por}\ (\mathit{lift\ false},\ \perp) & = & \perp\ , \\
\mathit{por}\ (\mathit{lift}\ u,\ \mathit{lift}\ v) & = & \mathit{lift}\ (u\ \vee\ v)\ .
\end{array}$$

To see this, let δ_1 be the greatest projection such that $\delta_1\ (\mathit{lift\ true},\ \mathit{lift\ true}) = (\mathit{lift\ true},\ \perp)$, and δ_2 be the greatest projection such that $\delta_2\ (\mathit{lift\ true},\ \mathit{lift\ true}) = (\perp,\ \mathit{lift\ true})$. Then $\mathit{ID} \circ \mathit{por} \sqsubseteq \mathit{por} \circ \delta_1$ and $\mathit{ID} \circ \mathit{por} \sqsubseteq \mathit{por} \circ \delta_2$, but since $(\delta_1 \sqcap \delta_2)\ (\mathit{lift\ true},\ \mathit{lift\ true}) = (\perp,\ \perp)$, it is not true that $\mathit{ID} \circ \mathit{por} \sqsubseteq \mathit{por} \circ (\delta_1 \sqcap \delta_2)$. As will be discussed in Section 4, existence of least backward abstractions can be guaranteed by restricting the space of functions under consideration; in the meantime, we nonetheless try to find as small backward abstractions as possible.

Even when a function does not have a least backward abstraction, it may have *minimal* backward abstractions: a backward abstraction is minimal if no other backward abstraction approximates it. For example, *por* above has two minimal backward abstractions, determined by whether *ID* is mapped to a projection that maps $(\mathit{lift\ true},\ \mathit{lift\ true})$ to $(\perp,\ \mathit{lift\ true})$, or to $(\mathit{lift\ true},\ \perp)$.

A minimal backward abstraction of a continuous function may not be monotonic. For example, the function $f = \lambda((x, y), (z, t)).(\mathit{por}\ x\ y,\ \mathit{por}\ z\ t)$ has non-monotonic minimal backward abstractions. To see this, let δ_1 be the least projection mapping $(\mathit{lift\ true},\ \mathit{lift\ true})$ to $(\perp,\ \mathit{lift\ true})$ so that $\mathit{ID} \circ f \sqsubseteq f \circ \delta_1$, and δ_2 be the least projection mapping $(\mathit{lift\ true},\ \mathit{lift\ true})$ to $(\mathit{lift\ true}, \perp)$ so that $\mathit{ID} \circ f \sqsubseteq f \circ \delta_2$. Then δ_1 and δ_2 are incomparable, and there is a minimal backward abstraction of f that maps $\mathit{ID} \times \mathit{ID}$ to $\delta_1 \times \delta_2$ and $\mathit{ID} \times \mathit{BOT}$ to $\delta_2 \times \mathit{BOT}$.

A continuous function might not even have a minimal backward abstraction. Let f map infinite lists of $\mathbf{1}_\perp$ to $\mathbf{1}_\perp$, such that f returns $\mathit{lift} \perp$ if any element of its argument is $\mathit{lift} \perp$ and $\perp$ otherwise. (The function f is in fact computable: it is the least upper bound of the functions f_i, $1 \leq i \leq n$ where f_i returns the i^{th} element of its argument. Operationally, this can be realised by simulating parallel reduction by diagonalisation: for each $i \geq 1$ starting with 1, reduce the i^{th} cons cell and perform a single reduction step on the head of each of the first through i^{th} cons cells. If at any point a head cell reaches normal form (has value $\mathit{lift} \perp$), stop and return that value.) Now for each $i \geq 0$, let l_i be the list with the first i elements equal to $\perp$ and all subsequent elements equal to $\mathit{lift}\perp$, and let l_∞ be the infinite list of $\perp$. Define γ_i to be the greatest projection that maps l_0 to l_i, and γ_∞ to be the greatest projection that maps every l_i to l_∞. The important property of each γ_i is that it maps no list other than l_∞ to l_∞, so $\mathit{ID} \circ f = f \circ \gamma_i$ for all i. However, $\sqcap_{i \geq 0} \gamma_i = \gamma_\infty$, so $f \sqsupset f \circ \gamma_\infty$. Hence f has no minimal backward abstraction. We note that this is not a consequence of glb on projections differing from glb on continuous functions—here the glb is the same.

The following states that if we restrict ourselves to monotonic backward abstractions, then nothing is lost by the further restriction to continuous backwards abstractions.

Proposition 3.1
If τ is a monotonic backward abstraction of a continuous function f then there is a continuous backward abstraction $\tau' \sqsubseteq \tau$ of f.

Proof
In the following we write $f\ X$ to mean $\{f\ x \mid x \in X\}$ and similarly for functions of more than one argument. Let τ be a monotonic backward abstraction of a continuous function f. Then there is a unique continuous projection transformer τ' such that $\tau'\ \gamma = \tau\ \gamma$ for all finite γ, and

$\tau' \sqsubseteq \tau$. To show that τ' is a backward abstraction of f we need only show that τ' meets the safety condition at infinite arguments. Now every infinite γ is the lub of some directed set X of finite elements, and $\bigsqcup(X \circ f) \sqsubseteq \bigsqcup(f \circ \tau'\ X)$, so $(\bigsqcup X) \circ f \sqsubseteq f \circ \tau'\ (\bigsqcup X)$ since τ', f, and function composition are continuous. $\square$

Corollary 3.2
Every minimal monotonic backward abstraction of a continuous function is continuous.

