Abstract Interpretation, Logical Relations, and Kan Extensions

SAMSON ABRAMSKY, *Imperial College, London*

Abstract

We develop a formalism for abstract interpretation based on logical relations. As a case study, we use this formalism to give new proofs of correctness for strictness analysis on the typed λ-calculus, and also for termination analysis. We then go on to a deeper study of the duality between safety and liveness properties, and the construction of abstraction functions which can be used to give the best possible interpretations of higher-type constants. This turns out to be a special case of the construction of Kan extensions in category theory. Necessary and sufficient conditions are given for abstraction functions to be definable over continuous type structures and for the abstraction functions themselves to be continuous.

Keywords: Abstract interpretation; strictness analysis; λ-calculus; logical relations; safety; liveness; powerdomains; Kan extensions.

1. Introduction

A technique of growing importance in static program analysis (as applied, for example, in advanced compiler technology) is the use of non-standard interpretations (or semantics) of programming languages to yield partial information on selected aspects of program behaviour. The task of a theoretical framework for this technique is to provide appropriate concepts to formalize the notion of *correctness* of an analysis, and mathematical tools and theories to support *proofs* of correctness of particular analyses. In addition, such theories may provide guide-lines for *obtaining* suitable analyses in the first place, and criteria for establishing that a given analysis is the *best possible* under some initial assumptions. Following the pioneering work of the Cousot's in establishing such a framework for flowchart programs [2], Mycroft applied similar ideas to functional programming languages [13], and introduced a number of new analyses, one of which, *strictness analysis* [12], has become of increasing importance in the implementation of functional programming languages and the design of new architectures to support such languages [3]. Subsequently, Nielson has developed a very general and sophisticated framework for abstract interpretation based on denotational language definitions [17].

The question of extending Mycroft's original form of strictness analysis to

© Oxford University Press

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

richer programming languages has concerned a number of researchers [25], [7], [8]. In [1] a theoretical framework for strictness analysis in the typed λ-calculus is developed.

All the work in [13], [14], [17], [1] is based on the use of power domains and continuous functions to relate 'standard' and 'non-standard' interpretations of a given programming language. By contrast, [15], see also [16], proposes a relational formalism, in a very general setting. In this paper, we take up Mycroft's suggestion, but with a number of differences. Our notion of interpretation is less general than his, we consider typed rather than untyped λ-calculus, and we use the standard notion of *logical relation* [19], which is an important tool in the study of the λ-calculus [20], [11]. Our motivation for these choices is that we wish to expose latent mathematical structure as clearly as possible, rather than to strive for maximum generality. Nevertheless, we are able to offer a striking and non-trivial case study, namely strictness analysis itself.

We develop a new approach to proving the correctness of strictness analysis using logical relations. We are able to give simple, structured proofs of correctness both for strictness analysis for higher-order functions in the typed λ-calculus—i.e. the main result in [1]—and also for the dual *termination analysis* of [12].

The only previous correctness proofs for the latter analysis appeared in [14] and [17] and were technically very heavy, using 'generalized domain theory' (limits of transfinite chains, etc.) and non-standard powerdomain constructions. While the development in [1] stays within standard domain theory it also involves quite complex calculations with powerdomains. By contrast, our definitions are simple and regular, and our correctness proofs are trivialities once we have established a few general results on logical relations.

We then go on to study *abstraction functions* which pick out 'best' representatives from the abstract values related to a given concrete value. Such functions can be used, for example, to find best possible interpretations of higher-type constants. We develop a rather general theory, based on the duality between *safety* and *liveness* properties (strictness and termination being an instance of this duality). Some illuminating connections are found with logical relations, on the one hand, and *Kan extensions* [10], a fundamental notion from category theory on the other. We also develop necessary and sufficient conditions for continuity of these abstraction functions; these conditions being the role of *powerdomains* in abstract interpretation into sharper focus. Finally, we describe a new analysis, *totality* analysis and prove its soundness.

We shall assume some familiarity with domain theory throughout this paper: see, for example, [6]. We shall use the following notation for step functions:

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

Notation. We write $K(D)$ for the set of *compact* or *finite* elements of an algebraic cpo D. Given algebraic domains D, E and finite elements $b \in D, e \in E$, we write

$$[b, e]: D \to_c E$$

for the *step function* defined by

$$[b, e]x = \begin{cases} e, & x \sqsupseteq b \\ \bot & \text{otherwise} \end{cases}.$$

If D, E are Scott domains, the finite elements of $[D \to_c E]$ are the lubs of consistent (i.e. bounded) finite sets of step functions.

Note. The present paper was completed in manuscript in October 1985, and circulated to a few researchers. An incomplete version (Sections 1–5 only) had previously (September 1985) been circulated under the title 'Strictness Analysis via Logical Relations'.

2. Syntax and semantics of typed λ-calculus

Syntax of types

Base types A, B, . . .

Type expressions $\sigma, \tau, \ldots$

$\tau ::= \mathrm{A} \mid \tau_1 \to \tau_2$.

A *language* L is specified by giving a set of base types and a set of typed *constants* c_τ.

Syntax of typed terms

For each type σ, we have an infinite set of typed variables

$$Var_\sigma = \{x^\sigma, y^\sigma, z^\sigma, \ldots\}.$$

Now $\Lambda_T(\mathrm{L})$, the typed λ-calculus over a language L, comprises typed terms $e : \sigma$ formed according to the following rules:

1. $x^\sigma : \sigma$
2. $c_\sigma : \sigma$
3. $\dfrac{e : \tau}{\lambda x^\sigma \cdot e : \sigma \to \tau}$
4. $\dfrac{e_1 : \sigma \to \tau \quad e_2 : \sigma}{e_1 e_2 : \tau}$

Semantics of typed λ-calculus: interpretation

An *interpretation* I of a language L is specified by

$$\mathrm{I} = (\{D'_A\}, \{c'_\sigma\})$$

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

where

- D^I_A is a cpo (complete partial order) for each base type A. This is extended to D^I_σ for all types σ by

$$D^I_{\sigma\to\tau} = [D^I_\sigma \to_c D^I_\tau]$$

  where $[D \to_c E]$ is the cpo of all continuous functions from D to E, ordered pointwise.
- $c^I_\sigma \in D^I_\sigma$.

An interpretation I induces a semantic function

$$E^I : \Lambda_T(L) \to Env^I \to \cup D^I_\sigma$$

where

$$Env^I = \{Env^I_\sigma\},\ Env^I_\sigma = Var_\sigma \to D^I_\sigma$$

$$E^I[\![x^\sigma]\!]\rho = \rho x^\sigma$$

$$E^I[\![c_\tau]\!]\rho = c^I_\tau$$

$$E^I[\![\lambda x^\sigma \cdot e]\!]\rho = \lambda \xi \in D^I_\sigma \cdot E^I[\![e]\!]\rho[x^\sigma \mapsto \xi]$$

$$E^I[\![e_1 e_2]\!]\rho = (E^I[\![e_1]\!]\rho)(E^I[\![e_2]\!]\rho).$$

(Here $\rho[x^\sigma \mapsto d]$, for $d \in D^I_\sigma$ is the environment in Env^I such that

$$\rho[x^\sigma \mapsto d]x^\sigma = d,$$

$$\rho[x^\sigma \mapsto d]y^\tau = \rho y^\tau (x^\sigma \neq y^\tau).)$$

Note that $E^I[\![e]\!]\rho$ is in D^I_σ if e has type σ.

3. Logical relations

Given a language L, there will be a *standard* interpretation inducing a *standard* semantics—i.e. the intended denotational semantics of the language. Other interpretations will induce non-standard semantics, which may be used in static analysis of programs. This suggests that it may be useful to relate interpretations.

3.1. Definition

A *relation* $R : I \to J$ is a family $\{R_\sigma\}$, with $R_\sigma \subseteq D^I_\sigma \times D^J_\sigma$.

A relation is *logical* [PLO73] if for all σ, τ:

$(*) fR_{\sigma\to\tau}g \Leftrightarrow [\forall a \in D^I_\sigma,\ b \in D^J_\sigma \cdot aR_\sigma b \Rightarrow faR_\tau gb]$.

Clearly, any family R_A of relations on the base types can be extended uniquely to a logical relation by using (*) as a *definition*.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

3.2. Proposition

Let $R : I \to J$ be a logical relation. Suppose that $c^I_\sigma R_\sigma c^J_\sigma$ for all constants c_σ. Then: $\forall e : \sigma \forall \rho \in Env^I$, $\rho' \in Env^J$

$$\rho R \rho' \Rightarrow (E^I[\![e]\!]\rho) R_\sigma (E^J[\![e]\!]\rho').$$

(Here $\rho R \rho'$ means $\forall x^\tau \cdot \rho x^\tau R_\tau \rho' x^\tau$.)

Proof

By induction on e. The cases for variables and constants are immediate from the hypothesis. For $e = e_1 e_2$, $e_1 : \sigma \to \tau$, $e_2 : \sigma$ by induction hypothesis $(E^I[\![e_1]\!]\rho) R_{\sigma\to\tau} (E^J[\![e]\!]\rho')$ and $(E^I[\![e_2]\!]\rho) R_\sigma (E^J[\![e_2]\!]\rho')$, so since R is logical

$$E^I[\![e_1 e_2]\!]\rho = (E^I[\![e_1]\!]\rho)(E^I[\![e_2]\!]\rho) R_\tau (E^J[\![e_1]\!]\rho')(E^J[\![e_2]\!]\rho') = E^J[e_1 e_2]\!]\rho'.$$

For $e = \lambda x^\sigma \cdot e_1 : \sigma \to \tau$ let $d_1 \in D^I_\sigma$, $d_2 \in D^J_\sigma$ with $d_1 R_\sigma d_2$. Then $\rho R \rho' \Rightarrow \rho[x^\sigma \mapsto d_1] R \rho'[x^\sigma \mapsto d_2]$ so by induction hypothesis

$$\begin{aligned}(E^I[\![\lambda x^\sigma \cdot e_1]\!]\rho) d_1 &= (E^I[\![e_1]\!]\rho[x^\sigma \mapsto d_1]) R_\tau (E^J[\![e_1]\!]\rho[x^\sigma \mapsto d_2]) \\ &= (E^J[\![\lambda x^\sigma \cdot e_1]\!]\rho') d_2.\end{aligned}$$

Since R is logical, this implies

$(E^I[\![\lambda x^\sigma e_1]\!]\rho) R_{\sigma\to\tau} (E^J[\![\lambda x^\sigma e_1]\!]\rho')$ ∎

3.3. Definition

Let $P = \{P_\sigma\}$ be a property of relations.

R inherits P if

R_A satisfies P_A for all base types A, implies that R_σ satisfies P_σ for all types σ.

We now list some properties of relations $R : I \to J$

1. *Strictness*:
   $\bot^I_\sigma R_\sigma \bot^J_\sigma$.
2. *Inductiveness*:
   For all chains $\{x_n\}$ in D^I_σ, $\{y_n\}$ in D^J_σ: $[\forall n \cdot x_n R_\sigma y_n] \Rightarrow \bigsqcup x_n R_\sigma \bigsqcup y_n$.
3. $\bot$*-reflection*:
   $x R_\sigma \bot^J_\sigma \Rightarrow x = \bot^I_\sigma$.
4. $\top$*-universality*:
   Each D^J_A has a top element $\top^J_A$ (and thus each D^J_σ has a $\top^J_\sigma$) and $x R_\sigma \top^J_\sigma$.
5. *S-monotonicity*:
   $x \sqsubseteq x' R_\sigma y' \sqsubseteq y \Rightarrow x R_\sigma y$.
6. *L-monotonicity*:
   $x \sqsupseteq x' R_\sigma y' \sqsupseteq y \Rightarrow x R_\sigma y$.

