PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1988,
edited by ICOT.

TRANSFORMATION OF STRICTNESS-RELATED ANALYSES BASED ON ABSTRACT INTERPRETATION

Mizuhito OGAWA and Satoshi ONO
NTT Software Laboratories
3-9-11 Midori-cho, Musashino-shi, Tokyo 180 Japan
CS-net : mizuhito@sonami-2.ntt.jp, ono@sonami-2.ntt.jp

Abstract

A new formalization method for strictness-related analyses on first-order applicative languages is proposed. For this purpose, the concept *HOMomorphic Transformer* (HOMT) is introduced. Intuitively speaking, a HOMT is a special instance of abstract interpretation. A set of HOMTs, furthermore, is an algebraic space, where equivalence relations (or reduction rules) are defined. This paper clarifies that HOMTs can be used not only as a formalization tool for possibly non-monotonic strictness-related analyses, but also as a transformational mechanism between these analyses. Thus, equivalent and hierarchical relationships among these analyses can be discussed on a unified basis.

First, we show that an effective subset of strictness-related analyses can be formalized as HOMTs. Next, it is clarified that forward / backward conversion operations can also be formalized as HOMTs named *isomorphic converters*. Finally, it is shown that transformational operations that induce weaker GDAs from a GDA, can also be formalized as HOMTs named *projective inducers*. Thus, the equivalence of GDAs can be proved by the equivalence of HOMTs in the proposed algebra, and hierarchical relationships among GDAs can be shown by the existence of projective inducers.

1 Introduction

Stimulated by an urgent need for efficient implementation of lazy applicative languages, many *Global Dataflow Analyses* (GDAs) have been proposed [1,11,17]. Most notable GDAs are a class of *Strictness-Related GDAs (SRAs)*, that collect information on the strictness of functions. SRAs [13] are classified into either *Strictness Analysis* (SA) [2,3,7,10,19], *Relevance Analysis* (RA) or *Computation Path Analysis* (CPA) [14,15,16]. An SA detects a set of parameters that should be evaluated to obtain the resulting value of a function. Conversely, an RA detects a set of parameters that *may* be evaluated. CPA is a generalization of both SA and RA, detecting a set of all possible computation paths.

Abstract interpretation [1,4,9,10] has been proposed as the basis for formalizing GDAs. In addition to continuous GDAs (e.g. SA), non-monotonic GDAs (e.g. RA and CPA) can be uniformly formalized by extending the framework of abstract interpretation [12]. The following questions, however, remain to be solved:

> Some analyses can be formalized as *forward* as well as *backward* [5,7,8]. How can the equivalence between GDAs having different formalizations be shown? (Equivalence problem)
>
> From the viewpoint of analytical power, there exist hierarchical relationships among SRAs. That is, one analysis has more analytical power than others, and the results of a weaker analysis can be induced from those of a stronger one. How can such hierarchical relationships be shown? (Hierarchy problem.)

These problems can not be answered by only providing a uniform formalization method. Apparently, we require some transformational mechanisms that can induce equivalent or weaker GDAs from a GDA.

This paper proposes a new formalization method for SRAs, named *HOMomorphic Transformer* (HOMT). A HOMT is a functional that maps a function on original domains to a function on abstract domains. At this point, HOMTs can be viewed as a subclass of abstract interpretation. A set of HOMTs, however, is also an algebraic space, that has equivalence relations (or reduction rules) on integers, such as $1+2 \rightarrow 3$. A HOMT is represented by a composition of *Unit HOMTs* (U-HOMTs), each of which, in turn, is specified by a newly proposed *Quadruplet Representation*(QR). Reduction rules on HOMTs are clarified in terms of QRs.

We solve the above problems as follows: First, we show that an effective subset of SRAs can be formalized as HOMTs. Next, it is clarified that forward / backward conversion operations can also be formalized as HOMTs named *isomorphic converters*. Thirdly, it is shown that

transformational operations that induce weaker SRAs from an SRA, can also be formalized as HOMTs named *projective inducers*. Thus, the equivalence of SRAs can be proved by the equivalence of HOMTs in the proposed algebra, and hierarchical relationships among SRAs can be shown by the existence of projective inducers.

2 Intuitive Comparison among SRAs

2.1 SRAs as HOMTs

SRAs are made up of three kinds of GDAs. That is, SAs, RAs, and CPAs [13]. CPA computes the Property Dependency Parameter Set (PDPS), which is intuitively a set of all possible demand patterns of the function when demands are propagated to resulting value. RA detects relevant parameters which may need to be evaluated when demands propagate. SA detects requisite parameters which always need to be evaluated when demands propagate.

On the other hand, a HOMT, h_f, is a functional which maps continuous functions f on computational domains to $h_f(f)$ on (possibly finite) abstract domains (See Section 3.1). Thus, SRAs are formalized by HOMTs as follows. Assume h_f be a HOMT such that $h_f(f)$ preserves the objective property of an SRA on the original program f. Note that $h_f(f)$ may be not computable even if abstract domains are finite, since $h_f(f)$ reflects the exact run-time property which is never clarified before actual execution. Thus, the algorithms of SRAs are formalized according to the following two steps. First, compute the approximation $h_{cf}(f)$ of $h_f(f)$, where $h_{cf}(f)$ is the solution of some recursive equation on abstract domains. This result is called the *computed HOMT* (See Section 3.3). Next, execute $h_{cf}(f)$ for all possible instances on abstract domains. As a result, the approximated property on f is detected, instead of the exact but not computable run-time property on f.

With the formalizations of SRAs as above, the equivalence of two SRAs which have different corresponding HOMTS, $h_{f\,1}$, $h_{f\,2}$, is clarified by finding HOMTs, h_{12}, h_{21}, that transform to each other as

$$h_{f\,1}(f) \stackrel{h_{12}}{\rightarrow} h_{f\,2}(f) \quad \text{and,}$$
$$h_{f\,2}(f) \stackrel{h_{21}}{\rightarrow} h_{f\,1}(f)$$

for each continuous function f. Similarly, the hierarchical relationship between two SRAs is clarified by finding a HOMT that transforms the stronger one into the weaker one.