Henceforth we will only consider continuous backward abstractions.

Even when a strict function has a least (that is, most informative) backward abstraction, it may not be possible to determine that the function is strict from this backward abstraction. Consider all of the monotonic functions from $\mathbf{1}_\perp$ to $\mathbf{1}_\perp$, defined by

$$\begin{array}{llll} \mathit{bot}\ \perp = \perp, & \mathit{id}\ \perp = \perp, & \mathit{top}\ \perp = \mathit{lift}\ \perp, \\ \mathit{bot}\ (\mathit{lift}\ \perp) = \perp, & \mathit{id}\ (\mathit{lift}\ \perp) = \mathit{lift}\ \perp, & \mathit{top}\ (\mathit{lift}\ \perp) = \mathit{lift}\ \perp. \end{array}$$

There are only two projections on $\mathbf{1}_\perp$, namely ID and BOT. The least backward abstractions of *bot* and *top* are the same, the function that maps both ID and BOT to BOT, though *bot* is strict and *top* is not. (The least backward abstraction of *id* is the identity.) Further, so long as the result domain is not $\mathbf{1}$, no single backward abstraction can determine that a function is strict, since any backward abstraction of any function is a backward abstraction of $\lambda x.v$ for any v. This example also shows that the least backward abstraction (or set of all backward abstractions) of a function sometimes determines that function (here *id*) and sometimes does not (here *bot* and *top*).

Interestingly, if a function is head strict, then this fact is determined by some backward abstraction: the function that maps every projection to H is a backward abstraction of a head-strict function.

As shown in [4], by defining the projection STR on every lifted domain $U_\perp$ by

$$\begin{array}{lcl} STR\ \perp & = & \perp, \\ STR\ (\mathit{lift}\ \perp) & = & \perp, \\ STR\ (\mathit{lift}\ v) & = & \mathit{lift}\ v, \text{ if } v \neq \perp, \end{array}$$

we have that f is strict if and only if $STR \circ f_\perp \sqsubseteq f_\perp \circ STR$. Put another way, a continuous function f is strict if and only if there is a continuous backward abstraction τ of $f_\perp$ such that $\tau\ STR \sqsubseteq STR$ (define $\tau\ \alpha = STR$ if $\alpha \sqsubseteq STR$, and $\tau\ \alpha = ID$ otherwise).

Besides ID, BOT, and STR, there is one further projection ABS defined on every lifted domain:

$$\begin{array}{lcl} ABS\ \perp & = & \perp, \\ ABS\ (\mathit{lift}\ v) & = & \mathit{lift}\ \perp. \end{array}$$

For all f we have $ABS \circ f_\perp \sqsubseteq f_\perp \circ ABS$.

So long as U differs from the one-point domain, projections ID, ABS, STR and BOT on $U_\perp$ are all distinct, and form a lattice in which ABS and STR are incomparable. All other projections on any lifted domain lie between ID and ABS, or between STR and BOT. In fact, there is an isomorphism between the lattice of projections between ID and ABS and the projections between STR and BOT. This isomorphism maps each projection between STR and BOT to its least upper bound with ABS; its inverse maps each projection between ID and ABS to its greatest lower bound with STR. Further, every projection in $|\,U_\perp\,|$ between ID and ABS is of the form $\gamma_\perp$, $\gamma \in |\,U\,|$. Hence every projection in $|\,U_\perp\,|$ is either of the form $\gamma_\perp$ or $\gamma_\perp \sqcap STR$. (A revealing observation is that $|\,U_\perp\,|$ is isomorphic to $|\,U\,| \times \mathbf{2}$, where $(\gamma,\ \perp)$ and $(\gamma,\ \mathit{lift}\ \perp)$ in the latter domain correspond to $\gamma_\perp \sqcap STR$ and $\gamma_\perp$ in the former, respectively.) To get the effect of lifting a projection and then taking its glb with STR we introduce the operator $\cdot_{\underline{\perp}}$, pronounced "strict lift", defined by

$$\begin{array}{lcl}
\gamma_{\underline{\perp}}\ \perp & = & \perp\ , \\
\gamma_{\underline{\perp}}\ (\mathit{lift}\ \perp) & = & \perp\ , \\
\gamma_{\underline{\perp}}\ (\mathit{lift}\ v) & = & \mathit{lift}\ (\gamma\ v),\ \text{if}\ v \neq \perp\ .
\end{array}$$

Then $\gamma_{\underline{\perp}} = \gamma_{\perp} \sqcap STR$, and $\gamma_{\perp} = \gamma_{\underline{\perp}} \sqcup ABS$. Further, we have $STR = ID_{\underline{\perp}}$ and $ABS = BOT_{\perp}$; together with the facts $BOT_{\underline{\perp}} = BOT$ and $ID_{\perp} = ID$ we can dispense with the special names STR and ABS.

As was shown for *por*, it does not follow from $\gamma \circ f \sqsubseteq f \circ \delta_1$ and $\gamma \circ f \sqsubseteq f \circ \delta_2$ that $\gamma \circ f \sqsubseteq f \circ (\delta_1 \sqcap \delta_2)$. However, in the case when δ_1 is $ID_{\underline{\perp}}$ this does follow. This is trivially true if $\delta_2 \sqsubseteq ID_{\underline{\perp}}$. Otherwise:

Proposition 3.3
For continuous $f \in U_{\perp} \to V_{\perp}$ and projection γ, if $\gamma \circ f \sqsubseteq f \circ \delta_{\perp}$ and $\gamma \circ f \sqsubseteq f \circ ID_{\underline{\perp}}$, then $\gamma \circ f \sqsubseteq f \circ \delta_{\underline{\perp}}$.