We say that a relation $R \subseteq X \times Y$ is *total* if $\forall x \in X \cdot \exists y \in Y \cdot xRy$

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

3.4. Proposition

Every logical relation inherits 1, 2 and 4–6. Every *total* logical relation inherits 3.

Proof

By induction on types, with the base case given (since we are proving inheritance). We give the argument for (3) as illustration. Suppose $fR_{\sigma\to\tau}\perp^J_{\sigma\to\tau}$, and let $a\in D^I_\sigma$. Since R is total, for some $b\in D^J_\sigma$, $aR_\sigma b$ and hence

$$faR_\tau \perp^J_{\sigma\to\tau} b = \perp^J_\tau.$$

By induction hypothesis, this implies $fa = \perp^I_\tau$. We have shown that

$$\forall x\in D^I_\sigma \cdot fa = \perp^I_\tau,$$

i.e. $f = \perp^I_{\sigma\to\tau}$. ∎

Open problem

Give syntactic conditions on properties of relations (e.g. as second-order formulae) which guarantee inheritance by logical relations. ∎

We now apply some of the above concepts to the interpretation of the recursion combinators. From now on, we assume that languages always contain constants $Y_{(\sigma\to\sigma)\to\sigma}$ for each σ. I is a *least fixed point* interpretation if

$$Y^I_{(\sigma\to\sigma)\to\sigma} = \lambda f\in D^I_{\sigma\to\sigma}\cdot \bigsqcup_{n=0}^{\infty} f^n(\perp).$$

3.5. Proposition

Let $R: I\to J$ be a strict, inductive logical relation. Then

$$fR_{\sigma\to\sigma}g \Rightarrow fix(f)R_\sigma fix(g).$$

Proof

$\perp^I_\sigma R_\sigma \perp^J_\sigma$, since R is strict. Now assume $f^n(\perp^I_\sigma)R_\sigma g^n(\perp^J_\sigma)$. Then $fR_{\sigma\to\sigma}g$ and R logical imply that $f^{n+1}(\perp^I_\sigma) = f(f^n(\perp^I_\sigma))R_\sigma g(g^n(\perp^J_\sigma)) = g^{n+1}(\perp^J_\sigma)$.

Thus $\forall n\cdot f^n(\perp^I_\sigma)R_\sigma g^n(\perp^J_\sigma)$, which implies

$$fix(f) = \bigsqcup_{n=0}^{\infty} f^n(\perp^I_\sigma) R_\sigma \bigsqcup_{n=0}^{\infty} g^n(\perp^J_\sigma) = fix(g)$$

since R is inductive. ∎

3.6. Corollary

If I, J are least fixed point interpretations, and $R: I\to J$ is the logical relation induced by strict, inductive relations on the base types, then for all σ

$$Y^I_{(\sigma\to\sigma)\to\sigma} R_{(\sigma\to\sigma)\to\sigma} Y^J_{(\sigma\to\sigma)\to\sigma}$$

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

4. Safety and liveness

The inheritance properties of logical relations suggest that it may be profitable to look for systematic ways of deriving properties at base types. Let

$$f : D \to E$$

be a continuous function between domains. We shall derive two relations from f which will be basic to all our subsequent work.

4.1. Definition

1. R_f the *safety relation* derived from f: $dR_f e \Leftrightarrow fd \sqsubseteq e$.
2. R^f, the *liveness relation* derived from f: $dR^f e \Leftrightarrow fd \sqsupseteq e$.

We record some basic properties of these relations.

4.2. Proposition

1. R_f is inductive and S-monotone. If E has a top element, it is $\top$-universal. It is strict iff f is (i.e. $f \perp_D = \perp_E$) and $\perp$-reflecting iff f is (i.e. $fa = \perp_E \Rightarrow a = \perp_D$).
2. R^f is strict, inductive and L-monotone.

Proof

For illustration, we prove inductiveness of R_f. Suppose $\forall n d_n R_f e_n$, i.e. $\forall n f(d_n) \sqsubseteq e_n$. Then $f(\bigsqcup d_n) = \bigsqcup f(d_n) \sqsubseteq \bigsqcup e_n$. (This is the only part of the proposition which requires continuity of f.) ∎

4.3. Corollary

1. If $f_A : D_A^I \to D_A^J$ is a continuous map for each base type A, and R^S is the logical relation induced from $R_A^S = R_{f_A}$, then: R^S is inductive and S-monotone. It is $\top$-universal (and hence total) if each D_A^J has a top. It is strict iff each f_A is, and $\perp$-reflecting if each f_A is and R^S is total.
2. Now let R^L be induced from $R_A^L = R^{f_A}$. Then R^L is strict, inductive and L-monotone.

Proof

Combine 3.4 and 4.2. ∎

We shall now investigate the provenance of Definition 4.1 a little more deeply. The reader should be aware that the material in the remainder of this section is *not* a prerequisite for the treatment of strictness and termination analysis in Section 5.

Safety and liveness: a categorical view

For the remainder of this section, and also in Section 6, we shall assume familiarity with elementary category theory [10] and lattice theory [24].

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

We return to our basic situation

$$f : D \to E.$$

We shall associate two adjunctions with the continuous map f; they are both instances of Lawvere's adjunction between *existential quantification* and *substitution.*

4.4. The safety adjunction

$$CD \underset{\exists_f^S}{\overset{f^{-1}}{\leftrightarrows}} CE.$$

1. CD, CE are the lattices (hence, categories) of Scott-closed subsets of D, E respectively.
2. $\exists_f^S(B) = \{f(b) \mid b \in B\}^*$, where X^* is the Scott-closure of X.
3. $f^{-1}(C) = \{d \in D \mid f(d) \in C\}$. This is well-defined because f is continuous.

This cuts down to an adjunction

$$P_H D \underset{P_H f}{\overset{f^{-1}}{\leftrightarrows}} P_H E,$$

where P_H is the Hoare powerdomain functor (see, for example, [21]), iff f is strict.

The Abs and Conc maps in [1] are obtained from the abs maps defined there precisely as the components of this adjunction. Our terminology arises from Plotkin's proposal to identify safety properties with closed sets. Note that

$$dR_f e \Leftrightarrow \{d\}^* \subseteq f^{-1}(\{e\}^*) \Leftrightarrow \exists_f^S(\{d\}^*) \subseteq \{e\}^*.$$

4.5. The liveness adjunction

$$AD \underset{\exists_f^L}{\overset{f^{-1}}{\leftrightarrows}} AE.$$

1. AD, AE are the lattices of upper-closed (or 'Alexandrov-open') subsets of D, E respectively.
2. $\exists_f^L(U) = \uparrow\{f(u) \mid u \in U\}$ where $\uparrow X = \{y \mid \exists x \in X \cdot x \sqsubseteq y\}$.
3. $f^{-1}(V) = \{u \in D \mid f(u) \in V\}$. This is well-defined because f is monotone (continuity is *not* needed here).

Note that this adjunction would *fail* if we replaced the Alexandrov-open by the Scott-open subsets. Indeed, by the Adjoint Functor Theorem, we know that f^{-1} has a left adjoint iff it preserves meets; but inverse images of continuous maps do not in general preserve meets in the lattice of Scott-opens (in the language of topos theory, they are *geometric* but not *logical* morphisms).

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

If D, E are bounded-complete algebraic cpo's with bases K(D), K(E), then

$$R^f \upharpoonright (K(D) \times K(E))$$

is the *approximable mapping* associated with f in the sense of Scott [23]. We have:

1. $\perp_D R^f \perp_E$
2. $d \sqsupseteq d' R^f e' \sqsupseteq e \Rightarrow dR^f e$.
3. $dR^f e_1, dR^f e_2 \Rightarrow Con(e_1, e_2)$ and $dR^f(e_1 \sqcup e_2)$.

Our terminology arises from Plotkins's proposal to identify liveness properties with upper-closed sets (= intersections of Scott-open sets). Note that

$$dR^f e \Leftrightarrow \uparrow\{d\} \subseteq f^{-1}(\uparrow\{e\}) \Leftrightarrow \exists_f^L(\uparrow\{d\}) \subseteq \uparrow\{e\}$$

5. Strictness and termination

We now turn to our specific analyses. For the remainder of this section, we fix a language L and a standard interpretation I, subject to the following assumptions:

1. The base types include *bool*.
2. The constants comprise:
   (*a*) recursion combinators $Y_{(\sigma\to\sigma)\to\sigma}$,
   (*b*) conditionals $cond_{bool\to A\to A\to A}$,
   (*c*) first-order functions $c_{A_1,\ldots A_n\to B}(n \geq 0)$.
3. D^I_{bool} is the three-point flat domain

$$\begin{array}{ccc} tt \cdot & & \cdot ff. \\ & \cdot & \\ & \perp & \end{array}$$

4. I is a normal interpretation.
5. Conditionals are interpreted as usual:
   cond tt x y = x
   cond ff x y = y
   cond $\perp$ x y = $\perp$
6. First-order constants are interpreted as *strict, total* functions, i.e. for c: $A_1 \to \ldots A_n \to B$, $c^I a_1 \ldots a_n = \perp^I_B$ iff $\exists i : 1 \leq i \leq n \cdot a_i = \perp^I_{A_i}$.

(The assumption of totality—i.e. the left-to-right implication above—is not used in the proof of correctness of strictness analysis. It is made to simplify the exposition.)

We have two non-standard interpretations, S (for strictness) and T (for

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

termination). For each base type A

$$D_A^S = D_A^T = 2(=\{0, 1\}, 0 \leq 1).$$

S and T are both least fixed point interpretations.

For first-order constants $c: A_1 \to \ldots A_n \to B$, $c^S = c^T = \lambda x_1^2 \ldots \lambda x_n^2 \cdot \bigwedge_{i=1}^n x_i$. S and T differ only (for our particular choice of constants) in their interpretations of conditionals

$$cond^S_{bool \to A \to A \to A} = \lambda x^2 \cdot \lambda y^2 \cdot \lambda z^2 \cdot x \wedge (y \vee z)$$
$$cond^T_{bool \to A \to A \to A} = \lambda x^2 \cdot \lambda y^2 \cdot \lambda z^2 \cdot x \wedge y \wedge z.$$

The intention is that we use S to infer that certain programs do *not* terminate (a safety property), and T to infer that certain programs *do* terminate (a liveness property). In order to formulate these notions precisely, we need to relate S and T to I. For each base type A, we define [12]

$$HALT_A : D_A^I \to 2$$
$$HALT_A(a) = \begin{cases} 1, & a \neq \bot_A^I \\ 0, & a = \bot_A^I \end{cases}.$$

Now let $R_A^S = R_{HALT_A}$, $R_A^T = R^{HALT_A}$.

$$R^S : I \to S,\ R^T : I \to T$$

are the logical relations induced from $\{R_A^S\}$, $\{R_A^T\}$.

5.1. Proposition

1. R^S is strict, inductive, $\top$-universal, $\bot$-reflective and S-monotone.
2. R^T is strict, inductive, and L-monotone.

Proof

By 4.3, we only need to verify that $HALT_A$ is strict and $\bot$-reflective. This is immediate from the definitions (indeed, $HALT_A$ is *uniquely determined* by these properties). ∎

5.2. Lemma

For all $a \in D_A^I$:

1. $aR_A^T 0$.
2. $aR_A^T 1 \Leftrightarrow a \neq \bot_A^I$.