Section 2.2 and 2.3 examine both the equivalence problem and the hierarchy problem of the already proposed algorithms of SRAs.

2.2 Equivalent problems on SRAs

An SRA may be either forward or backward. An SRA is said to be a forward analysis, if it clarifies the properties of results from the properties of parameters. Conversely, an SRA is said to be a backward analysis, if it clarifies the conditions satisfied by parameters from the properties of results.

Forward SA (FSA) is an example of SA as a forward analysis [10]. Similar algorithms are found in [1,3].

FSA interprets a function, f, on a flat domain (such as *Integer* or, *Boolean*) to a {0,1}-valued function f'_{FSA}, where 0 means totally undefined and 1 means possibly defined. Thus, f'_{FSA} returns 1 if there possibly exists a computable *real* instance of f, and returns 0 if there never exists a computable *real* instance of f. For instance, $if(x,y,z)$ is interpreted to

$$\begin{array}{lllll} if'_{FSA}: & (1,1,1) \rightarrow 1, & (1,1,0) \rightarrow 1, \\ & (1,0,1) \rightarrow 1, & (0,1,1) \rightarrow 0, \\ & (1,0,0) \rightarrow 0, & \text{etc.} \end{array}$$

Then, requisite parameters can be detected by firstly testing f'_{FSA} for all {0,1}-input patterns, next collecting the set of minimum input patterns that returns 1 (called 1-*frontier* in[1]), and finally detecting requisite parameters that are always required to be 1 in all patterns in the 1-frontier. For instance, $if(x,y,z)$, the 1-frontier is {(1,1,0),(1,0,1)}, and then, the requisite parameters are $\{x\}$.

On the other hand, Boolean-algebraic SA (BSA) [7,13] is an example of SA as a backward analysis. BSA interprets a function, f, to a function f'_{BSA} which is a symbolic manipulation on the set-characteristic expressions of input parameters. For example, $if(x,y,z)$ is interpreted to

$$if'_{BSA}(x',y',z') \;=\; \lambda x'y'z'.(x' \cup y') \cap (x' \cup z')$$

Then, requisite parameters are collected by substituting actual variable names to corresponding set-characteristic expressions. For instance, requisite parameters of $if(a,b,b)$ are calculated as

$$\begin{array}{ll} & (\ \lambda x'y'z'.(x' \cup y') \cap (x' \cup z')\ )(\{a\},\{b\},\{b\}) \\ = & (\{a\} \cup \{b\}) \cap (\{a\} \cup \{b\}) = \{a,b\} \end{array}$$

In both FSA and BSA, these interpretations for user defined functions are induced by ordinary fixpoint calculus based on given interpretations on primitive functions on abstract domains.

From an application view point, the analytical powers of FSA and BSA are equivalent except that FSA can detect diverged functions when its 1-frontier is an empty set, whereas BSA cannot.

2.3 Hierarchy problems on SRAs

The hierarchy of SRAs arises from 2 reasons :

> Objective property of program is the same for SRA_1 and SRA_2, but abstraction is more accurate on SRA_2 than SRA_1. (Approximation hierarchy)
> Objective property of program itself is more informative on SRA_2 than SRA_1. (Property hierarchy)

The approximation hierarchy arises from the fact that an SRA is a compile-time technique whereas the objective property is a run-time property. Thus, approximation accuracy is traded off with computational complexity (including termination). Therefore, even for the same objective property, there are many selections for approximation levels. These levels can be measured by domain abstractions.

The notable examples are SAs on non-flat domains (eg. lazy list structures, streams). [1,3,10,8,6]. The notable feature of non-flat domains is *non-strictness*. That is, lazy functions such as $cons\text{-}stream(x,y), head(x), tail(x)$ (as in Scheme) admit partially evaluated data for both inputs and outputs.

A trivial extension TSA1 (resp. TSA2) of FSA interprets functions to {0,1}-valued functions where the data structure is approximated as 1 if completely evaluated (resp. possibly evaluated), and as 0 if not completely evaluated (resp. never evaluated).

Conversely, SA on streams [1,6,8,19] called NSA interprets functions as {0,1,2,3}-valued functions. In the abstract domain {0,1,2,3}, 0 means never evaluated, 1 means values that are evaluated at least outermost *cons*, 2 means values that are evaluated until the length of list is clarified, and 3 means completely evaluated values.

Thus, for instance, a parameter x in $length(x)$ is analyzed as not requisite by TSA1 and as possibly requisite by TSA2, whereas it is analyzed as requisite at level 2 by NSA.

The property hierarchy is found in the relation among CPA, SA, and RA. [13]. For example,

$$
\begin{array}{rl}
f(x,y,z,p,q) = & \\
\quad if\ p>0\ then & (if\ p=1 \\
 & \quad then\ (if\ z=0\ then\ x\ else\ y) \\
 & \quad else\ f(z,z,0,p-1,x)) \\
\quad else & f(0,0,z,1,y)
\end{array}
$$

is analyzed as

$$
\left\{
\begin{array}{cc}
\text{PDPS} & \{\{x,z,p\},\{y,z,p\},\{z,p\},\{p\}\} \\
\text{relevant parameters} & \{x,y,z,p\} \\
\text{requisite parameters} & \{p\}
\end{array}
\right.
$$

Roughly speaking, the union of all elements in PDPS is a set of relevant parameters, and the intersection is a set of requisite parameters. Therefore, SA and RA are the projections of CPA. This fact arises from PDPS itself being more informative than requisite parameters and relevant parameters.