Proof
Suppose that $\gamma \circ f \sqsubseteq f \circ \delta_{\perp}$ and $\gamma \circ f \sqsubseteq f \circ ID_{\underline{\perp}}$, then we need to show that $\gamma\ (f\ v) \sqsubseteq f\ (\delta_{\underline{\perp}}\ v)$ for all v. This clearly holds if $\delta_{\perp}\ v = \delta_{\underline{\perp}}\ v$. Suppose that $\delta_{\perp}\ v \neq \delta_{\underline{\perp}}\ v$. Then it must be that $\mathit{lift}\ \perp\ \sqsubseteq v$, and $\delta_{\perp}\ v = \mathit{lift}\ \perp$ and $\delta_{\underline{\perp}}\ v = \perp$. From $\gamma \circ f \sqsubseteq f \circ \delta_{\perp}$ and $\delta_{\perp}\ v = \mathit{lift}\ \perp$ we have $\gamma\ (f\ v) \sqsubseteq f\ (\mathit{lift}\ \perp)$; by monotonicity and $\mathit{lift}\ \perp\ \sqsubseteq v$ we have $\gamma\ (f\ (\mathit{lift}\ \perp)) \sqsubseteq \gamma\ (f\ v)$ and $f\ (\mathit{lift}\ \perp) \sqsubseteq f\ v$; all together:

$$\gamma\ (f\ (\mathit{lift}\ \perp)) \sqsubseteq \gamma\ (f\ v) \sqsubseteq f\ (\mathit{lift}\ \perp) \sqsubseteq f\ v\ .$$

Idempotence of γ requires that $\gamma\ (f\ (\mathit{lift}\ \perp)) = \gamma\ (f\ v)$. From $\gamma \circ f \sqsubseteq f \circ ID_{\underline{\perp}}$ we have $\gamma\ (f\ (\mathit{lift}\ \perp)) \sqsubseteq f\ \perp$, then by the last equality $\gamma\ (f\ v) \sqsubseteq f\ \perp$; finally since $\delta_{\underline{\perp}}\ v = \perp$ we have $\gamma\ (f\ v) \sqsubseteq f\ (\delta_{\underline{\perp}}\ v)$ as desired. □

Then we can show the following.

Corollary 3.4
If f is a strict continuous function and τ is any continuous backward abstraction of $f_{\perp}$, then there is a continuous backward abstraction τ' of $f_{\perp}$ such that $\tau' \sqsubseteq \tau$ and $\tau'\ ID_{\underline{\perp}} \sqsubseteq ID_{\underline{\perp}}$.

Proof
Since f is strict, $ID_{\underline{\perp}} \circ f_{\perp} \sqsubseteq f_{\perp} \circ ID_{\underline{\perp}}$. Define $\tau'\ \alpha$ to be $\tau\ \alpha$ if $BOT_{\perp} \sqsubseteq \alpha$, otherwise $(\tau\ \alpha) \sqcap ID_{\underline{\perp}}$. Then τ' is continuous, and by Proposition 3.3 and the previous discussion τ' is a backward abstraction of $f_{\perp}$. □

For any continuous function f, the function $f_{\perp}$ is strict and bottom reflecting. This follows from the fact that for all domains U and V, the operator $\cdot_{\perp}$ is an isomorphism between the domain of continuous functions $U \to V$ and the domain $U_{\perp} \stackrel{\text{sb}}{\to} V_{\perp}$ of continuous, strict, bottom-reflecting functions. Though the function $f_{\perp}$ contains no more information than f, projections on the argument and result domains of $f_{\perp}$, and hence a backward abstraction for $f_{\perp}$, may contain more information than those for f since the projections on the lifted domains have the additional degree of freedom to map values to the new bottom element. Intuitively, a value that is mapped to the new bottom element may be thought of as 'not sufficiently defined', or 'unacceptable'. Projections on lifted domains may then be regarded as giving lower bounds on the definedness of values in the corresponding unlifted domain, and thus lower bounds on the degree of evaluation of expressions that take values in the unlifted domains. For example, $STR \in |\,U_{\perp}\,|$ maps $\mathit{lift}\ \perp$ (which corresponds to $\perp$ in U) to $\perp$, indicating that $\perp$ in U is not an acceptable value. If expression `f` denotes function f, then $STR \circ f_\perp \sqsubseteq f_\perp \circ STR$ may be interpreted as "if the result of f must be more defined than $\perp$, then the argument of f must be more defined than $\perp$", that is, f is strict. Operationally, in any context in which it is safe (and in particular, necessary) to evaluate an application of `f`, then it is safe to evaluate its argument.

The backward abstractions of a function $f_\perp$ can reveal more than just simple strictness in f, as the following shows.

Proposition 3.5
Every continuous strict bottom-reflecting function is determined by its continuous backward abstractions.

In order to prove this theorem, we need some results regarding what we will call projection-based *forward analysis*, wherein the goal is, given function f and projection δ, to find projection γ such that $\gamma \circ f \sqsubseteq f \circ \delta$. Projection-based forward analysis is interesting in its own right, for example, Launchbury uses it in his study and implementation of partial evaluation [7]. Some further details on the power of forward analysis may be found in [6].

A projection transformer τ is a *forward abstraction* of f if for all δ we have $(\tau\ \delta) \circ f \sqsubseteq f \circ \delta$. Then

Proposition 3.6
Every continuous function has a greatest forward abstraction, and it is continuous.