Proof

Immediate from the definitions. ∎

We shall now check that the interpretations of constants in S and T are correctly related to those in I. It will be convenient to use the following

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

observation: for any logical relation $R: J \to K$

$$(*)\ fR_{\sigma_1 \to \ldots \sigma_n \to \tau} g \Leftrightarrow \forall a_1, \ldots, a_n \cdot \forall b_1, \ldots, b_n. \bigwedge_{i=1}^{n} a_i R_{\sigma_i} b_i \Rightarrow (fa_1 \ldots a_n) R_\tau (gb_1 \ldots b_n),$$

which is just a trivial manipulation of the definitions.

5.3. Lemma

For each first-order constant $c: \sigma$, with $\sigma = A_1 \to \ldots A_n \to B$:

1. $c^I R^S_\sigma c^S$.
2. $c^I R^T_\sigma c^T$.

Proof

1. Let $a_1, \ldots a_n, b_1, \ldots b_n$ be given as in (*), with $\bigwedge_i a_1 R^S_{A_i} b_i$. We consider two cases:
   - I. $\exists i: 1 \le i \le n \cdot b_i = 0$. In this case, $\bigwedge_i b_i = 0$, but since R^S is $\bot$-reflecting, $a_i = \bot^I_{A_i}$, hence $c^I a_1 \ldots a_n = \bot^I_B R^S_B 0$ since R^S is strict.
   - II. $\forall i: 1 \le i \le n \cdot b_i = 1$ in this case, $\bigwedge_i b_i = 1$ and since R^S is $\top$-universal,
   $$c^I a_1 \ldots a_n R^S_B 1.$$
2. Again, let $a_1, \ldots, a_n, b_1, \ldots b_n$ be given with $\bigwedge_i a_i R^T_{A_i} b_i$
   - I. $\exists i: 1 \le i \le n \cdot b_i = 0$ In this case $\bigwedge_i b_i = 0$, and by 5.2 (i),
   $$c^I a_1 \ldots a_n R^T_B 0.$$
   - II. $\forall i: 1 \le i \le n \cdot b_i = 1$. In this case, $\bigwedge_i b_i = 1$, but by 5.2 (ii), $\forall i: 1 \le i \le n \cdot a_i \ne \bot^I_{ai}$, and hence $c^I a_1 \ldots a_n \ne \bot^I_B$ (this is where totality of the first-order functions is used), and so
   $$c^I a_1 \ldots a_n R^T_B 1$$
   by 5.2 (2) again. ∎

5.4. Lemma

For each conditional *cond*: σ, with $\sigma = bool \to A \to A \to A$.

1. $cond^I R^S_\sigma cond^S$.
2. $cond^I R^T_\sigma cond^T$.

Proof

1. Again let $a_1, a_2, a_3, b_1, b_2, b_3$ be given as in (*). We consider two cases:
   - *I.* $b_1 = 0$. Since R^S is $\bot$-reflecting , $a_1 = \bot^I_{bool}$, and $cond^I a_1 a_2 a_3 = \bot^I_A R^S_A 0 = cond^S b_1 b_2 b_3$.
   - II. $b_1 = 1$. By $\top$-universality of R^S, we need only consider the case

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

where $b_2 = b_3 = 0$. In this case, by $\perp$-reflection $a_2 = a_3 = \perp^I_A$, and

$$cond^I a_1 a_2 a_3 = \perp^I_A R^S_A 0 = cond^S b_1 b_2 b_3.$$

2. similar ∎

Collecting these results together, we obtain:

5.5. Proposition

For all constants $c : \sigma$ of L:

1. $c^I R^S_\sigma c^S$.
2. $c^I R^T_\sigma c^T$.

Proof

Combine 3.6, 5.3, and 5.4. ∎

We can now formulate our correctness criterion for strictness analysis.

Correctness of S

$$\forall e : \sigma \to \tau \cdot \forall \rho \in Env^I, \quad \rho' \in Env^S : \rho R^S \rho'.$$

$$E^S[\![e]\!]\rho' \perp^S_\sigma = \perp^S_\tau \Rightarrow E^I[\![e]\!]\rho \perp^I_\sigma = \perp^I_\tau.$$

This says that, if e is strict under S, it denotes a strict function under the standard semantics. Thus we can soundly infer that a function is strict by testing for strictness in the non-standard finite domain $D^S_{\sigma\to\tau}$. This is the main result in [1].

5.6. Theorem

S is correct.

Proof.

Let e, ρ, ρ' be given as in the statement of correctness. Now

$$(E^I[\![e]\!]\rho) R^S_{\sigma\to\tau} (E^S[\![e]\!]\rho')$$

by 3.2 and 5.5, and $\perp^I_\sigma R^S_\sigma \perp^S_\sigma$ since R^S is strict. Since R^S is logical, this implies

$$(E^I[\![e]\!]\rho \perp^I_\sigma) R^S_\tau (E^S[\![e]\!]\rho' \perp^S_\sigma).$$

But $E^S[\![e]\!]\rho' \perp^S_\sigma = \perp^S_\tau$ by hypothesis, and since is R^S $\perp$-reflecting, this implies

$$E^I[\![e]\!]\rho \perp^I_\sigma = \perp^I_\tau$$

as required. ∎

We now turn to the correctness of T. What this means at higher types is less clear than for S, so we will only formulate it for first-order types.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

Correctness of T

$$\forall e : A_1 \to \ldots A_n \to T \cdot \forall \rho \in Env^I,\ \rho' \in Env^T : \rho R^T \rho'.$$
$$E^T[\![e]\!]\rho' 10 \ldots 0 = 1 \Rightarrow$$
$$\forall a_1 \in D^I_{A_1}, \ldots, a_n \in D^I_{A_n} \cdot a_1 \neq \perp^I_{A_1} \Rightarrow$$
$$E^I[\![e]\!]\rho a_1 \ldots a_n \neq \perp^I_B.$$

This says that if termination of the first argument of the function denoted by T guarantees termination of the result in T, then we may soundly infer that this property holds in the standard interpretation. This can obviously be generalized to combinations of arguments, but for notational simplicity, we shall use the above formulation.

5.7. Theorem

T is correct.

Proof

Let $e, \rho, \rho', a_1, \ldots, a_n$ be given as in the statement of correctness, and write $\sigma = A_1 \to \ldots A_n \to B$. Now $a_1 \neq \perp^I_{A_1} \Rightarrow a_1 R^T_{A_1} 1$ by 5.2 (ii), and $a_i R^T_{A_i} 0$, $1 \leq i \leq n$, by 5.2 (i). Also, by 3.2 and 5.5

$$(E^I[\![e]\!]\rho) R^T_\sigma (E^T[\![e]\!]\rho')$$

and so

$$(E^I[\![e]\!]\rho a_1, \ldots, a_n) R^T_B (E^T[\![e]\!]\rho' 10 \ldots 0)$$

since R^T is logical. By hypothesis, $E^T[\![e]\!]\rho' 10 \ldots 0 = 1$, and so by 5.2 (ii)

$$E^I[\![e]\!]\rho a_1 \ldots a_n \neq \perp^I_B$$

as required. ∎

The reader is invited to compare the proofs in [14], [17], [1] with the above arguments.

As a corollary of theorems 5.6 and 5.7, we obtain.

5.8. The sandwich lemma

$$\forall e : A,\ \forall \rho \in Env^I,\quad \rho_1 \in Env^T,\quad \rho_2 \in Env^S \cdot \rho R^T \rho_1 \wedge \rho R^S \rho_2 \Rightarrow$$
$$E^T[\![e]\!]\rho_1 \sqsubseteq HALT_A(E^I[\![e]\!]\rho) \sqsubseteq E^S[\![e]\!]\rho_2.$$

A *program* is a closed term of ground type. As an instructive and amusing consequence of 5.8, we have:

5.9. Theorem

The halting problem for programs with no occurrences of the conditional is decidable.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

Proof

It is only in the interpretation of the conditional that S and T differ. Hence for a program e with no occurrences of the conditional

$$E^T[\![e]\!] = E^S[\![e]\!]$$

and by the Sandwich Lemma

$$HALT_A(E^I[\![e]\!]) = E^T[\![e]\!] = E^S[\![e]\!].$$

As all domains are finite and the constants are explicitly tabulated, the fixpoints can be computed by successive approximations in the standard way. Thus, '$HALT_A(E^I[\![e]\!]) = 1$' is recursive, since both E^T and E^S are. ∎

6. Abstraction functions and Kan extensions

Our treatment of strictness and termination analysis exemplifies the following schematic approach. Let I be the 'standard' interpretation, and J the 'non-standard' or 'abstract' interpretation. First, we define abstraction functions

$$f_A : D^I_A \to D^J_A$$

for each base type A; then derive relations R_A from f_A; and finally induce a logical relation R from the R_A. In this section, we shall consider the possibility of defining abstraction functions

$$abs_\sigma : D^I_\sigma \to D^J_\sigma$$

at all types, starting from $abs_A = f_A$.

Intuitively, the idea is to pick out, from those abstract values related to a given concrete value, the 'best' one. One important application of such functions is to the interpretation of constants. Given a constant c_σ, with standard interpretation c^I_σ, we want to define c^J_σ as the 'best possible' abstraction of c^I_σ with respect to our relation R. If we have abstraction functions at all types, we can just set $c^J_\sigma = abs_\sigma(c^I_\sigma)$.

We shall begin by developing a theory of abstraction functions in a rather general setting. Our theory will be based on the safety/liveness duality.

First, we need to consider a modified notion of interpretation. Given $I = (\{D^I_A\}, \{c^I_\sigma\})$, our definition of $\{D^I_\sigma\}$ builds the *continuous* type structure over the $\{D^I_A\}$. We shall now consider the *monotone* type structure over I, MD^I_σ, defined by

$$MD^I_A = D^I_A$$

$$MD^I_{\sigma\to\tau} = [MD^I_\sigma \to_m MD^I_\tau]$$

where for posets D, E

$$[D \to_m E]$$

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

is the poset of monotone maps from D to E, ordered pointwise. We also drop the stipulation that the base domains D^I_A are cpo's; they may be arbitrary posets.

Note that the notion of a relation, and the process of uniquely extending a relation on the base types to a logical relation, still make sense for monotone type structures (indeed in a very much greater generality—see, for example, [11]), and Proposition 3.2 still holds good in this setting.

For the remainder of this section, we will make the following assumptions:

(A1) We are given interpretations I, J such that each D^J_A (hence each MD^J_σ) is a complete lattice.

(A2) We are given monotone maps

$$abs_A : D^I_A \to D^J_A$$

for each base type A.

On this basis, we shall define *safety abstraction maps*

$$abs^S_\sigma : MD^I_\sigma \to MD^J_\sigma$$

and *liveness abstraction maps*

$$abs^L_\sigma : MD^I_\sigma \to MD^J_\sigma$$

for all σ.

6.1. Definition

(i) $abs^S_A = abs_A$
$abs^S_{\sigma\to\tau} fd = \bigsqcup \{(abs_\tau \circ f)d' \mid abs_\sigma d' \sqsubseteq d\}$.

(ii) $abs^L_A = abs_A$
$abs^L_{\sigma\to\tau} fd = \bigsqcap \{(abs_\tau \circ f)d' \mid abs_\sigma d' \sqsupseteq d\}$.

6.2. Proposition

abs^S_σ, abs^L_σ are well-defined monotone maps for all σ.

Proof

By induction of σ. The base case is given by our definitions. For the inductive step, we must show that $abs^S_{\sigma\to\tau} fd$, $abs^L_{\sigma\to\tau} fd$ are monotone in both arguments—monotonicity in d is required for these functions to be well defined.