3 Algebraic Structure on HOMTs

3.1 Construction of U-HOMTs and their Quadruplet Representations

A HOMT, h_f, is defined to be a functional which maps continuous functions on computational domains to those on abstract domains. A HOMT is constructed as a composition of U-HOMTs. A U-HOMT, h_f, is a functional from continuous functions on *power domains* [1] $PD[D]$ to those on *power domains* $PD[Abs]$, where h_f is induced from the domain abstraction $abs : D \to Abs$ as below.

[Definition : domain abstraction]
A continuous, onto map $abs : D \to Abs$ is said to be a *domain abstraction* iff

$$
\left\{
\begin{array}{ll}
\forall x \in abs^{-1}(a_1), & \exists y \in abs^{-1}(a_2) \quad \text{s.t. } x \sqsubseteq_D y \\
\forall y \in abs^{-1}(a_2), & \exists x \in abs^{-1}(a_1) \quad \text{s.t. } x \sqsubseteq_D y
\end{array}
\right.
$$

for $\forall a_1, a_2 \in Abs$ s.t. $a_1 \sqsubseteq_{Abs} a_2$, is satisfied. The domain Abs is said to be an abstract domain. If Abs is a finite domain, $abs : D \to Abs$ is said to be a finite domain abstraction.

Since, U-HOMTs are functionals on functions on $PD[D]$ to those on $PD[Abs]$, at first a *lifting interpretation* is required. This interprets functions on D to functions on $PD[D]$. This interpretation is naturally induced from the commutative diagram shown in Fig.1. We will use the lifting interpretation as a default. Thus, we do not denote it explicitly.

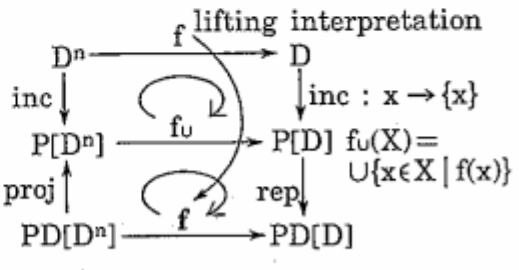

Fig. 1 Lifting interpretation.

[1] A power domain $PD[D]$ is a power set $P[D] = \{X \subset D | X \neq \phi\}$ with definedness ordering $\sqsubseteq$ on it [18] .

Table 1. Typical selections of parameters of QRs.

$$\left\{\begin{array}{l}\underline{\text{domain abstraction}}\\ \text{base domain abstraction}\\ abs_b : x \rightarrow \left\{\begin{array}{ll}1 & (\text{if } x \equiv \bot)\\ 0 & (\text{if } x \not\equiv \bot)\end{array}\right.\\ \qquad (\text{where } \mathbf{0} \sqsubseteq \mathbf{1}.)\\ \text{identical abstraction}\\ abs_I : x \rightarrow x \quad (\forall x \in D)\end{array}\right\} \times \left\{\begin{array}{l}\underline{\text{direction}}\\ \text{covariant}\\ (+)\\ \\ \text{contravariant}\\ (-)\end{array}\right\} \times \left\{\begin{array}{lll}\underline{\text{power domain construction}}\\ \text{preorder} & \text{closure rep.} & \text{invariant rep.}\\ \subseteq & id & id\\ \sqsubseteq_{EM} & Conv & Conv\\ \sqsubseteq_0 & RC & Min\\ \sqsubseteq_1 & LC & Max\end{array}\right\}$$

where,

$$\underline{\text{preorder}} \qquad \left\{\begin{array}{l}\underline{\text{closure rep.}}\ (\subseteq\text{-maximal rep.})\\ \underline{\text{invariant rep.}}\ (\subseteq\text{-minimal rep.})\end{array}\right.$$

$$X \sqsubseteq_{EM} Y \quad \text{iff} \quad X \sqsubseteq_0 Y \wedge X \sqsubseteq_1 Y$$

$$X \sqsubseteq_0 Y \quad \text{iff} \quad RC(X) \supseteq RC(Y)$$

$$X \sqsubseteq_1 Y \quad \text{iff} \quad LC(X) \subseteq LC(Y)$$

$$X \sqsubseteq_{-} Y \quad \text{iff} \quad X \sqsupseteq Y \quad \text{for } \sqsubseteq \text{ as above}$$

$$Conv(X) \stackrel{\text{def}}{=} LC(X) \cap RC(X)$$

$$\left\{\begin{array}{ll}RC(X) & \stackrel{\text{def}}{=} \{x \in D \mid \exists y \in X \ s.t.\ y \sqsubseteq x\}\\ Min(X) & \stackrel{\text{def}}{=} \{x \in X \mid \neg\exists y \in X \ s.t.\ y \sqsubset x\}\\ LC(X) & \stackrel{\text{def}}{=} \{x \in D \mid \exists y \in X \ s.t.\ x \sqsubseteq y\}\\ Max(X) & \stackrel{\text{def}}{=} \{x \in X \mid \neg\exists y \in X \ s.t.\ x \sqsubset y\}\end{array}\right.$$

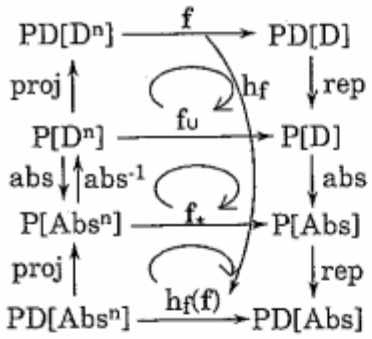

Fig. 2 Construction of a covariant U-HOMT h_f.

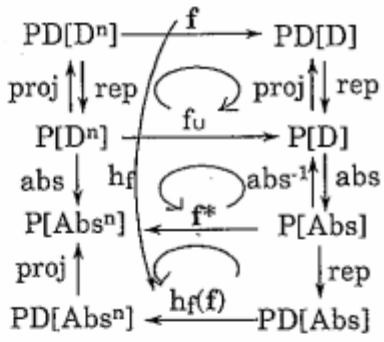

Fig. 3 Construction of a contravariant U-HOMT h_f.