Proof
Let f and δ be fixed. Let X be the set of projections γ such that $\gamma \circ f \sqsubseteq f \circ \delta$. Since X is not empty (it always contains BOT), $\bigsqcup X$ always exists. So

$$\begin{array}{rcll} (\bigsqcup X) \circ f & = & \bigsqcup (X \circ f) & [\text{continuity } \circ] \\ & \sqsubseteq & f \circ \delta & [\text{safety condition}] \end{array}$$

Since lub in the lattice of projections and lub for projections in the domain of continuous functions is the same, $\bigsqcup X$ is the greatest projection satisfying the safety condition.

Now we show that the greatest forward abstraction τ of f is continuous. It is clear that τ is monotonic. Let X be a directed set of projections on the argument domain of f. The image of a directed set under a monotonic function is directed, so $\tau\ X$ is directed, and $\bigsqcup(\tau\ X) \sqsubseteq \tau\ (\bigsqcup X)$. On the other hand, by the definition of τ we have $\tau\ (\bigsqcup X) \sqsubseteq f \circ (\bigsqcup X)$, so by the continuity of composition $\tau\ (\bigsqcup X) \sqsubseteq \bigsqcup(f \circ X)$, hence by algebraicity there is some $\delta \in X$ such that $\tau\ (\bigsqcup X) \sqsubseteq \bigsqcup(f \circ \delta)$, so $\tau\ (\bigsqcup X) \sqsubseteq \tau\ \delta$, hence $\tau\ (\bigsqcup X) \sqsubseteq \bigsqcup(\tau\ X)$. We conclude that $\bigsqcup(\tau\ X) = \tau\ (\bigsqcup X)$, and hence that τ is continuous. $\square$

Proposition 3.7
Every continuous strict bottom-reflecting function is determined by its greatest forward abstraction.

Proof
Let $f \in U \to V$ be continuous, strict, and bottom reflecting, and let τ be the greatest forward abstraction of f. Then $\tau \in |\,U\,| \to |\,V\,|$. Given domain T, For all $c \in T$ let $\gamma_c \in |\,T\,|$ be defined by

$$\begin{array}{l} \gamma_c\ x = \perp, \text{ if } x \sqsubseteq c\ , \\ \gamma_c\ x = x, \text{ if } x \not\sqsubseteq c\ . \end{array}$$

It is not hard to see that for $x \in U$, it must be that $\tau\ \gamma_x = \gamma_{(f\ x)}$. Since $\gamma_c = \gamma_d$ iff $c = d$, it is straightforward to reconstruct f from τ. $\square$

Proposition 3.8
The greatest forward abstraction of a continuous function is determined by its continuous backward abstractions.

Proof
First we observe that $\gamma \circ f \sqsubseteq f \circ \delta$ if and only if there exists a continuous backward abstraction τ such that $\delta = \tau\ \gamma$ (define $\tau\ \beta$ to be δ if $\beta \sqsubseteq \gamma$ and ID otherwise). Second, if X is the set of projections γ such that $\gamma \circ f \sqsubseteq f \circ \delta$, then as shown in the proof of Proposition 3.6, the greatest forward abstraction of f maps δ to $\bigsqcup X$. $\square$

Proof of Proposition 3.5
That a continuous strict bottom-reflecting function is determined by its continuous backward abstractions follows from the fact that function is determined by its greatest forward abstraction, which is in turn determined by its continuous backward abstractions. $\square$

Often we work with functions with argument domains that are finite (e.g. *Bool*) or of finite depth (e.g. $Int = \mathbf{Z}_\perp$).

Corollary 3.9
Every monotonic, strict, bottom-reflecting function $f \in U \to V$ such that U is of finite depth is determined by its minimal backward abstractions.

For example, $por_\perp$ is determined by its two minimal backward abstractions.

Following we show a dual to the fact that the greatest forward abstraction of a function is determined by its backward abstractions.

Proposition 3.10
The set of backward abstractions of a continuous function is determined by its greatest forward abstraction.

Proof
Let τ' be the greatest forward abstraction of a function f. Then the projection transformer τ is a backward abstraction of f if and only if $\tau' \circ \tau \sqsupseteq id$. $\square$

Hence the greatest forward abstraction of a function contains the same information as its set of backward abstractions.

We now briefly relate this to reversal of abstract interpretations [9]. If τ' is any forward abstraction of a continuous function f then any τ such that $\tau' \circ \tau \sqsupseteq id$ is a backward abstraction of f. Similary, if τ is any backward abstraction of f then any τ' such that $\tau \circ \tau' \sqsubseteq id$ is a forward abstraction of f. In particular, when τ' is the greatest forward abstraction of f, the set of projection transformers τ such that $\tau' \circ \tau \sqsupseteq id$ is a *relational reversal* of τ', and contains the same information as τ'. In general this set cannot be characterised by a single element, that is, a value τ such that $\tau' \circ \tau \sqsupseteq id$ and $\tau \circ \tau' \sqsubseteq id$—such a τ would be the least backward abstraction of f. When a least backward abstraction τ exists, τ' and τ form a *Galois connection.* Propositions 3.7 and 3.10 can be shown to follow in part from the more general theory of reversal of abstract interpretations—the only significant difference is that we work with infinite rather than finite lattices and so consider continuity in addition to monotonicity.

In general, when we wish to determine strictness properties of some function f we will find backward abstractions of $f_\perp$ rather than of f. Following is a first step in this direction.