(I) $abs^S_{\sigma\to\tau} fd$
Let $X_{f,d} = \{(abs^S_\tau \circ f)d' \mid abs^S_\sigma d' \sqsubseteq d\}$
Then $f \sqsubseteq g \Rightarrow \forall x \in X_{f,d} \cdot \exists y \in X_{g,d} \cdot x \sqsubseteq y \Rightarrow \bigsqcup X_{f,d} \sqsubseteq \bigsqcup X_{g,d}$,

which proves monotonicity in f.

Again, $d_1 \sqsubseteq d_2 \Rightarrow X_{f,d_1} \subseteq X_{f,d_2} \Rightarrow \bigsqcup X_{f,d_1} \sqsubseteq \bigsqcup X_{f,d_2}$, which proves monotonicity in d.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

(II) $abs^L_{\sigma\to\tau} fd$.
Let $Y_{f,d} = \{(abs^L_\tau \circ f)d' \mid abs^L_\sigma d' \sqsupseteq d\}$.

Then $f \sqsubseteq g \Rightarrow \forall y \in Y_{g,d} \cdot \exists x \in Y_{fd} \cdot x \sqsubseteq y$

$$\Rightarrow \bigsqcap Y_{fd} \sqsubseteq \bigsqcap Y_{g,d},$$

which proves monotonicity in f.

Again, $d_1 \sqsubseteq d_2 \Rightarrow Y_{f,d_1} \supseteq Y_{f,d_2}$

$$\Rightarrow \bigsqcap Y_{f,d_1} \sqsubseteq \bigsqcap Y_{f,d_2},$$

which gives monotonicity in d. ∎

We shall now establish a fundamental property of abstraction functions with respect to application.

6.3. The semi-homomorphism property

(i) $(abs^S_{\sigma\to\tau} f)(abs^S_\sigma a) \sqsupseteq abs^S_\tau(fa)$.
(ii) $(abs^L_{\sigma\to\tau} f)(abs^L_\sigma a) \sqsubseteq abs^L_\tau(fa)$.

Diagrammatically

$$\begin{array}{ccc} MD^I_{\sigma\to\tau} \times MD^I_\sigma & \xrightarrow{apply^I_{\sigma,\tau}} & MD^I_\tau \\ \downarrow{\scriptstyle abs_{\sigma\to\tau} \times abs_\sigma} & & \downarrow{\scriptstyle abs_\tau} \\ MD^J_{\sigma\to\tau} \times MD^J_\sigma & \xrightarrow{apply^J_{\sigma,\tau}} & (R)MD^{J^{(R)}}_\tau \end{array}$$

with (R) = ⊒ for safety, and (R) = ⊑ for liveness.

Proof

1. $(abs^S_{\sigma\to\tau} f)(abs^S_\sigma a) = \bigsqcup X$,
where $X = \{(abs^S_\tau \circ f)a' \mid abs^S_\tau a' \sqsubseteq abs^S_\tau a\}$.
But $(abs^S_\tau \circ f)a \in X$, so $(abs^S_\tau \circ f)a \sqsubseteq (abs^S_{\sigma\to\tau} f)(abs^S_\sigma a)$.
2. Dual to (i). ∎

We can now state a striking result which connects abstraction functions to logical relations in a very strong way. Let R^S be the logical relation induced from

$$R^S_A = R_{abs_A}.$$

Similarly, define R^L as the logical relation induced from

$$R^L_A = R^{abs_A}.$$

Then we have

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

6.4. The uniformization theorem

For all σ

(i) $R^S_\sigma = R_{abs^S_\sigma}$.

(ii) $R^L_\sigma = R^{abs^L_\sigma}$.

Proof

(i) By induction on σ. The base case is given from the definitions. At the inductive step, we must show

$$fR^S_{\sigma\to\tau}g \Leftrightarrow abs^S_{\sigma\to\tau}(f) \sqsubseteq g$$

i.e.

$$[\forall a, b \cdot aR^S_\sigma b \Rightarrow faR^S_\tau gb] \Leftrightarrow abs^S_{\sigma\to\tau}(f) \sqsubseteq g$$

or equivalently, using the inductive hypothesis

$$[\forall a, b \cdot abs^S_\sigma a \sqsubseteq b \Rightarrow (abs^S_\tau \circ f)a \sqsubseteq gb] \Leftrightarrow abs^S_{\sigma\to\tau}(f) \sqsubseteq g.$$

The left-hand side of this equivalence reduces to

$$(*)\forall a \cdot (abs^S_\tau \circ f)a \sqsubseteq (g \circ abs^S_\sigma)a.$$

To see that it implies (*), take $b = abs_\sigma a$, while the converse follows from the monotonicity of g. So what we must prove is

(†) $abs^S_\tau \circ f \sqsubseteq g \circ abs^S_\sigma \Leftrightarrow abs^S_{\sigma\to\tau}(f) \sqsubseteq g$.

($\Leftarrow$) $abs^S_\tau(fa) \sqsubseteq (abs^S_{\sigma\to\tau} f)(abs^S_\sigma a)$ by the semi-homomorphism property, $\sqsubseteq g(abs^S_\sigma a)$ by hypothesis.

($\Rightarrow$) $abs^S_{\sigma\to\tau} fa = \bigsqcup X$,
where $X = \{(abs^S_\tau \circ f)b \mid abs^S_\sigma b \sqsubseteq a\}$.
By hypothesis, $\forall b \cdot abs^S_\sigma b \sqsubseteq a \Rightarrow (abs^S_\tau \circ f)b \sqsubseteq (g \circ abs^S_\sigma)b \sqsubseteq ga$, so ga is an upper bound for X.
Hence $abs^S_{\sigma\to\tau} fa \sqsubseteq ga$.

(ii) Dual to (i). ∎

In order to interpret this result, note that as a Corollary we have that

$$graph(abs^S_\sigma) \subseteq R^S_\sigma,$$

$$graph(abs^L_\sigma) \subseteq R^L_\sigma.$$

Thus abs^S_σ *uniformizes* the relation R^S_σ and, moreover, is the best possible function which does so, in the sense that, for each $d \in D'_\sigma$, $abs^S_\sigma(d)$ is the least element of the set $\{d' \in D'_\sigma \mid dR^S_\sigma d'\}$. The situation for abs^L_σ is dual.

The generality of our definitions and arguments to date will perhaps have made the reader suspect that what we are really doing is category theory. This suspicion is fully justified; what we have been doing is *exactly* the construction of Kan extensions for poset categories, albeit in an 'iterated' fashion which may be novel.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

We shall now spell out the correspondence explicitly, since Kan extensions do not seem to have found their way into the Computer Science literature to date. Our reference for Kan extensions will be [10], pp. 232–34. We begin by stating the definitions for the special case where the categories are posets and the functors monotone maps.

Definition

Let

$$\begin{array}{ccc} M & \xrightarrow{T} & A \\ {\scriptstyle K}\downarrow & & \\ C & & \end{array}$$

be a diagram of posets and monotone maps. A *right Kan extension* of T along K is a monotone map

$$R : C \to_m A$$

satisfying

1. $R \circ K \sqsubseteq T$.
2. $\forall S : C \to_m A.\ S \circ K \sqsubseteq T \Rightarrow S \sqsubseteq R$.

   Dually, a left Kan extension of T along K is a monotone map

   $$L : C \to_m A$$

   satisfying.
3. $L \circ K \sqsupseteq T$.
4. $\forall S : C \to_m A.\ S \circ K \sqsupseteq T \Rightarrow S \sqsupseteq L$.

Conditions (1) and (2) are easily seen to be equivalent to

$$\text{(RKE)} \quad \forall S : C \to_m A. \quad S \circ K \sqsubseteq T \Leftrightarrow S \sqsubseteq R$$

and similarly (3) and (4) are equivalent to

$$\text{(LKE)} \quad \forall S : C \to_m A. \quad S \circ K \sqsupseteq T \Leftrightarrow S \sqsupseteq L$$

Now suppose that A is a complete lattice. The formula for computing the right Kan extension as a pointwise limit—MacLane's equation (3) on p. 234 becomes

$$(*) \quad Rc = \bigsqcap \{Ta \mid Ka \sqsupseteq c\}.$$

Dually, the left formula for the left Kan extension reads

$$(**) \quad Lc = \bigsqcup \{Ta \mid Ka \sqsubseteq c\}.$$

Note that the problem of constructing a right Kan extension of T along K specializes to the problem of constructing a left adjoint of K when $T = id$. In this case, the formula (*) above specializes to the Adjoint Functor Theorem.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

Dually, left Kan extensions specialize to right adjoints. Thus as MacLane remarks [10], pV): 'the notion of Kan extensions . . . is the deeper form of the basic construction of adjoints'.

We are now ready to spell out the correspondence with our construction of abstraction functions. The situation is

$$\begin{array}{ccc} MD^I_\sigma & \xrightarrow{f} & MD^I_\tau \\ {\scriptstyle abs_\sigma}\downarrow & & \downarrow{\scriptstyle abs_\tau} \\ MD^J_\sigma & & MD^J_\tau \end{array}$$

The correspondence with the previous diagram is between abs_σ and K and $(abs_\tau \circ f)$ and T. The condition (LKE) for $abs^S_{\sigma\to\tau}(f)$ to be the left Kan extension of abs^S_σ along $abs^S_\tau \circ f$ then becomes

$$\forall g \in MD^J_{\sigma\to\tau} \cdot g \circ abs^S_\sigma \sqsupseteq abs^S_\sigma \circ f \Leftrightarrow g \sqsupseteq abs^S_{\sigma\to\tau}(f).$$

This is exactly the formula (†) which emerged in the proof of the Uniformization Theorem. The formula (**) for the pointwise construction of $abs^S_{\sigma\to\tau}(f)$ becomes

$$abs^S_{\sigma\to\tau} fc = \bigsqcup\{(abs^S_\tau \circ f)a \mid abs^S_\tau a \sqsubseteq c\}$$

which is exactly our definition of $abs^S_{\sigma\to\tau}$. The correspondence between $abs^L_{\sigma\to\tau}$ and the right Kan extension goes through in the same way.

We have shown

6.5. Observation

1. $abs^S_{\sigma\to\tau}(f)$ is the left Kan extension of $abs^S_\tau \circ f$ along abs^S_σ.
2. $abs^L_{\sigma\to\tau}(f)$ is the right Kan extension of $abs^L_\tau \circ f$ along abs^L_σ.

The connection with Kan extensions made by 6.5 is 'local' in character; it concerns the pointwise construction of abstraction maps.

From a more global point of view, we have the following situation:

$$\begin{array}{ccc} T_{ccc}(L) & \xrightarrow{E^J} & J \\ {\scriptstyle E^I}\downarrow & & \\ I & & \end{array}$$

a diagram of Cartesian closed categories (CCC's) and CCC-morphisms.[1] Here $T_{CCC}(L)$ is the initial CCC over the language L, and E^I, E^J are the (suitably factored) semantic functions induced by our interpretations I and J, viewed now as CCC's. We can consider the construction of Kan extensions for this situation; and we might hope that $abs^L = \{abs^L_\sigma\}$, $abs^S = \{abs^S\}$,

[1] Actually, this is somewhat imprecise, as we have not introduced product types. However, our remarks in this paragraph should be sufficiently clear, at least at a heuristic level.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

viewed globally as putative CCC-morphisms, would provide such constructions. However, this is not the case; abs^S and abs^L are *not* functorial, but only 'semi-functorial', (*lax* in the two-categorical sense, cf. [9]) in the sense of 6.3 and for the same reasons.

We shall now record the little we presently know about these global structures.