A U-HOMT h_f is then naturally induced from domain abstraction *abs* as in the commutative diagrams shown in Fig.2 and Fig.3.

Note that there are two types of U-HOMTs, *covariant U-HOMTs* and *contravariant U-HOMTs*, corresponding to *forward analyses* and *backward analyses*. There are two differences between them.

For covariant U-HOMTs, $h_f(f)$ is induced from a function, f, itself. Conversely, for contravariant U-HOMTs, $h_f(f)$ is induced from a *function inverse* f^{-1}.

Therefore, contravariant U-HOMTs require an *appendiculate power domain* $PD[D]^+ \stackrel{\text{def}}{=} PD[D] \cup \{\phi\}$, whereas covariant U-HOMTs require an ordinary power domain $PD[D]$ $(= \{X | X \in D, X \neq \phi\}$ with preorder $\sqsubseteq$) (The function inverse may require ϕ as a result value.)

The parameters which specify these constructions are *direction* (that is, whether covariant or contravariant), *domain abstraction*, and *power domain construction*. Power domain construction is composed of the selection of preorder $\sqsubseteq$ on power set and its representative function *rep*.

[Definition : Quadruplet representation.]

A U-HOMT, h_f, is specified by a *quadruplet* $q = (abs, dir, \sqsubseteq, rep)$ which is a pair of *domain abstraction* $abs: D \rightarrow Abs$, *direction* dir (whether covariant $(+)$ or contravariant $(-)$), *preorder* $\sqsubseteq$, and *representative function* rep. A *quadruplet* $q = (abs, dir, \sqsubseteq, rep)$ is called the *quadruplet representation (QR)* of a U-HOMT, h_f.

Table 1 presents typical selections for these parameters (QRs), although not all selections induce useful

SRAs.

Remark Appendiculate power domains $PD^+[D] = PD[D] \cup \{\phi\}$ are constructed from the extension of orderings $X \sqsubseteq_0 \phi$, $\phi \sqsubseteq_1 X$, $\phi \subseteq X$, $\forall X \in PD[D]$. Attention should be paid on that there is no well-defined extension of $\sqsubseteq_{EM}$ on $PD^+[D]$.

[Examples: FSA, Strictness Information Analysis (SIA)]

FSA as a HOMT has $\mathrm{QR}_{FSA} = (abs_b, +, \sqsubseteq_1, Max)$.

Strictness Information Analyses (SIA) [13], which is the variation of backward SA, as a HOMT has $\mathrm{QR}_{SIA} = (abs_b, -, \sqsubseteq_{-0}, Min)$. (Detailed discussion on these formalization will be found in Section.4.1.)

3.2 HOMTs as a composition of U-HOMTs

On the composition $h'_f \circ h_f$ of U-HOMTs h_f and h'_f, the problem is the difference of power domain construction between $range(h_f)$ and $domain(h'_f)$, although they have same base domain. That is, definedness (pre)order may differ within the same power set. Therefore, some special techniques, such as *torsional composition*, is required. This method first embeds the power domain $range(h_f)$ to the power set, and then projects the power set to the power domain $domain(h'_f)$.

[Definition: Composition of U-HOMTs]

The composition $h'_f \circ h_f$ of U-HOMTs h_f and h'_f is defined to be

$$h'_f \circ h_f : f \rightarrow h'_f(proj' \circ rep \circ h_f(f) \circ proj \circ rep')$$

for $\forall f$: continuous function on D. (Fig.4.)

The composition of U-HOMTs is said to be a *covariant HOMT* if the product of all directions is $+$, and a *contravariant HOMT* if the product is $-$.

[Example: CPA]

CPA as a HOMT has

$$\mathrm{QR}_{CPA} = (abs_b, +, \subseteq, id) \circ (id, -, \sqsubseteq_{-0}, Min)$$ [12].

To clarify the hierarchy of analytical power among SRAs, *analytical order* shown below is introduced, This analytical order naturally induces extensional equality among SRAs. This equality is called *algebraic structure* on SRAs.

[Definition: Analytical order on HOMTs]

$$HOMT_1 \preceq HOMT_2 \iff \exists HOMT \text{ s.t. } HOMT_1 = HOMT \circ HOMT_2$$

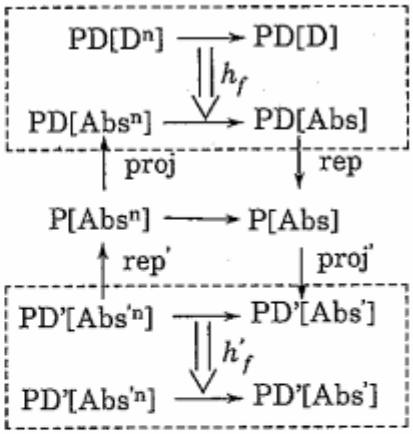

Fig. 4 (Torsional) Composition of U-HOMTs.

[Definition: Analytical equivalence on HOMTs, algebraic structure on HOMTs]

$$HOMT_1 \simeq HOMT_2 \iff HOMT_1 \preceq HOMT_2 \wedge HOMT_2 \preceq HOMT_1$$

$\simeq$ on HOMTs are said to be *algebraic structure* on HOMTs.

In most cases, analytical equivalence between SRAs is easily found from the following *reduction theorem* on QRs of SRAs.

[Reduction Theorem]

Let h_f and h'_f be U-HOMTs, and $q = (abs, dir, \sqsubseteq, rep)$ and $q' = (abs', dir', \sqsubseteq', rep')$ be their QRs, respectively. Then, the composition $h'_f \circ h_f$ is analytically equivalent to a U-HOMT, that is, the composition $q' \circ q$ can be reduced to

$$(abs', dir', \sqsubseteq', rep') \circ (abs, dir, \sqsubseteq, rep) = (abs' \circ abs, dir' \circ dir, \sqsubseteq', rep')$$

if one of following conditions are satisfied.