Fact 3.11
For all f we have $\gamma \circ f \sqsubseteq f \circ \delta$ iff $\gamma_\perp \circ f_\perp \sqsubseteq f_\perp \circ \delta_\perp$.

So, given a backward abstraction of a function f we may readily derive a backward abstraction τ of $f_\perp$—we may safely take $\tau\gamma_{\underline{\perp}} = \delta$ when $\tau\ \gamma_\perp = \delta$. Though this takes no advantage of the lifting, it does say that no information available via a backward abstraction is lost by lifting, since $\cdot_\perp$ on functions is an injection. That $BOT \circ f \sqsubseteq f \circ BOT$ allows the following improvement. For any monotonic backward abstraction τ of f, and τ' defined by $\tau'\ BOT = BOT$ and $\tau'\ \gamma = \tau\ \gamma$ for $\gamma \neq BOT$, we have τ' is a monotonic backward abstraction of f and $\tau' \sqsubseteq \tau$.

Proposition 3.12
If τ is a continuous backward abstraction of a continuous function f, then there is a strict continuous backward abstraction τ' of f such that $\tau' \sqsubseteq \tau$. Hence every minimal monotonic backward abstraction of a continuous function is strict and continuous.

Since every monotonic backward abstraction of a continuous function is approximated by a strict continuous backward abstraction, we restrict our attention to the strict continuous projection transformers, which form a sublattice of the lattice of projection transformers.

The following proposition and its proof come from [4].

Proposition 3.13
If $\gamma_{\underline{\perp}} \circ f_\perp \sqsubseteq f_\perp \circ \delta$ then $\gamma_\perp \circ f_\perp \sqsubseteq f_\perp \circ (\delta \sqcup BOT_\perp)$.

Proof

$$
\begin{array}{lll}
 & \gamma_\perp \circ f_\perp & \\
= & (\gamma_{\underline{\perp}} \sqcup BOT_\perp) \circ f_\perp & [\text{defn } \cdot_{\underline{\perp}}] \\
= & (\gamma_{\underline{\perp}} \circ f_\perp) \sqcup (BOT_\perp \circ f_\perp) & [\text{defn } \sqcup] \\
\sqsubseteq & (f_\perp \circ \delta) \sqcup (f_\perp \circ BOT_\perp) & [BOT_\perp \circ f_\perp \sqsubseteq f_\perp \circ BOT_\perp] \\
= & f_\perp \circ (\delta \sqcup BOT_\perp) & [\text{defn } \sqcup]
\end{array}
$$

□

Thus, in defining a backward abstraction τ of a function $f_\perp$, if $\gamma_{\underline{\perp}} \circ f_\perp \sqsubseteq f_\perp \circ \delta$ then it is safe to take $\tau\ \gamma_\perp = \delta \sqcup BOT_\perp$. Not only is this safe, but as we show, it is the right choice. Let a projection transformer τ have the *guard property* if when $\tau\ \gamma_{\underline{\perp}} = \delta$, then $\tau\ \gamma_\perp = \delta \sqcup BOT_\perp$. (In [4], an operator "guard" is defined to facilitate definition of strict projection transformers with the guard property. Here we have separated the issues.)

Fact 3.14
If τ is a backward abstraction of a continuous function $f_\perp$, then for all γ the projection $\tau\ \gamma_\perp$ has the form $\delta_\perp$ for some δ.

Proposition 3.15
If τ is a continuous backward abstraction of a continuous function $f_\perp$, then there is a strict continuous backward abstraction τ' of f with the guard property such that $\tau' \sqsubseteq \tau$.

Proof
Let τ be a continuous backward abstraction of a continuous function $f_\perp$, and define τ' by

$$
\begin{array}{lll}
\tau'\ BOT_{\underline{\perp}} & = & BOT_{\underline{\perp}} \\
\tau'\ \gamma_{\underline{\perp}} & = & \tau\ \gamma_{\underline{\perp}},\ \text{if } \gamma \neq BOT \\
\tau'\ \gamma_\perp & = & \tau\ \gamma_{\underline{\perp}} \sqcup BOT_\perp\ .
\end{array}
$$

Then τ' is a strict continuous backward abstraction of $f_\perp$ with the guard property. Now for all γ we have

$$
\begin{array}{lll}
 & \gamma_{\underline{\perp}} \sqsubseteq \gamma_\perp & [\text{defns } \cdot_\perp,\ \cdot_{\underline{\perp}}] \\
\Rightarrow & \tau\ \gamma_{\underline{\perp}} \sqsubseteq \tau\ \gamma_\perp & [\text{monotonicity } \tau] \\
\Rightarrow & (\tau\ \gamma_{\underline{\perp}}) \sqcup BOT_\perp \sqsubseteq (\tau\ \gamma_\perp) \sqcup BOT_\perp & [\text{monotonicity } \sqcup] \\
\Rightarrow & (\tau\ \gamma_{\underline{\perp}}) \sqcup BOT_\perp \sqsubseteq \tau\ \gamma_\perp & [\text{Fact 3.14}] \\
\Rightarrow & \tau'\ \gamma_\perp \sqsubseteq \tau\ \gamma_\perp & [\text{defn } \tau']
\end{array}
$$

So $\tau' \sqsubseteq \tau$ as required. □

Corollary 3.16
Every minimal monotonic backward abstraction of a continuous function $f_\perp$ is strict, continuous, and has the guard property.