We shall say that $F = \{F_\sigma\}$

$$F_\sigma : MD^I_\sigma \to_m MD^J$$

is a *safety morphism* if

$$\forall e : \sigma \in \Lambda_T(L) \cdot \forall \rho^I \in Env^I \cdot \forall \rho^J \in Env^J$$
$$F \circ \rho^I \sqsubseteq \rho^J \Rightarrow F_\sigma(E^I[\![e]\!]\rho^I) \sqsubseteq E^J[\![e]\!]\rho^J.$$

Dually, G is a *liveness morphism* if

$$\forall e : \sigma \in \Lambda_T(L) \cdot \forall \rho^I \in Env^I \cdot \forall \rho^J \in Env^J$$
$$G \circ \rho^I \sqsupseteq \rho^J \Rightarrow G_\sigma(E^I[\![e]\!]\rho^I) \sqsupseteq E^J[\![e]\!]\rho^J.$$

Note that, if every $d \in \bigcup D^I_\sigma$ is denoted by some closed term (or if the domains D^I_0 are algebraic and every *compact* element is denoted by some closed term) then every safety morphism approximates every liveness morphism (where $F \sqsubseteq G \equiv \forall_\sigma F_\sigma \sqsubseteq G_\sigma$).

Now we have:

6.6. Theorem

1. abs^S is a safety morphism iff for all constants $c_\sigma \in L$, $abs^S_\sigma(c^I_\sigma) \sqsubseteq c^J_\sigma$.
2. abs^L is a liveness morphism iff for all $c_\sigma \in L$, $abs^L_\sigma(c^I_\sigma) \sqsupseteq c^J_\sigma$.

Proof

We prove (i) only. Necessity is obvious. For sufficiency, let R^S be the logical relation induced from $R^S_A = R_{abs_A}$. By the Uniformization theorem, $R^S_\sigma = R_{abs^S_\sigma}$ for all σ. Hence by the hypothesis $c^I_\sigma R^S_\sigma c^J_\sigma$ for all $c_\sigma \in L$. Again, $abs^S \circ \rho^I \sqsubseteq \rho^J$ iff $\rho^I R^S \rho^J$. The result now follows from 3.2. ∎

Note that the Sandwich Lemma (5.8) and Lemma 5.2 of [1] are special cases of this result.

As a final remark on the global aspects of the abstraction maps, say that $F = \{F_\sigma\}$ has the *safety semi-homomorphism property* if for all σ, τ

$$F_{\sigma\to\tau}(f)F_\sigma(s) \sqsupseteq F_\tau(fa)$$

and similarly the *liveness semi-homomorphism property* is

$$F_{\sigma\to\tau}(f)F_\sigma(a) \sqsubseteq F_\tau(fa).$$

Now it is easy to see that, if F has the safety semi-homomorphism property,

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

then

$$F_{\sigma\to\tau}fb \supseteq \bigsqcup \{(F_\tau \circ f)d \mid F_\sigma d \sqsubseteq b\}$$

while if F has the liveness semi-homomorphism property then

$$F_{\sigma\to\tau}fb \sqsubseteq \bigsqcap \{(F_\tau \circ f)d \mid F_\tau d \supseteq b\}.$$

Continuous Kan extensions

We have developed a general theory of abstraction functions. However, for applications in abstract interpretation we are usually interested in the continuous rather than the monotone type structures; while for various reasons we may also wish the abstraction functions themselves to be continuous. This suggests two progressive requirements:

(C1.S) The functions

$$abs^S_\sigma : D^I_\sigma \to D^J_\sigma$$

are well-defined on the *continuous* type structure.

(C2.S) The functions abs^S_σ from (C1.S) are continuous. Similarly we have

(C1.L) The functions

$$abs^L_\sigma : D^I_\sigma \to D^J_\sigma$$

are well-defined.

(C2.L) The functions of (C1.L) are continuous.

We shall now investigate conditions for these requirements to be satisfied. For the rest of this section, we shall operate under the following additional assumptions:

(A3) Each D^I_A, hence D^I_σ is a bounded-complete algebraic cpo (i.e. a *domain* in the sense of [23]).

(A4) Each D^I_A, hence each D^I_σ, is an algebraic lattice.

(A5) Each $abs_A : D^I_A \to D^J_A$ is continuous.

Based on these assumptions, we shall develop a rather complete theory of conditions under which (C1.S) and (C2.S), or (C1.L) and (C2.L) are satisfied. This theory maintains the duality between safety and liveness; and shows how power-domains arise inevitably when continuous abstraction maps are required.

We begin with safety.

6.7. Definition

Let

$$f : D \to E$$

be a continuous map of algebraic cpo's. Then f *satisfies property* (F) iff it maps finite elements of D to finite elements of E.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

6.8. Proposition

Suppose each abs_A satisfies (F). Then for all σ, abs^S_σ satisfies (F), (C1.S), and (C2.S).

Proof

By induction on σ. The base case is given. For the inductive step, we must show:

(i) $abs^S_{\sigma\to\tau} fd$ is continuous in d (this gives (C1.S)),
(ii) $abs^S_{\sigma\to\tau} fd$ is continuous in f (this gives (C2.S)),
(iii) f finite $\Rightarrow abs^S_{\sigma\to\tau} f$ finite (this gives (F)).

(i) Continuity in d. As we have already shown that $abs^S_{\sigma\to\tau}$ is monotonic in f and d, we need to show that

$$abs^S_{\sigma\to\tau} f \bigsqcup_{n=0}^{\infty} d_n \sqsubseteq \bigsqcup_{n=0}^{\infty} abs^S_{\sigma\to\tau} f d_n$$

i.e.
$$\bigsqcup X \sqsubseteq \bigsqcup_{n=0}^{\infty} \left(\bigsqcup X_n\right)$$

where
$$X = \{(abs^S_\tau \circ f)d' \mid abs^S_\sigma d' \sqsubseteq \bigsqcup d_n\}$$
$$X_n = \{(abs^S_\tau \circ f)d' \mid abs^S_\sigma d' \sqsubseteq d_n\}.$$

Now it is sufficient to show that for every finite $b \sqsubseteq \bigsqcup X$, $\exists n \cdot b \sqsubseteq \bigsqcup X_n$. Given such a b, note that

$$b \sqsubseteq (abs^S_\tau \circ f)d' \Rightarrow \exists \text{finite } b' \cdot b' \sqsubseteq d' \;\&\; b \sqsubseteq (abs^S_\tau \circ f)b'$$

since abs^S_τ is continuous by inductive hypothesis. Now $abs^S_\sigma b' \sqsubseteq abs^S_\sigma d' \sqsubseteq \bigsqcup d_n$, and since by inductive hypothesis abs^S_σ satisfies (F), $abs^S_\sigma b'$ is finite. Hence $\exists n \cdot abs^S_\sigma b' \sqsubseteq d_n$ and $b \sqsubseteq \bigsqcup X_n$, as required.

(ii) Continuity in f. Again, we only need to show

$$abs_{\sigma\to\tau} \bigsqcup_{n=0}^{\infty} f_n d \sqsubseteq \bigsqcup_{n=0}^{\infty} abs^S_{\sigma\to\tau} f_n d,$$

i.e.
$$\bigsqcup Y \sqsubseteq \bigsqcup_{n=0}^{\infty} \left(\bigsqcup Y_n\right)$$

where
$$Y = \{(abs^S_\tau \circ \bigsqcup_{n=0}^{\infty} f_n)d' \mid abs^S_\sigma d' \sqsubseteq d\}$$
$$Y_n = \{(abs^S_\tau \circ f_n)d' \mid abs^S_\sigma d' \sqsubseteq d\}.$$

Once again we consider any finite $b \sqsubseteq \bigsqcup Y$. Note that

$$b \sqsubseteq \left(abs^S_\tau \circ \bigsqcup_{n=0}^{\infty} f_n\right)d' = \bigsqcup_{n=0}^{\infty} (abs^S_\tau \circ f_n)d' \text{ since } abs^S_\tau \text{ is continuous by } IH$$

$$\Rightarrow \exists n \cdot b \sqsubseteq (abs^S_\tau \circ f_n)d'.$$

This immediately yields $b \sqsubseteq Y_n$ as required.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

(iii) $abs^S_{\sigma\to\tau}$ satisfies (F). Let $f = \bigsqcup\{[b_1, e_1], \ldots, [b_n, e_n]\}$, for step functions $[b_i, e_i]$. We claim that

$$abs^S_{\sigma\to\tau} f = \bigsqcup\{[abs^S_\sigma(b_1), abs^S_\tau(e_1)], \ldots, [abs^S_\sigma(b_n), abs^S_\sigma(e_n)]\}.$$

First, by inductive hypothesis abs^S_σ and abs^S_τ both satisfy (F), so $abs^S_\sigma(b_i)$, $abs^S_\tau(e_i)$ are finite, $i = 1, \ldots, n$. Now

$$\begin{aligned} abs^S_{\sigma\to\tau} f d &= \bigsqcup\{(abs^S_\tau \circ f)d' \mid abs^S_\sigma d' \sqsubseteq d\} \\ &= \bigsqcup\{abs^S_\tau(e_i) \mid abs^S_\sigma b_i \sqsubseteq d\} \\ &= (\bigsqcup\{[abs^S_\sigma(b_1), abs^S_\tau(e_1)], \ldots, [abs^S_\sigma(b_n,) abs^S_\tau(e_n)]\})d. \ \blacksquare \end{aligned}$$

Note that the proof of (ii) did *not* use property (F). Thus we can draw the following:

6.9. Corollary

If abs^S_σ satisfies (C1.S) for all σ, then abs^S_σ satisfies (C2.S) for all σ.

6.10. Proposition

A necessary condition for abs^S to satisfy (C1.S) is that abs_A satisfies (F) for all base types A.

Proof

Suppose for some A, abs_A does not satisfy (F). Then for some finite $b \in D'_A$ $abs_A(b) = d = \bigsqcup d_n$ an infinite lub of finite elements. Now we consider two cases.

Case 1. abs_A is a constant function. Then

$$abs^S_{A\to A} id_{D'_A} d = \bigsqcup\{d\} = d$$

while for all n

$$abs^S_{A\to A} id_{D'_A} d_n = \bigsqcup \varnothing = \bot,$$

so $abs^S_{A\to A}$ does not satisfy (C1.S).

Case 2. If abs_A is not a constant function, for some finite $e \in D'_A$ we must have $abs_A(e) \neq abs_A(\bot)$. Now

$$abs^S_{A\to A}[b, e]d = abs_A(e)$$

while for all n

$$abs^S_{A\to A}[b, e]d_n \sqsubseteq \bigsqcup\{abs_A(\bot)\} = abs_A(\bot).$$

Thus $abs^S_{A\to A}$ does not satisfy (C1.S) in this case either. ∎

We summarize our progress so far in

6.11. Theorem

A necessary and sufficient condition for abs^S to satisfy C1.S and C2.S is that all abs_A satisfy (F).

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

We shall now connect our discussion of continuity of abs^S with the use of powerdomains in abstract interpretation. We begin by recalling some basic facts and definitions.

6.12. Definition

1. P_H° is the *Hoare powerdomain functor* (with empty set)

$$P_H^\circ D = (CD, \subseteq)$$
$$P_H^\circ(f)S = \{fs \mid s \in S\}^*.$$

2. If L is an algebraic lattice, then the least upper bound operation, viewed as a function

$$\bigsqcup_L : P_H^\circ L \to L$$

is continuous. Moreover

$$\bigsqcup_L X^* = \bigsqcup X.$$

Now suppose we are given a continuous map $f : D \to E$ between algebraic domains.