[1] For $dir' = +$,

(a) $\sqsubseteq' \ll \sqsubseteq$ or,
(b) $\sqsubseteq' \ll \sqsubseteq_-$
where $\sqsubseteq' \ll \sqsubseteq$ iff $X \sqsubseteq Y \Rightarrow X \sqsubseteq' Y$ (Fig.5).

[2] For $dir' = -$,

$$\begin{cases} \text{(a)} & (\sqsubseteq, \sqsubseteq') = (\sqsubseteq_{\pm 0}, \sqsubseteq_{\pm 1}) \wedge rep = Min \\ \text{(b)} & (\sqsubseteq, \sqsubseteq') = (\sqsubseteq_{\pm 1}, \sqsubseteq_{\pm 0}) \wedge rep = Max \\ \text{(c)} & (\sqsubseteq, \sqsubseteq') = (\subseteq, \subseteq) \end{cases}$$

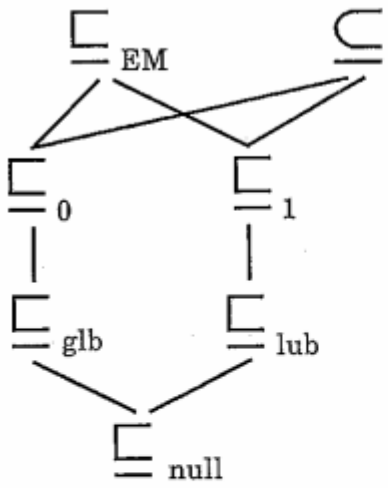

Fig.5 Relations among preorders.

3.3 Continuous HOMTs

There is a useful subclass called *continuous HOMT*. The validity of this class is confirmed to include SAs.

[Definition: continuous HOMT.]
Let $h = (h_\tau, h_f)$ be a HOMT. h is said to be *continuous* iff h_f is continuous, that is, $h_f(lub\{f^{(n)}\}) = lub\{h_f(f^{(n)})\}$ for arbitrary an ascending chain of continuous functions $f^{(0)} \sqsubseteq f^{(1)} \sqsubseteq f^{(2)} \sqsubseteq \cdots$

The following theorem is the sufficient condition for continuity of HOMTs.

[**Continuity Theorem**]
A HOMT, h_f, is continuous if its QR $(abs, dir, \sqsubseteq, rep)$ satisfies one of the following conditions.

$$dir = + \text{ and } \sqsubseteq \in \{\sqsubseteq_0, \sqsubseteq_1, \sqsubseteq_{EM}\}.$$
$$\text{or, } dir = - \text{ and } \sqsubseteq \in \{\sqsubseteq_{-0}, \sqsubseteq_{-1}\}.$$

Remark 1 Continuity theorem gives the condition for continuity on U-HOMTs. Conversely, all our known HOMTs are non-monotonic if their QRs are irreducible, such as CPA and RA.

In general, $h_f(f)$ that reflects a run-time property of f is not computable, even if the abstract domain Abs is finite. Therefore, instead of $h_f(f)$, we introduce $h_{cf}(f)$ which approximates $h_f(f)$, and is computable if the abstract domain is finite.

[Definition: Computed HOMT]
A *computed HOMT* h_{cf} is defined to be

$$h_{cf}(f) \stackrel{\text{def}}{=} fix(h_\tau(\tau))$$

for $\forall f = fix(\tau)$, $\forall \tau$: recursion equation, where $h_\tau(\tau)$ is defined to be a syntactically indentical (resp. inverse) equation, but replaces each primitive function $priv$ with $h_f(priv)$, if h_f is a covariant (resp. contravariant) HOMT.

Remark 2 h_{cf} obviously satisfies the homomorphic condition $h_{cf}(f \circ g) = h_{cf}(f) \circ h_{cf}(g)$. This is why it is called a HOMomorphic Transformer.

The relation which justifies that a computed HOMT h_{cf} *properly* approximates a HOMT h_f is called safeness.

[Definition: Safeness]
A HOMT, h_f, is said to be *safe* iff $h_f(f) \sqsubseteq h_{cf}(f)$ for all continuous functions f.

Remark 3. Safeness $h_f \sqsubseteq h_{cf}$, is easily proved if h_f is continuous, $h_f(\Omega) \equiv \Omega'$, and $h_f(priv_1^* \circ priv_2^*) \sqsubseteq h_f(priv_1^*) \circ h_f(priv_2^*)$ for an arbitrary composition of primitive functions $priv_1^*, priv_2^*$. Thus, HOMTs which are enumerated in continuity theorem satisfy safeness.

Remark 4. Though CPA is a non-monotonic SRA, CPA has been proved to be safe, independently from the framework of HOMTs [14].

3.4 Non-monotonic HOMTs

The reason why a HOMT is composed as the composition of U-HOMTs instead of a U-HOMT itself, is non-continuity of some SRAs, such as CPA and RA.

The non-continuity of SRAs arises from two reasons.

- An approximating chain $h_f(f^{(i)})$ $(i = 0, 1, 2, \cdots)$ is non-monotonic.
- A HOMT h_f and $\sqcup$ (*lub-operater*) are not commutative. (i.e. $h_f(f) \equiv h_f(\sqcup f^{(i)}) \not\equiv \sqcup h_f(f^{(i)})$)

For instance, the following two examples for CPA correspond to the conditions above. The first example is

$$foo_1(x,y) = por_3(x, y, foo_1(and_2(not(x), not(y)), foo_1(pand_2(x,y), pand_2(x,y)))$$

where por_3, and_2, and $pand_2$ are *parallel-or*, *strict-and*, and *parallel-and*, respectively.