We conclude that for continuous functions, it is sensible to restrict attention to strict continuous backward abstractions, and for continuous, strict, bottom-reflecting functions from lifted domains to lifted domains, to restrict attention to strict, continuous backward abstractions with the guard property. Just as the space $U_\perp \xrightarrow{\mathrm{sb}} V_\perp$ of strict and bottom-reflecting continuous functions is a subdomain of the domain of continuous functions $U_\perp \to V_\perp$, the space $|\,V_\perp\,| \xrightarrow{\mathrm{sg}} |\,U_\perp\,|$ of strict continuous projection transformers with the guard property is a sublattice of the continuous projection transformers $|\,V_\perp\,| \to |\,U_\perp\,|$: it is isomorphic to $|\,V\,| \xrightarrow{\mathrm{s}} |\,U_\perp\,|$, where $\xrightarrow{\mathrm{s}}$ contructs the domain of strict continuous functions.

4 Stability and Backward Analysis

As shown, an arbitrary continuous function may not have a least backward abstraction, or even a minimal backward abstraction. However, least backward abstractions always exist for *stable* functions. The theory of stability was developed by Berry [8] in an attempt to extend the characterisation of *sequential* functions to include higher order functions. At first order the stable functions are a superset of the sequential functions, and this is hypothesised to be the case at higher order.

Definition
A continuous function f is *stable* if for all x and y such that $y \sqsubseteq f\ x$, there exists a least value $M(f, x, y) \sqsubseteq x$ such that $y \sqsubseteq f\ (M(f, x, y))$.

The simplest function that is continuous but not stable is $f \in \mathbf{1}_\perp \times \mathbf{1}_\perp$ where f maps $(\perp, \perp)$ to $\perp$ and all other arguments to *lift* $\perp$—here *lift* $\perp \sqsubseteq f$ (*lift* $\perp$, *lift* $\perp$) but there is no least x such that *lift* $\perp \sqsubseteq f\ x$. However, parallel-or is regarded as the archetypical non-stable function, and it plays an important role in the development of the theory of stability. In fact, at first order the stable functions are a superset of the sequential functions: an example (due to Berry) of a function that is stable but not sequential is the least monotonic function h such that h (*lift true*, *lift false*, $\perp$) $= h$ ($\perp$, *lift true*, *lift false*) $= h$ (*lift false*, $\perp$, *lift true*) $=$ *lift true*. Note that h is not the three-argument analog of parallel-or (which is not stable), since h (*lift false*, *lift true*, $\perp$) $= \perp$.

A well-known and useful consequence of the definition of stability is the following.

Proposition 4.1
Given stable f, for all x_1, x_2 such that there exists y such that $x_1,\ x_2 \sqsubseteq y$, we have $f\ (x_1 \sqcap x_2) = (f\ x_1) \sqcap (f\ x_2)$.

Proof
We have $f\ x_1 \sqcap f\ x_2 \sqsubseteq f\ y$ (monotonicity of f), so there is a least $x' \sqsubseteq y$ such that $f\ x_1 \sqcap f\ x_2 \sqsubseteq f\ x'$. Since $x_1,\ x_2 \sqsubseteq y$ it must be that $x' \sqsubseteq x_1,\ x_2$ and hence $x' \sqsubseteq x_1 \sqcap x_2$, so $f\ (x_1 \sqcap x_2) \sqsupseteq (f\ x_1) \sqcap (f\ x_2)$. However, $f\ (x_1 \sqcap x_2) \sqsubseteq f\ x_1,\ f\ x_2$, so $f\ (x_1 \sqcap x_2) \sqsubseteq (f\ x_1) \sqcap (f\ x_2)$. We conclude that $f\ (x_1 \sqcap x_2) = (f\ x_1) \sqcap (f\ x_2)$. $\square$

Proposition 4.2
Every stable function has a least backward abstraction.

Proof
Let $f \in U \to V$ be a stable function and $\gamma \in |\,V\,|$ be fixed. Since f is stable there is a least

function δ such that $\gamma \circ f \sqsubseteq f \circ \delta$. Clearly δ is weaker than the identity; to show that it is a projection we show that it is monotonic, continuous, and idempotent, in that order.

Let $x,\ y \in U$, with $x \sqsubseteq y$. Now

$$\begin{array}{rcll} \gamma\ (f\ x) & \sqsubseteq & \gamma\ (f\ y) & [\text{monotonicity } f \text{ and } \gamma] \\ & \sqsubseteq & f\ (\delta\ y) & [\text{definition } \delta] \end{array}$$

Since $\delta\ a$ is the least value below a such that $\gamma\ (f\ a) \sqsubseteq f\ (\delta\ a)$, it cannot be that $\delta\ a$ is larger than $\delta\ b$. If $\delta\ a$ were incomparable to $\delta\ b$, then $\delta\ a \sqcap \delta\ b$ would be strictly smaller than $\delta\ a$ or $\delta\ b$, and

$$\begin{array}{cll} & f\ (\delta\ a \sqcap \delta\ b) & \\ = & f\ (\delta\ a) \sqcap f\ (\delta\ b) & [\delta\ a \sqsubseteq\ a,\ \delta\ b \sqsubseteq\ b,\ f \text{ stable}] \\ \sqsupseteq & \gamma\ (f\ a) \sqcap \gamma\ (f\ b) & [\text{definition } \delta, \text{ property } \sqcap] \\ = & \gamma\ (f\ a) & [\text{monotonicity of } f] \end{array}$$

contrary to $\delta\ a$ being least. Hence $\delta\ a \sqsubseteq \delta\ b$ and we conclude that δ is monotonic.