6.13. Definition

The *concretization map* associated with f is

$$\gamma_f : E \to P_H^\circ(D)$$
$$\gamma_f(e) = \{d \mid fd \sqsubseteq e\}.$$

This map is always well-defined (this requires the presence of the empty set in the powerdomain) and monotonic. Note that

$$\gamma_f(e) = f^{-1}(\{e\}^*)$$

in terms of the safety adjunction of Section 4.

6.14. Proposition

γ_f is continuous iff f satisfies (F).

Proof

Suppose f satisfies (F). Then for all finite $b \in D$

$$b \in \gamma_f(\bigsqcup e_n) = \{d \mid fd \sqsubseteq \bigsqcup e_n\}$$
$$\Rightarrow fb \sqsubseteq \bigsqcup e_n \Rightarrow \exists n \cdot fb \sqsubseteq e_n$$
$$\textit{since } f\ b \textit{ is finite} \Rightarrow b \in \gamma_f(e_n).$$

So γ_f is continuous.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

Now suppose that f does not satisfy (F). Then for some finite $b \in D$, $fb = \bigsqcup e_n$, an infinite lub of finite elements. Now $b \in \gamma_f(\bigsqcup e_n)$, but for all n, $b \notin \gamma_f(e_n)$, so for all n, $\{b\}^* \not\subseteq \gamma_f(e_n)$. Since $\{b\}^*$ is a finite element of $P_H^\circ(D)$, this implies $\{b\}^* \not\subseteq \bigsqcup_{n=0}^{\infty} \gamma_f(e_n)$, and so γ_f is not continuous. ∎

Note that if abs_σ^S, abs_τ^S, $\gamma_{abs_\sigma^S}$ are continuous, and if f is a continuous map from D_σ^I to D_σ^γ we can define

$$(PDS)\varphi(f) = \bigsqcup_L \circ P_H^\circ(abs_\tau \circ f) \circ \gamma_{abs_\sigma}$$

We have

$$\begin{aligned}\varphi(f)d &= \bigsqcup_L \circ P_H^\circ(abs_\tau \circ f) \circ \gamma_{abs_\sigma}(d) \\ &= \bigsqcup (\{(abs_\tau^S \circ f)d' \mid abs_\sigma^S d' \sqsubseteq d\}^*) \\ &= \bigsqcup \{(abs_\tau^S \circ f)d' \mid abs_\sigma^S d' \sqsubseteq d\} \\ &= abs_{\sigma\to\tau}^S fd.\end{aligned}$$

Moreover, in this case $abs_{\sigma\to\tau}^S$ is clearly continuous. Thus we have:

6.15. Theorem

A necessary and sufficient condition for abs^S to satisfy C1.S and C2.S is that it can be defined by (PDS) via the Hoare powerdomain.

We now turn to liveness, our treatment of which will parallel that for safety. Once again, we consider the basic situation $f : D \to E$, f a continuous map of algebraic domains.

6.16. Definition

f satisfies property (C) iff, for all $e \in E$, $\{d \in D \mid fd \sqsupseteq e\}$ is Scott compact; i.e. there is a finite set $\{b_1, \ldots, b_k\}$ of finite elements of D such that every $d \in D$ with $fd \sqsupseteq e$ is above one of the b_i.

Note that the above set is always upper-closed. Thus property (C) is equivalent to saying that the map

$$\delta_f : E \to CU(D)$$
$$\delta_f(e) = \{d \in D \mid fd \sqsupseteq e\}$$

where CU(D) is the set of compact, upper-closed subsets of D, is well defined. Note that

$$\delta_f(e) = f^{-1}(\uparrow\{e\})$$

in terms of the liveness adjunction of Section 4. Once again, we recall a few definitions and facts about powerdomains.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

6.17. Definition

(i) P_S° is the Smyth powerdomain functor (with empty set)

$$P_S^\circ(D) = (CU(D), \supseteq)$$
$$P_S^\circ(f)X = \uparrow\{fx \mid x \in X\}.$$

(ii) For any algebraic lattice L, the greatest lower bound operation, viewed as a function

$$\bigsqcap_L : P_S^\circ(L) \to L$$

is continuous and $\bigsqcap_L(\uparrow X) = \bigsqcap X$.

(iii) For any Scott domain (and more generally, any coherent algebraic cpo [24]) $P_S^\circ(D)$ is closed under finite intersections and unions, and intersections of $\supseteq$-chains.

Now suppose that $f : D \to E$ satisfies property (C). Then $\delta_f : E \to P_S^\circ(D)$ is well defined. Moreover, we have:

6.18. Proposition

If $\delta_f : E \to P_S^\circ(D)$ is well-defined, it is continuous.

Proof

First, $e \sqsubseteq e' \Rightarrow \delta_f(e) \supseteq \delta_f(e')$, so δ_f is monotone. Now

$$\begin{aligned}\delta_f(\bigsqcup e_n) &= \{d \mid fd \supseteq \bigsqcup e_n\} \\ &= \{d \mid \forall n \cdot fd \supseteq e_n\} \\ &= \bigcap_{n=0}^{\infty} \{d \mid fd \supseteq e_n\} \\ &= \bigsqcup_{n=0}^{\infty} \delta_f(e_n). \blacksquare\end{aligned}$$

6.19. Proposition

Suppose that abs_A satisfies (C) for all base types A. Then for all σ, abs_σ^L satisfies (C1.L), (C2.L), and (C).

Proof

By induction on σ. The base case is given. For the inductive step, assume abs_σ^L, abs_τ^L are continuous and satisfy (C). Then $\delta_{abs_\sigma^L}$ is well-defined and continuous, and we have

$$(\text{PDL}) abs_{\sigma\to\tau}^L f = \bigsqcap_L \circ P_S^\circ(abs_\tau^L \circ f) \circ \delta_{abs_\sigma^L}$$

so $abs_{\sigma\to\tau}^L$ satisfies (C1.L) and (C2.L). Moreover, from 6.18 and 6.17 (iii) we

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

see that it is sufficient to verify property (C) on *finite* elements of $D^I_{\sigma\to\tau}$. Accordingly, let $f = \bigsqcup\{[b_1, e_1], \ldots, [b_n, e_n]\}$ be a finite element of $D^I_{\sigma\to\tau}$. For convenience, we shall write δ_σ instead of δ_{abs_σ}, etc. Now

$$\delta_{\sigma\to\tau}(f) = \bigcap\{\delta_{\sigma\to\tau}([b_1, e_1]), \ldots, \delta_{\sigma\to\tau}([b_n, e_n])\}$$

so by 6.17 (iii) it is sufficient to show that $\delta_{\sigma\to\tau}([b, e])$ is Scott-compact for step-functions $[b, e]$. Now let us introduce the function neighbourhood notation

$$(U, V) = \{f \mid f(U) \subseteq V\}$$

for sets U, V. Then we have

$$\begin{aligned}
\delta_{\sigma\to\tau}([b, e]) &= \{f \mid abs^L_{\sigma\to\tau} f \sqsupseteq [b, e]\} \\
&= \{f \mid abs^L_{\sigma\to\tau} fb \sqsupseteq e\} \\
&= \{f \mid \forall d' \cdot abs^L_\sigma d' \sqsupseteq b \Rightarrow (abs^L_\tau \circ f)d' \sqsupseteq e\} \\
&= \{f \mid \forall d' \cdot d' \in \delta_\tau(b) \Rightarrow f(d') \in \delta_\tau(e)\} \\
&= (\delta_\sigma(b), \delta_\tau(e)).
\end{aligned}$$

Now by inductive hypothesis $\delta_\sigma(b)$ and $\delta_\tau(e)$ are Scott-compact, while since b and e are finite, $\delta_\sigma(b) = (abs^L_\sigma)^{-1}(\uparrow\{b\})$ is Scott-open, since by inductive hypothesis again abs^L_σ is continuous, and similarly for $\delta_\tau(e)$. Now Scott-compact-open sets in algebraic domains have the form $\uparrow E$, E a finite set of finite elements. Hence $\delta_\sigma(b) = \uparrow\{b_1, \ldots, b_n\}$, $\delta_\tau(e) = \uparrow\{e_1, \ldots, e_m\}$ for finite elements $b_1, \ldots, b_n, e_1, \ldots, e_m$.

So, $\delta_{\sigma\to\tau}([b, e]) = (\uparrow\{b_1, \ldots, b_n\}, \uparrow\{e_1, \ldots, e_m\})$

$$= \bigcap_{i=1}^{n} \bigcup_{j=1}^{m} \uparrow[b_i, e_j]$$

which is Scott-compact using 6.17 (iii) again. ∎

We now turn to the necessity of (C).

6.20. Proposition

A necessary condition for abs^L to satisfy (C1.L) and (C2.L) is that all the base maps abs_A satisfy (C).

Proof

Suppose some abs_A has a point $d \in D^I_A$ such that $\delta_A(d)$ is not Scott-compact. Note that in this case abs_A cannot be a constant function, so for some finite $e \in D^I_A$, $abs_A(e) \neq abs_A(\bot)$. Now failure of Scott-compactness for $\delta_A(d)$ means there is an infinite set of finite elements $C = \{c_0, c_1, \ldots\}$ such that $\delta_A(d) \subseteq \uparrow C$ and for all finite $F \subseteq C$, $\delta_A(d) \not\subseteq \uparrow F$. We now define a sequence of functions

$$f_n : D^I_A \to D^I_A$$

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

by

$$f_0 = [c_0, e]$$

$$f_{n+1} = f_n \sqcup [c_{n+1}, e]$$

(the lubs being taken here are easily seen to be consistent, so this is well-defined).

Then

$$abs^L_{A\to A} \sqcup f_n d = \bigsqcap \{abs_A(e)\} = abs_A(e)$$

while for all n

$$abs^L_{A\to A} f_n d, \bigsqcap \{abs_A(e) = abs_A(\bot)\} = abs_A(\bot)$$

so $abs^L_{A\to A}$ does not satisfy (C2.L). ∎

We can summarize our results on liveness in two theorems parallel to 6.11 and 6.15.

6.21. Theorem

A necessary and sufficient condition for abs^L to satisfy C1.L and C2.L is that all base maps satisfy (C).

6.22. Theorem

A necessary and sufficient condition for abs^L to satisfy C1.L and C2.L is that it can be defined by (PDL) via the Smyth powerdomain.

We have been at pains throughout our exposition to emphasize the thoroughgoing duality between safety and liveness. We now record an important difference between them. Recall that in Corollary 6.9 we observed that if abs^S is well-defined over the continuous type structure, it is continuous. Thus properties (C1.S) and (C2.S) are in fact not separable. However, for liveness this is *not* the case. As a counter-example, we have:

6.23. Theorem

The $HALT_A$ maps of Section 5 do not satisfy (C) unless D^I_A is finite. However, because each D^I_A is finite, $MD^I_\sigma = D^I_\sigma$ for all σ, and abs^L satisfies (C1.L).

Note that the $HALT_A$ maps trivially satisfy property (F).

6.24. Proposition

If each abs_A has finite image, then for all σ abs^L_σ satisfies (C1.L) and has finite image.

Proof

By induction on σ. The base case is given. For the inductive step, we show:

(i) $abs^L_{\sigma\to\tau} f$ is continuous, for all $f \in D^L_{\sigma\to\tau}$.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

(ii) $abs^L_{\sigma\to\tau}$ has finite image.