Then, an approximating chain $h_f(foo_1^{(i)})$ (that is, real PDPS of $foo_1^{(i)}$ for $i = 0, 1, 2, \cdots$) is non-monotonic as

$h_f(\Omega)$	$h_f(foo_1^{(1)})$	$h_f(foo_1^{(2)})$	$h_f(foo_1^{(3)})$
ϕ	$\{\{x\}\{y\}\}$	$\{\{x\}\{y\}\{x,y\}\}$	$\{\{x\}\{y\}\}$

whereas an approximating chain $h_{cf}(foo_1^{(i)})$ (that is, computed PDPS of $foo_1^{(i)}$ for $i = 0, 1, 2, \cdots$) is monotonic as

$h_{cf}(\Omega)$	$h_{cf}(foo_1^{(1)})$	$h_{cf}(foo_1^{(2)})$	$h_{cf}(foo_1^{(3)})$
ϕ	$\{\{x\}\{y\}\}$	$\{\{x\}\{y\}\{x,y\}\}$	$\{\{x\}\{y\}\{x,y\}\}$

The second example is

$$\begin{aligned} foo_2(x,y) &= por_2(g(x,y), gate(true, x+y)) \\ g(x,y) &= if\ x=0\ then\ true\ else\ g(x-1,y) \end{aligned}$$

Then,

$$h_f(foo_2^{(i)}) = \{\{x\}(when\ x \le i), \{x,y\}(otherwise)\}$$

and consequently, $\sqcup\{h_f(foo_2^{(i)})\} = \{\{x\},\{x,y\}\}$, whereas $h_f(\sqcup\{foo_2^{(i)}\}) = \{\{x\}\}$ (real PDPS).

Conversely, $h_{cf}(foo_2^{(i)}) = \{\{x\},\{x,y\}\}$ and consequently, $\sqcup\{h_{cf}(foo_2^{(i)})\} = \{\{x\},\{x,y\}\}$ (computed PDPS).

4 Isomorphic Conversion and Projective Induction among SRAs

4.1 SRAs as HOMTs, revisited

This section induces two special kind of HOMTs as transformation methods among SRAs in order to solve both the equivalence problem and the hierarchy problem. They are called the *isomorphic converter* and the *projective inducer*. An isomorphic converter transforms a forward (resp. backward) SRA into an analytically equivalent backward (resp. forward) SRA. On the other hand, a projective inducer transforms a forward (resp. backward) SRA to a weaker forward (resp. backward) SRA.

Thus, the equivalence problem is clarified as the existence of an isomorphic converter. Similarly, the hierarchy problem is clarified as the existence of a projective inducer.

For this purpose, SRAs are redefined in terms of HOMTs.

[Definition: base domain abstraction.]

The base domain abstraction $abs_b: D \to Abs$ is extended inductively according to structure of domain D.

$$abs_B: x \to \begin{cases} constructor(abs_B(y), abs_B(z)) \\ (\ x = constructor(y,z)\ ) \\ abs_b(x) \\ (\ \text{otherwise}\ ) \end{cases}$$

Intuitively speaking, base domain abstraction concerns whether the value is defined or not, but ignores the value itself.

[Definition: SRA]

Let h_f be a HOMT, and $abs: D \to Abs$ be a domain abstraction. Then, h_f is said to be an *SRA* iff there exists abs_S s.t. $abs = abs_S \circ abs_B$. (abs_S is said to be *structural domain abstraction.*)

4.2 Isomorphic conversion between forward/backward SRAs

[Definition: Isomorphic converter.]

Let $h_{1\,f}$, $h_{2\,f}$ be HOMTs. Contravariant HOMTs h_f and h'_f are said to be *isomorphic converter* iff $h_{1\,f} \approx h_f \circ h_{2\,f}$ and $h_{2\,f} \approx h'_f \circ h_{1\,f}$ are satisfied.

Let us consider an example of isomorphic conversion between FSA (forward SA) and SIA (backward SA) which is a natural extension of BSA. First, a QR of FSA is presented. Next, a QR of SIA is presented. Finally, isomorphic conversion is given from reduction theorem.

A QR of FSA is $q_{FSA} = (abs_b, +, \sqsubseteq_1, Max)$. This is shown from the correspondence among interpretations on primitive functions and definedness ordering on abstract domains.

Let $h_{FSA\,f}$ be a HOMT whose QR is q_{FSA}. Equivalence among interpretations on primitive functions is checked by testing all possible values on abstract domains. For example, by FSA, $if(x,y,z)$ are interpreted to

$$\begin{aligned} if'_{FSA}: \ & (1,1,1) \to 1, \quad (1,1,0) \to 1, \\ & (1,0,1) \to 1, \quad (0,1,1) \to 0, \\ & (1,0,0) \to 0, \quad \text{etc.} \end{aligned}$$

and by h_{FSA},

$$\begin{aligned} h_{FSA\,f}(if): \ & Max(\{(1,1,1)\}) \to Max(\{1\}), \\ & Max(\{(1,1,0)\}) \to Max(\{1\}), \\ & Max(\{(1,0,1)\}) \to Max(\{1\}), \\ & Max(\{(0,1,1)\}) \to Max(\{0\}), \\ & Max(\{(1,0,0)\}) \to Max(\{0\}), \\ & \text{etc.} \end{aligned}$$

Thus, equivalence among if'_{FSA} and $h_{FSA\,f}(if)$ is easily shown from an embedding : $x \in \{0,1\} \to \{x\} \in PD(\{0,1\})$. And, equivalence of definedness order is obvious from $x \sqsubseteq y \Rightarrow \{x\} \sqsubseteq_1 \{y\}$.