Let $X \subseteq V$ be a directed set. Since δ is monotonic, $\delta\ X$ is directed, and $\delta\ (\bigsqcup X)\ \sqsupseteq\ \sqcup\ (\delta\ X)$ again since δ is monotonic. Now

$$\begin{array}{cll} & \gamma\ (f\ (\bigsqcup X)) & \\ = & \sqcup\ (\gamma\ (f\ X)) & [\text{continuity } f,\ \gamma] \\ \sqsubseteq & \sqcup\ (f\ (\delta\ X)) & [f,\ \delta \text{ monotonic}, \gamma \circ f \sqsubseteq f \circ \delta] \\ = & f\ (\sqcup\ (\delta\ X)) & [f \text{ continuous}] \\ \sqsubseteq & f\ (\delta\ (\bigsqcup X)) & [\delta \text{ monotonic}]\ . \end{array}$$

Since $\delta\ (\bigsqcup X)$ is by definition the least value weaker than $\bigsqcup X$ such that $\gamma\ (f\ (\bigsqcup X)) \sqsubseteq f\ (\delta\ (\bigsqcup X))$, and $\sqcup\ (\delta\ X) \sqsubseteq \bigsqcup X$ (since δ is weaker then the identity), it must be that $\sqcup\ (\delta\ X) = \bigsqcup X$. We conclude that δ is continuous.

Let $x \in V$, then $\gamma\ (f\ x) \sqsubseteq f\ (\delta\ x)$. Now $\delta\ (\delta\ x)$ is by definition the least value bounded above by $\delta\ x$ such that $\gamma\ (f\ (\delta\ x)) \sqsubseteq f\ (\delta\ (\delta\ x))$. Since $\gamma \circ f \circ \delta = \gamma \circ f$, it must be that $\delta\ (\delta\ x)$ is the least value bounded above by $\delta\ x$ such that $\gamma\ (f\ x) \sqsubseteq f\ (\delta\ (\delta\ x))$. However, $\delta\ x$ is the least value bounded above by x such that $\gamma\ (f\ x) \sqsubseteq f\ (\delta\ x)$; hence it must be that $\delta\ x = \delta\ (\delta\ x)$, and we conclude that δ is idempotent.

Thus the projection δ is the least function such that $\gamma \circ f \sqsubseteq f \circ \delta$. It follows that there is a least function τ such that $\gamma \circ f \sqsubseteq f \circ (\tau\ \gamma)$ for all G. It is clear that τ is monotonic; Proposition 3.1 tells us that τ is continuous. $\Box$

We write $|\,f\,|$ to denote the least backward abstraction of f.

Proposition 4.3
Every stable function f is determined by the least backward abstraction of $f_\perp$.

Proof
Proposition 3.8 tells us that the greatest forward abstraction of a stable function is determined by its least backward abstraction. If f is stable then so is $f_\perp$, so by Proposition 3.7 f is determined by the least backward abstraction of $f_\perp$. $\Box$

The safety condition for backward abstractions of continuous functions is not monotonic in the following sense. Recall the definitions *bot*, *id*, and *top*. The least backward abstractions of *bot* and *top* are $\lambda\alpha.BOT$, and the least backward abstraction of *id* is $\lambda\alpha.\alpha$, so there are backward abstractions of both *bot* and *top* that are not safe for *id*. More generally, if τ is a backward abstraction of f and τ' is a backward abstraction of f', and $f \sqsubseteq f'$, then it is not generally the case that either τ' is a backward abstraction of f or that τ is a backward abstraction of f'. Since *bot*, *id*, and *top* are stable, this example shows that the mapping to least backward abstractions is not monotonic under the usual function ordering. However, the mapping is monotonic when the ordering on the argument domain is the *stable* ordering.

Definition
For stable f and g the stable ordering $\sqsubseteq_s$ is defined by $f \sqsubseteq_s g$ iff $f \sqsubseteq g$ and for all x, y, if $y \sqsubseteq f\ x$ then $M(f, x, y) = M(g, x, y)$.

Proposition 4.4
For all stable functions f and g, we have that, $f \sqsubseteq_s g$ implies $|\,f\,| \sqsubseteq |\,g\,|$.

Proof
Let $f,\ g \in U \to V$ be stable functions with $f \sqsubseteq_s g$, and let $\gamma \in |\,V\,|$. Then by the definition of $\sqsubseteq_s$ we have $f \sqsubseteq g$ and for all x we have $M(f, x, \gamma\ (f\ x)) = M(g, x, \gamma\ (f\ x))$. Now since $f \sqsubseteq g$ we have $\gamma\ (f\ x) \sqsubseteq \gamma\ (g\ x)$, so that $M(f, x, \gamma\ (f\ x)) \sqsubseteq M(g, x, \gamma\ (g\ x))$, since M is monotonic in its third argument. Since $M(f, x, \gamma\ (f\ x)) = |\,f\,|\ \gamma\ x$, and similarly for g, we have that $f \sqsubseteq_s g$ implies $|\,f\,| \sqsubseteq |\,g\,|$ $\square$

For example, for *bot*, *id*, and *top* as defined before we have $bot \sqsubset_s id$, $bot \sqsubset_s top$, but $id \not\sqsubseteq_s top$ and $top \not\sqsubseteq_s id$.

5 Two Implications for Real Analyses

We identify two sources of imprecision in the first-order projection-based backward strictness analysis of Wadler and Hughes [4] and show how they can be eliminated by making the analysis relational. In essence, given a function definition denoting a function f in the standard semantics, their analysis determines a projection transformer that is (isomorphic to) a backward abstraction of $f_\perp$.