(iii) Let $\{d_n\}$ be a chain in D^I_σ. By induction hypothesis, $I = IM\ abs^L_\sigma$ is finite. Let $Y = \{x \in I \mid \exists n \cdot x \not\supseteq d_n\}$. Then Y is finite, and so for some n_0, $\forall k \geq n_0$, $\forall y \in Y y \not\supseteq d_k$. Let $X = I - Y$. Then $\forall k \geq n_0$,

$$abs^L_{\sigma\to\tau} f d_k = \bigsqcap \{(abs^L_\tau \circ f)d' \mid abs^L_\sigma d' \in X\}$$
$$= abs^L_{\sigma\to\tau} f \sqcup dn$$

and so $abs^L_{\sigma\to\tau} f$ is continuous.

(iv) Define an equivalence relation $\sim_\sigma$ on D_σ by

$$x \sim_\sigma y \Leftrightarrow \delta_\sigma(x) = \delta_\sigma(y).$$

Note that

$$x \sim_\sigma y \Rightarrow abs^L_{\sigma\to\tau} fx = abs^L_{\sigma\to\tau} fy$$

for all $f \in D^I_{\sigma\to\tau}$ Now define $Z \subseteq D^I_\tau$ by

$$Z = \{\bigsqcap Y \mid Y \subseteq Im\ abs^L_\tau\}$$

Note that $Im\ abs^L_{\sigma\to\tau} f \subseteq Z$ for all $f \in D^I_{\sigma\to\tau}$ Thus for all $f \in D^I_{\sigma\to\tau}$, $abs^L_{\sigma\to\tau} f$ factors as

$$\begin{array}{ccc} D^I_\sigma & \xrightarrow{abs^L_{\sigma\to\tau} f} & D^I_\sigma \\ {\scriptstyle s}\downarrow & & \uparrow{\scriptstyle i} \\ D^I_\sigma/{\sim_\sigma} & \xrightarrow{\hat{f}} & Z \end{array}$$

where s is the canonical surjection and i the inclusion. This gives a map

$$abs^L_{\sigma\to\tau} f \mapsto \hat{f}$$

which is injective, since s and i are defined uniformly over all f. Thus $card(Im\ abs^L_{\sigma\to\tau}) \leq card(Z^{D^I_\sigma/\sim_\sigma})$.

But $D^I_\sigma/\sim_\sigma$ is finite since $Im\ abs^L_\sigma$ is finite by inductive hypothesis, and

$$\delta_\sigma(d) = \bigcup\{(abs^L_\sigma)^{-1}(\{x\}) \mid x \in Im\ abs^L_\sigma \& x \supseteq d\}$$

and so the size of $Im\ \delta_\sigma$ is bounded by the number of subsets of $Im\ abs^L_\sigma$. Similarly, Z is finite. Thus $Z^{D^I/\sim_\sigma}$ is finite and so $Im\ abs^L_{\sigma\to\tau}$ is. ∎

6.25. Proposition

If abs^L_σ has infinite image on a chain in D^I_σ, and abs^L_τ is not continuous, then $abs^L_{\sigma\to\tau}$ does not satisfy (C1.L).

Proof

We are given $\{b_n\}$ in D^I_σ such that $\{abs^L_\sigma(b_n)\}$ is infinite, and $\{d_n\}$ in D^I_τ such that $abs^L_\tau(\bigsqcup_{n=0}^\infty d_n) \not\sqsubseteq \bigsqcup_{n=0}^\infty abs^L_\tau(d_n)$.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

We define

$$f^0 : K(D'_\sigma) \to D'_\tau$$

by

$$f^0 b = \begin{cases} \bigsqcup d_n \text{ if } \forall n \cdot b \not\sqsubseteq b_n \\ d_n,\ \mu n \cdot b \sqsubseteq b_n, \text{ otherwise.} \end{cases}$$

Claim

f^0 is monotone. Indeed, if $d \sqsubseteq d'$, then either $f^0 d' = \bigsqcup d_n \sqsupseteq f^0 d$ or $f^0 d' = d_n$ which implies $d \sqsubseteq b_k$ for some $k \leq n$ and so $f^0 d = d_k \sqsubseteq d_n$.

Now let

$$f : D'_\sigma \to D'_\tau$$

be the unique continuous extension of f^0

Let

$$c_n = abs^L_\sigma(b_n)$$

$$c = \bigsqcup abs^L_\sigma(b_n).$$

Now $abs^L_\sigma d \sqsupseteq c \Rightarrow fd = \bigsqcup d_n$ since if $\exists m \cdot d \sqsubseteq b_m$, then $abs^L_\sigma d \sqsubseteq c_m \sqsubseteq c \sqsubseteq abs^L_\sigma d$, and so $c_m = c$ contradicting $\{c_n\}$ infinite.

Thus

$$abs^L_{\sigma\to\tau} f \sqcup c_n = \bigsqcap \{(abs^L_\tau \circ f) d' \mid abs^L_\sigma d' \sqsupseteq c\}$$

$$= \bigsqcap \{abs^L_\tau(\bigsqcup d_n)\} = abs^L_\tau(\bigsqcup d_n).$$

For all n, $abs^L_{\sigma\to\tau} f c_n \sqsubseteq (abs^L_\tau \circ f) b_n = abs^L_\tau(d_n)$.

Thus $abs^L_{\sigma\to\tau} f \sqcup c_n = abs^L_\tau(\bigsqcup d_n) \not\sqsubseteq \bigsqcup_{n=0}^{\infty} abs^L_\tau(d_n) \sqsupseteq \bigsqcup_{n=0}^{\infty} abs^L_{\sigma\to\tau} f c_n$ and so $abs^L_{\sigma\to\tau} f$ is not continuous and $abs^L_{\sigma\to\tau}$ does not satisfy (C1.L). ∎

6.26. Proposition

If abs_A has infinite image, and finite image on all chains, then *either*

1. $abs^L_{A\to A}$ has infinite image on a chain in $D'_{A\to A}$, or
2. $abs^L_{A\to A}$ does not satisfy (C1.L).

Proof

Since abs_A has finite image on all chains and is continuous, $Im\, abs_A = Im(abs_A \restriction K(D'_A))$. As a consequence of this, $(abs_A)^{-1}(\uparrow\{d\})$ is open for all $d \in D'_A$. We now consider two cases:

Case 1. For some infinite set $\{d_n\} \subseteq Im\, abs_A$,

$$(**) \forall n \{d_j \mid d_n \sqsubseteq d_j\} \text{ is finite. Let } A_n = (abs_A)^{-1}(\uparrow\{d_n\}).$$

Define $id_n : D'_A \to D'_A$, $n \geq 0$, by

$$id_n(x) = \begin{cases} x,\ x \in A_n \\ \bot,\ x \notin A_n \end{cases}$$

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

id_n is continuous since A_n is open. Now define

$$f_n = \bigsqcup \{id_k \mid k \leq n\}.$$

This is well-defined since $id_n \sqsubseteq id_{D'_A}$ for all n. Now $\{f_n\}$ forms a chain in $D^I_{A\to A}$. For $k \leq n$

$$abs^L_{A\to A} f_n d_k = \bigsqcap abs_A \circ f_n(A_k) = \bigsqcap abs_A(A_k) = d_k.$$

By (**), $\exists m > n$, $\forall j \leq n$, $\forall k \geq m \cdot d_j \not\sqsubseteq d_k$, hence $A_k \notin \bigcup_{j\leq n} A_j$ and

$$abs^L_{A\to A} f_n d_k = \bigsqcap abs_A \circ f_n(A_k) = abs_A(\bot) \neq d_k.$$

Thus $\forall n\, \exists m > n\, \forall k \geq m \cdot abs^L_{A\to A} f_n \neq abs^L_{A\to A} f_k$, and $\{abs^L_{A\to A} f_n\}$ is infinite, establishing (1).

Case 2. For every infinite set $\{d_n\} \subseteq Im\, abs_A$. (††) $\exists n \cdot \{d_j \mid d_n \sqsubseteq d_j\}$ is infinite.

Claim

In this case, there is an infinite chain in $Im\, abs_A$.

Proof of claim. For all n we construct

- a strict chain $S_n = \langle d_0, \ldots d_{n-1}\rangle \subseteq Im\, abs_A$,
- an infinite set Z_n such that d_{n-1} is a lower bound of Z_n and $d_{n-1} \notin Z_n$.

Then $\{d_n\}$ will be the infinite chain in $Im\, abs_A$. Let $Z_0 = \{z^0_0, \ldots, z^0_n, \ldots\} \subseteq Im\, abs_A$ be some infinite sequence. By (††), $\exists k \{z^0_e \mid z^0_k \sqsubseteq z^0_e\}$ is infinite. Let $d_0 = z^0_k$. $Z_1 = \{z^0_e \mid z^0_k \sqsubseteq z^0_e\} - \{z^0_k\}$.

At the (n + 1)-th stage, apply (††) to Z_n similarly to produce d_{n+1}, Z_{n+1}. ∎

Now let $A = (abs^L_A)^{-1}(\uparrow \bigsqcup d_n)$ and take $a \in D^I_A$ with $abs(a) \neq abs_A(\bot)$ and define $f : D^I_A \to D^I_A$ by

$$fx = \begin{cases} a, & x \in A \\ \bot, & x \notin A. \end{cases}$$

Since A is open, f is continuous. Now

$$\forall n \circ abs^L_{A\to A} f d_n = abs_A(\bot)$$

while

$$abs^L_{A\to A} f \sqcup d_n = abs_A(a)$$

and so $abs^L_{A\to A}$ does not satisfy (C1.L), establishing (2). ∎

6.27. Theorem

Suppose some abs_B does not satisfy (C). Then a necessary and sufficient condition for abs^L to satisfy (C1.L) is that all abs_A have finite image.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

Proof

Sufficiency is Proposition 6.24. For necessity, we suppose abs_A has infinite image and consider a number of cases. By the proof of Proposition 6.20, the map $abs_B \to B$ is not continuous.

Case 1 abs_A has infinite image on a chain. In this case we can apply Proposition 6.25 with $\sigma = A$, $\tau = B \to B$.

Case 2 abs_A is finite on all chains. There are the two subcases according to Proposition 6.26.

Subcase 2.1 $abs^L_{A \to A}$ has infinite image on a chain. In this case, apply proposition 6.25 with $\sigma = A \to A$, $\tau = B \to B$.

Subcase 2.2 $abs^L_{A \to A}$ does not satisfy (C1.L) hence abs^L does not. ∎

We can summarize out results as follows:

T1 abs^S satisfies (C1.S) and (C2.S) iff all abs_A satisfy (F). Moreover, if abs^S satisfies (C1.S), it satisfies (C2.S).

T2 abs^L satisfies (C1.L) and (C2.L) iff all abs_A satisfy (C).

T3 Suppose some abs_B does not satisfy (C). Then abs^L satisfies (C1.L) iff all abs_A have finite image.

T4(i) abs^S satisfies (C1.S) and (C2.S) iff it is definable via the Hoare powerdomain.

T4(ii) abs^L satisfies (C1.L) and (C2.L) iff it is definable via the Smyth powerdomain.

We now list some open problems and directions arising from the work of this section:

1. Extend the continuity theory by weakening the assumptions, particularly (A3).
   Conjecture. All the results go through with (A3′). Each D'_A is SFP.
   There is perhaps some mathematical interest in developing a theory of continuous Kan extensions.
2. Characterize the conditions under which the situation of 6.23 arises, i.e. abs^L defined on the continuous type structure but not itself continuous.
   Question. Can this happen if J is an infinite interpretation?
3. Extend the framework to richer type structures (sums, products, recursive types, etc). (Much of this has already been done, at least as regards safety analysis, by Neilson [18].)
4. Generalize the framework, e.g. to arbitrary ccc's.
5. Develop the ideas discussed at 6.6.