SIA, the extension of BSA, is a HOMT $h_{SIA\,f}$ whose QR is $q_{SIA} = (abs_b, -, \sqsubseteq_{-0}, Min)$. The main difference between BSA and SIA is that SIA can detect diverged functions whereas BSA cannot. This is because a totally undefined function $\Omega(x_1,\ldots,x_n)$ is interpreted to $\lambda x'_1 \ldots x'_n.UNDEF$ by SIA with a special value $UNDEF$, whereas $\Omega(x_1,\ldots,x_n)$ is simply interpreted to $\lambda x'_1 \ldots x'_n.x'_1 \cup \ldots \cup x'_n$ (strict function) by BSA. Except $\Omega(x_1,\ldots,x_n)$, other functions are interpreted to the same abstract functions on both BSA and SIA. For

example, primitive functions $if(x,y,z)$ and $+(x,y)$ are interpreted to

$$h_{BSA\ f}(if) = \lambda x'y'z'.(x' \cup y') \cap (x' \cup z')$$
$$h_{SIA\ f}(if) : \{1\} \to Min(\{(1,1,1),(1,1,0),(1,0,1)\})$$

Thus, equivalence except $\Omega(x_1,\ldots,x_n)$ is easily shown under one-to-one mapping such as

$$\begin{array}{lcl} (x' \cup y') \cap (x' \cup z') & \leftrightarrow & \{(1,1,0),(1,0,1)\} \\ x' \cup y' & \leftrightarrow & \{(1,1)\} \\ UNDEF & \leftrightarrow & \phi \end{array}$$

Correspondence among definedness orderings is also easily checked from the fact $X \subseteq Y \Rightarrow X \sqsubseteq_0 Y$.

Remark. FSA and SIA are continuous SRAs from continuity theorem (See Section 3.3).

[Example: Equivalence of FSA and SIA.]

The equivalence of FSA and SIA is proved from reduction theorem (See Section 3.2). That is,

$$\begin{array}{ll} & \underline{(id,-,\sqsubseteq_{-0},Min)} \circ (abs_b,+,\sqsubseteq_1,Max) \\ = & (abs_b,-,\sqsubseteq_{-0},Min) \end{array}$$

$$\begin{array}{ll} & \underline{(id,-,\sqsubseteq_1,Max)} \circ (abs_b,-,\sqsubseteq_{-0},Min) \\ = & (abs_b,+,\sqsubseteq_1,Max) \end{array}$$

Thus, equivalence between FSA and SIA is clarified as the existence of underlined QRs which are isomorphic converters.

4.3 Projective induction among SRAs

[Definition : Projective inducer.]

Let $h_{1\ f}, h_{2\ f}$ be HOMTs. Covariant HOMTs h_f is said to be a *projective inducer* from $h_{1\ f}$ to $h_{2\ f}$ iff $h_{2\ f} \approx h_f \circ h_{1\ f}$ and $h_{1\ f} \not\approx h_{2\ f}$.

There are two cases that cause hierarchy of analytical power among SRAs : approximation hierarchy and property hierarchy, as mentioned in Section 2.3. These are clarified by the existence of projective inducers.

The first example is the relation among SAs on non-flat domains (e.g. streams), such as NSA, TSA1 and TSA2.

Let $abs_{NSA} : D \to \{0,1,2,3\}$, $abs_{TSA1} : D \to \{0,1\}$, and $abs_{TSA2} : D \to \{0,1\}$, be as shown in Section 2.3.

Then, their QRs as HOMTs are

$$\begin{array}{rcl} q_{NSA} & = & (abs_{NSA},+,\sqsubseteq_1,Max), \\ q_{TSA1} & = & (abs_{TSA1},+,\sqsubseteq_1,Max), \\ \text{and } q_{TSA2} & = & (abs_{TSA2},+,\sqsubseteq_1,Max). \end{array}$$

Let abstraction maps be

$$abs_{NSA\to TSA1} : \begin{cases} x \to 0 & (\text{for } x \in \{0,1,2\}) \\ x \to 1 & (\text{for } x \in \{3\}) \end{cases}$$
$$abs_{NSA\to TSA2} : \begin{cases} x \to 0 & (\text{for } x \in \{0\}) \\ x \to 1 & (\text{for } x \in \{1,2,3\}) \end{cases}$$

From reduction theorem,

$$\begin{array}{ll} & q_{TSA1} \\ = & (abs_{TSA1},+,\sqsubseteq_1,Max) \\ = & \underline{(abs_{NSA\to TSA1},+,\sqsubseteq_1,Max)} \\ & \circ\ (abs_{NSA},+,\sqsubseteq_1,Max) \end{array}$$

$$\begin{array}{ll} & q_{TSA2} \\ = & (abs_{TSA2},+,\sqsubseteq_1,Max) \\ = & \underline{(abs_{NSA\to TSA2},+,\sqsubseteq_1,Max)} \\ & \circ\ (abs_{NSA},+,\sqsubseteq_1,Max) \end{array}$$

Thus, approximation hierarchy among NSA, TSA1, and TSA2 is clarified as the existence of underlined QRs which are projective inducers.

The second example is the relation among SRAs such as CPA, SIA, and RA on flat domains.

A QR of CPA is $(abs_b,+,\sqsubseteq_{set},id) \circ (id,-,\sqsubseteq_{-0},Min)$. From reduction theorem, SIA is induced from CPA as

$$\begin{array}{lll} & (id,+,\sqsubseteq_0,Min) & \circ\ CPA \\ = & \underline{(id,+,\sqsubseteq_0,Min)} & \circ \\ & (\ (abs_b,+,\sqsubseteq_{set},id) & \circ\ (id,-,\sqsubseteq_{-0},Min)\ ) \\ = & (\ (id,+,\sqsubseteq_0,Min) & \circ\ (abs_b,+,\sqsubseteq_{set},id)\ ) \\ & \circ\ (id,-,\sqsubseteq_{-0},Min) & \\ = & (abs_b,+,\sqsubseteq_0,Min) & \circ\ (id,-,\sqsubseteq_0,Min) \\ = & (abs_b,-,\sqsubseteq_0,Min) & \\ = & SIA & \end{array}$$