The first source of imprecision is the the treatment of non-stable functions such as *por*. As we have seen two backward abstractions are required to determine $por_\perp$, and in general a continuous functions from domains which are finite (as are all of the abstract domains in implemented analysers) have more than one minimal backward abstraction. Equivalently, a continuous function f is determined by a function τ mapping projections γ to sets of projections $\{\delta_i\}$ such that $\gamma \circ f_\perp \sqsubseteq f_\perp \circ \delta_i$ for all i, and for functions from finite domains these sets are finite and are characterised by their minimal elements. Taking the upward closures of these sets gives functions from domains of projections to powerdomains of projections.

A second source of imprecision is the treatment of products. Ignoring for the moment the issue of lifted domains, given a projection γ on pairs from $U_1 \times U_2$ we seek to find δ_1 and δ_2 satisfying the safety condition $\gamma\ (x_1, x_2) \sqsubseteq (\delta_1\ x_1, \delta_2\ x_2)$. Their solution is to define

$$\begin{aligned} \delta_1\ x &= \sqcup_y\ fst\ (\gamma\ (x, y))\ , \\ \delta_2\ y &= \sqcup_x\ snd\ (\gamma\ (x, y))\ . \end{aligned}$$

In fact $\delta_1 \times \delta_2$ is the least element of $|\,U_1\,| \times |\,U_2\,| \subseteq |\,U_1 \times U_2\,|$ such that $\gamma \sqsubseteq \delta_1 \times \delta_2$. Factoring the projection in this way makes defining the non-standard semantics straightforward since it allows projections to be propogated 'backward' down expressions, but it loses information. However, every element of $|\,U_1 \times U_2\,|$ is characterised by a set of elements from $|\,U_1\,| \times |\,U_2\,|$ as follows.

Proposition 5.1
Let $\gamma \in |\,U_1 \times U_2\,|$. Let $\gamma' = \bigsqcup\{\delta_1 \times \delta_2 \in |\,U_1 \times U_2\,|,\ \delta_1 \times \delta_2 \sqsubseteq \gamma\}$ Then $\gamma = \gamma'$.

Proof
Certainly $\gamma' \sqsubseteq \gamma$. For any x and y let $(x', y') = \gamma\ (x, y)$. For any z and z' such

that $z' \sqsubseteq z$ define $p(z, z')$ to be the least projection such that $p(z, z')\ z = z'$. Then $p(x, x') \times p(y, y')\ (x, y) = (x', y')$. Define

$$q(x, y) \quad = \quad p(x,\ \mathit{fst}\ (\gamma\ (x, y))) \times p(y,\ \mathit{snd}\ (\gamma\ (x, y)))\ .$$

Then $\gamma = \sqcup_{x,y} q(x, y)$, and since for all x and y we have $q(x, y) \sqsubseteq \gamma$, it must be that $\gamma' = \gamma$. A similar construction may be done using glb instead of lub. $\square$

Let us call projections on products that can be expressed as products of projections *independent* projections. Then any projection transformer can be characterised by a function from sets of independent projections to sets of independent projections. Putting these two observations together, we have that any continuous, strict, bottom-reflecting function is determined by a function from a power domain of independent projections to a power-power domain (sets of sets) of projections. In fact we can do better than this; the construction of least backward abstractions of stable functions in [6] shows that every stable, strict, bottom-reflecting functions is determined by a function from a domain of independent projections to a powerdomain of independent projections, and every continuous, strict, bottom-reflecting functions is determined by a function from a domain of independent projections to a power-power domain of independent projections. Though this might greatly increase the sizes of the domains involved, it is conceivable that these ideas could be practically exploited in a restricted form, e.g. in a *locally relational* analysis [9].

References

[1] Wray, S. "A new strictness detection algorithm." In *Proceedings of the Workshop in Implementation of Functional Languages* (Aspenäs, Sweden). L. Augustsson et. al., eds. Report 17, Programming Methodology Group, Department of Computer Sciences, Chalmers University of Technology and University of Göteborg, Göteborg, Sweden.

[2] Augustsson, L. and Johnson, T. "The Chalmers Lazy ML Compiler." Department of Computer Science, Chalmers University of Technology, Göteborg, Sweden, 1988.

[3] Burn, G.L., "Using Projection Analysis in Compiling Lazy Functional Programs", *Proceedings of the 1990 ACM Conference on Lisp and Function al Programming*, ACM, 1990.

[4] Wadler, P., and Hughes, J. "Projections for Strictness Analysis." In *Proceedings of Functional Programming Languages and Computer Architecture* (Portland, Oregon). G. Kahn, ed. LNCS 274. Springer-Verlag, Berlin, 1987.

[5] Hunt, S. *Projection analysis and stable functions.* Unfinished manuscript.

[6] Davis, K. "A Note on the Choice of Domains for Projection-Based Program Analysis." In *Functional Programming: Proceedings of the 1991 Glasgow Workshop, 13-15 August 1991, Isle of Skye, Scotland.* P. Wadler *et al.*, eds. Springer Workshops in Computing. Springer-Verlag, 1992.

[7] Launchbury, J. *Projection Factorisations in Partial Evaluation.* PhD Thesis, Glasgow University, Nov 89. Distinguished Dissertation in Computer Science, Vol 1, CUP, 1991.

[8] Berry, G. "Stable models of typed lambda-calculi." In *Proceedings of the 5th ICALP* pp 375-387, LNCS 62. Springer-Verlag, Berlin, 1978.

[9] Hughes, R.J.M. and Launchbury, J. "Relational Reversal of Abstract Interpretation." In ??.