We shall conclude with an example of the application of our theory. We return to the analyses of Section 5, and show that the abstract interpreta-

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

tions of the constants are not merely correctly related to the standard interpretations (as shown in 5.3 and 5.4) but best possible.

Let abs^S, abs^L be the safety and liveness abstraction maps induced from the $HALT_A$. (Note that abs^S is exactly the family of maps defined in [1], the definitions used there being an instance of the (PDS) scheme at 6.15 above.)

The following lemma will be useful in our calculations. It is analogous to the formula (*) preceding 5.3.

6.28. Lemma

For $\sigma = \sigma_1 \to \ldots \sigma_n \to \tau$

(i) $abs^S_\sigma f b_1 \ldots b_n = \bigsqcup \{abs^S_\tau(f a_1 \ldots a_n) \mid \bigwedge_{i=1}^n abs^S_{\sigma_i} a_i \sqsubseteq b_i\}$.
(ii) $abs^L_\sigma f b_1 \ldots b_n = \bigsqcap \{abs^L_\tau(f a_1 \ldots a_n) \mid \bigwedge_{i=1}^n abs^L_{\sigma_i} a_i \sqsupseteq b_i\}$.

Now we have:

6.29. Proposition

For each first order constant $c : \sigma$ with $\sigma = A_1 \to \ldots A_n \to B$,

(i) $abs^S_\sigma(c^I) = c^S$,
(ii) $abs^L_\sigma(c^I) = c^T$.

Proof

We show (i) for illustration. Consider $b_1, \ldots, b_n \in 2$. If some $b_j = 0$, then since $HALT_{Ai}$ is $\perp$-reflecting.

$$\bigwedge_i HALT_{Ai} a_i \sqsubseteq b_i \Rightarrow a_j = \perp_{Ai} \Rightarrow c^I a_1 \ldots a_n = \perp_B \text{ since } c^I \text{ is strict}$$

$$\Rightarrow HALT_B(c^I a_1 \ldots a_n) = 0.$$

So $abs^S_\sigma c^I b_1 \ldots b_n = 0$ in this case.

If all $b_j = 1$, then $\exists a_1 \ldots a_n \cdot \bigwedge_i a_i \neq \perp_{Ai}$ and $HALT_{Ai} a i \sqsubseteq b_i$

$$\Rightarrow c^I a_1 \ldots a_n \neq \perp_B \text{ (this uses totality of } c^I)$$

$$\Rightarrow HALT_B(c^I a_1 \ldots a_n) = 1.$$

So $abs^S_\sigma c^I b_1 \ldots b_n = 1$ in this case. Thus $abs^S_\sigma c^I b_1 \ldots b_n = \bigwedge_i b_i = c^S b_1 \ldots b_n$. ∎

6.30. Proposition

For each condition $a^I\, cond : \sigma = bool \to A \to A \to A$:

(i) $abs^S_\sigma(cond^I) = cond^S$,
(ii) $abs^L_\sigma(cond^I) = cond^T$.

Proof

Similar to 6.29. ∎

7. Totality analysis

The termination analysis presented in Section 5 is of rather questionable

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

practical value. Indeed, it is not entirely clear *what* semantic property we are approximating at higher order types; hence our treatment of correctness only dealt with base types.

In this section we consider a more natural, and indeed very canonical notion of *totality*, and use our framework to derive an effective, finite approximation to ti.

Definition

We define a unary logical relation Tot on the standard interpretation D^I by

$$\mathsf{Tot}_A(d) \Leftrightarrow d \neq \bot_{D^I_A}$$
$$\mathsf{Tot}_{\sigma\to\tau}(f) \Leftrightarrow \forall d \in D_\sigma \cdot \mathsf{Tot}_\sigma(d) \Rightarrow \mathsf{Tot}_\tau(fd).$$

This is the property of (hereditary) *totality*; cf. [4], [5].

It is well known that the logical complexity of Tot, even on the computable elements of I, is unbounded in the analytical hierarchy [22]. This makes it somewhat remarkable that *complete* information on Tot can be obtained from our abstraction maps abs^L, and finite abstract domains D^L. (To avoid confusion, note that we are *not* claiming that complete information can be obtained by the abstract interpretation; rather, that taking the *standard* interpretation and then applying the abstraction map—which by this very result is shown to be highly non-computable—does not lose any information.)

We define $\iota_\sigma \in D^L_\sigma$ by induction on σ

$$\iota_A = 1$$
$$\iota_{\sigma\to\tau} = [\iota_\sigma, \iota_\tau].$$

7.1. Proposition

For all σ, $d \in D^I_\sigma$

$$\mathsf{Tot}_\sigma(d) \Leftrightarrow \iota_\sigma \sqsubseteq abs^L_\sigma(d).$$

Proof

By induction on σ. For base types A

$$\mathsf{Tot}_A(d) \Leftrightarrow d \neq \bot_A \Leftrightarrow abs^L_A(d) = 1 \Leftrightarrow \iota_A \sqsubseteq abs^L_A(d).$$

For the inductive step, we first prove that

$$\mathsf{Tot}_{\sigma\to\tau}(f) \Rightarrow \iota_{\sigma\to\tau} \sqsubseteq abs^L_{\sigma\to\tau}(f).$$

$$\begin{aligned}
\mathsf{Tot}_{\sigma\to\tau}(f) &\Rightarrow \forall d \in D_\sigma \cdot \mathsf{Tot}_\sigma(d) \Rightarrow \mathsf{Tot}_\iota(fd) \\
&\Rightarrow \forall d \in d_\sigma \cdot \iota_\sigma \sqsubseteq abs^L_\sigma(d) \Rightarrow \iota_\tau \sqsubseteq abs^L_\tau(fd) \quad \text{(IH)} \\
&\Rightarrow \iota_\tau \sqsubseteq \bigwedge \{abs^L_\tau \circ f(d) \mid \iota_\sigma \sqsubseteq abs^L_\sigma(d)\} \\
&\Rightarrow \iota_\tau \sqsubseteq abs^L_{\sigma\to\tau}(f)(\iota_\sigma) \\
&\Rightarrow \iota_{\sigma\to\tau} \sqsubseteq abs^L_{\sigma\to\tau} f.
\end{aligned}$$

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

For the converse

$$\begin{aligned}
&\iota_{\sigma\to\tau} \sqsubseteq abs^L_{\sigma\to\tau}(f) \& \mathsf{Tot}_\sigma(d) \\
\Rightarrow\ &\iota_\sigma \sqsubseteq abs^L_\sigma(d) \& \iota_{\sigma\to\tau} \sqsubseteq abs^L_{\sigma\to\tau}(f) && \text{(IH)} \\
\Rightarrow\ &\iota_\tau \sqsubseteq abs^L_{\sigma\to\tau}(f) abs^L_\sigma(d) \\
\Rightarrow\ &\iota_\tau \sqsubseteq abs^L_\tau(fd) && \text{Theorem 6.3 (ii)} \\
\Rightarrow\ &\mathsf{Tot}_\tau(fd) && \text{(IH). ∎}
\end{aligned}$$

We can now define *totality analysis*: for any (closed) term $M:\sigma$, we infer that M is total if

$$\iota_\sigma \sqsubseteq E^L[\![M]\!].$$

The following proposition establishes the soundness of this analysis:

7.2. Proposition

$$\iota_\sigma \sqsubseteq E^L[\![M]\!] \Rightarrow \mathsf{Tot}_\sigma(E^I[\![M]\!]).$$

Proof

Combine Proposition 7.1 and Theorem 6.6. ∎

Acknowledgements

The idea of using a formalism based on logical relations for abstract interpretation was inspired by a suggestion of Alan Mycroft's [15]. The theory of abstraction maps developed in Section 6 grew from my joint work with Geoffrey Burn and Chris Hankin on the specific case of strictness analysis [1]. In particular, Geoff deserves credit for having originally proposed the formulas for the abstraction maps. The connection with Kan extensions emerged from one of uncountably many conversations with Axel Poigné on matters categorical.

References

[1] G. L. Burn, C. L. Hankin, and S. Abramsky (1986). Strictness analysis for higher order functions, *Science of Computer Programming*, **7**, 249–78.
[2] P. Cousot and R. Cousot (1977). Abstract Interpretation: A unified lattice model for static analysis of programs by construction of approximation of fixpoints, *4th POPL*, 238–52.
[3] C. D. Clack and S. L. Peyton Jones (1985), Strictness analysis—a practical approach, In *Proceedings IFIP Symposium on Functional Programming Languages and Computer Architecture*. Springer-Verlag.
[4] Yu. L. Ershov (1976). Model C of the Partial Continuous Functionals.
[5] J.-Y. Girard (1986). System F of variable types, fifteen years later, *Theoretical computer Science*, 159–92. North Holland.
[6] C. Gunter and D. Scott (1990). Semantic domains. To appear in *Handbook of Theoretical Computer Science*. North Holland.
[7] P. Hudak and J. Young (1986). Higher-order strictness analysis in untyped lambda calculus, In *13th POPL*.

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014

[8] J. Hughes, (1986). Strictness detection in non-flat domains, In *Proceedings of the DIKU Workshop on Programs as Data Objects*, LNCS 217. Springer-Verlag.
[9] G. M. Kelly and R. M. Street (1974). Review of the elements of 2-categories. In A. Dold and B. Eckmann (eds.), *Category Seminar*, vol. 420 of *Lecture Notes in Mathematics*. Springer-Verlag.
[10] S. Mac Lane (1971). *Categories for the Working Mathematician*. Springer, Berlin.
[11] J. C. Mitchell (1990). Semantics of types. To appear in *Handbook of Theoretical Computer Science*. North Holland.
[12] A Mycroft (1980). The theory and practice of transforming call-by-need into call-by-value. In *Proceedings of the 4th International symposium on Programming*. Springer-Verlag.
[13] A. Mycroft (1981). Abstract Interpretation and Optimising Transformations for Applicative Programs. Ph.D. thesis, University of Edinburgh.
[14] A. Mycroft and F. Nielson (1983). Strong abstract interpretation using power domains, In *ICALP 1983*, LNCS 154. Springer-Verlag.
[15] A. Mycroft (1985). A relational framework for abstract interpretation. Unpublished, handwritten notes.
[16] A. Mycroft and N. D. Jones (1986). A relational framework for abstract interpretation, In *Proceedings of the DIKU Workshop on Programs as Data Objects*, LNCS 217, Springer-Verlag.
[17] F. Nielson (1984). A Denotational Framework for Data Flow Analysis. Ph.D. thesis, University of Edinburgh.
[18] F. Nielson (1988). Strictness analysis and denotational abstract interpretation, *Journal of Information and Computation*, **76** (1), 29–92.
[19] G. Plotkin (1973). Lambda Definability and Logical Relations. Technical Report, Department of AI, University of Edinburgh.
[20] J. P. Seldin and J. R. Hindley (1980). *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press.
[21] G. Plotkin (1981). Lecture Notes on Domain Theory. University of Edinburgh.
[22] D. S. Scott (1976). Data types as Lattices, *SIAM Journal of Computing*, **5** (3), 522–87.
[23] D. S. Scott (1982). Domains for Denotational Semantics, In *ICALP 1982*, LNCS 140. Springer-Verlag.
[24] S. Vickers (1989). *Topology via Logic*. Cambridge Tracts in Theoretical Computer Science 5. Cambridge University Press.
[25] S. Wray (1985). A new strictness detection algorithm, In *Proceedings of the Workshop on Implementation of Functional Languages*. Aspenaes, Sweden.

Received 1 September 1989

Downloaded from http://logcom.oxfordjournals.org/ at Freie Universitaet Berlin on May 15, 2014