Similarly, a pseudo-safe RA (which is safe on non-recursive functions, but not on recursive functions) called PRA has a QR $(abs_b,+,\sqsubseteq_1,Max) \circ (id,-,\sqsubseteq_{-0},Min)$ [12]. PRA is induced from CPA as

$$\begin{array}{lll} & \underline{(id,+,\sqsubseteq_1,Max)} & \circ\ CPA \\ = & (id,+,\sqsubseteq_1,Max) & \circ \\ & (\ (abs_b,+,\sqsubseteq_{set},id) & \circ\ (id,-,\sqsubseteq_0,Min)\ ) \\ = & (\ (id,+,\sqsubseteq_1,Max) & \circ\ (abs_b,+,\sqsubseteq_{set},id)\ ) \\ & \circ\ (id,-,\sqsubseteq_0,Min) & \\ = & (abs_b,+,\sqsubseteq_1,Max) & \circ\ (id,-,\sqsubseteq_{-0},Min) \\ = & PRA & \end{array}$$

These property hierarchies are clarified as the existence of QRs which are projective inducers.

5 Conclusion

A new formalization method for SRAs on first-order applicative languages was proposed. For this purpose, the concept called *HOMomorphic Transformer* (HOMT) was introduced. Intuitively speaking, a HOMT is a

special instance of abstract interpretation. A set of HOMTs is an algebraic space, where equivalence relations (or reduction rules) are defined. This paper has clarified that HOMTs can be used not only for a formalization of possibly non-monotonic SRAs, but also as a transformational mechanism between these analyses. Thus, equivalent and hierarchical relationships among these analyses can be discussed on a unified basis.

There are two directions for further works :

- Relation between algebraic formalization and abstract interpretation.
- Safeness of non-monotonic SRA.

Algebraic formalization was, also, used to investigate the relationship among SRAs [13]. However, the relationship between algebraic formalization and our framework is still open.

As shown in Section 4.3, HOMTs are not enough to formalize some safe non-monotonic SRAs, such as RA. For instance, PRA is not totally safe, although it is safe on non-recursive functions. In fact, $h_{PRA\ cf}(\Omega) \sqsubseteq h_{PRA\ f}(\Omega)$ holds, and it causes an interruption on safeness. Therefore, some other transformational method is required in addition to HOMTs under the restriction of safeness. Such a method must extract a totally undefined function $\Omega(x_1, \ldots, x_n)$ to a strict function (such as for BSA), or a constant function (such as for RA), without seriously affecting the other functions.

Acknowledgments

The authors would like to thank Dr. Katsuji Tsukamoto, Director of NTT Software Research Laboratory, for his guidance and encouragement. They also wish to thank Masaru Takesue and Dr. Naohisa Takahashi, for their useful discussions.

References

[1] Abramsky,S. and Hankin,C.(eds.), "Abstract interpretation of declarative languages," Ellis Horwood Limited (1987)

[2] Bloss,B. and Hudak,P., "Variations on strictness analysis," *ACM Conf. on LISP and Functional Programming*, pp.132-142 (1986)

[3] Clack,C. and Peyton Jones,S.L., "Strictness analysis – a practical approach," *Functional Programming Languages and Computer Architecture*, *LNCS* **201**, Springer-Verlag, pp.35-49 (1985)

[4] Cousot,P. and Cousot,R.,"Abstract Interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," *4th ACM POPL*, pp.238-252 (1977)

[5] Dybjer,P., "Inverse Image Analysis," *4th ICALP*, *LNCS* **267**, Springer-Verlag, pp.21-30 (1987)

[6] Hankin,C.L., Burn,G.L., and Peyton Jones,S.L., "A Safe Approach to Parallel Combinator Reduction," *Theoretical Computer Science*, **56**, pp.17-36 (1988)

[7] Hudak,P. and Young,R., "Higher-order strictness analysis in untyped lambda calculus," *13th ACM POPL*, pp.97-109 (1986)

[8] Hughes,J., "Strictness Detection in Non-Flat Domains," *LNCS* **217**, Springer-Verlag, pp.112-135 (1985)

[9] Mishra,P. and Keller,A.M., "Static Inference of Properties of Applicative Programs," *11th ACM POPL*, pp.235-244 (1984)

[10] Mycroft,A., "The theory and practice of transforming call-by-need into call-by-value," *LNCS* **83**, Springer-Verlag, pp.269-281 (1980)

[11] Nielson,F. "A Bibliography on abstract interpretation," *ACM SIGPLAN Notices*, **21**, 5, pp.31-38 (1986)

[12] Ogawa,M. and Ono,S., "Extension of abstract interpretation of functional programs and its application to global data flow analysis," *Symp. on Functional Programs*, Institute of Physical and Chemical Research Japan, pp.28-37 (1987) (in Japanese)

[13] Ono,S., "Computation Path Analysis : Towards an Autonomous Global Dataflow Analysis" *The Second France-Japan Artificial Intelligence and Computer Science Symposium, Sophia, France* (1987)

[14] Ono,S. and Takahashi,N., "Algorithm for computing dependency property sets in recursive function systems," *Trans. IEICE Japan*, **J69-D**, 5, pp.714-723 (1986) (in Japanese)

[15] Ono,S., Takahashi,N. and Amamiya,M., "Non-strict partial computation with a dataflow machine," *6th RIMS Symposium on mathematical methods in software science and engineering*, TR.**547**, *RIMS Kyoto Univ.*,pp.196-229 (1984)

[16] Ono,S., Takahashi,N. and Amamiya,M., "Optimized demand-driven evaluation of functional programs on a dataflow machine," *IEEE ICPP'86*, pp.421-428 (1986)

[17] Peyton Jones,S.L., "The implementation of functional programming languages," Prentice-Hall (1987)

[18] Smyth,M.B., "Power Domains," *JCSS* **16**, pp.23-36 (1978)

[19] Wadler,P., and Hughes,R.J.M., "Projections for Strictness Analysis," *Functional Programming Languages and Computer Architecture*, *LNCS* **274**, Springer-Verlag, pp.385-407 (1987)