Electronic Notes in Theoretical Computer Science 40 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume40.html 19 pages

A Superposition Operator for the Refinement of Algebraic Models

Claus Pahl [1]

School of Computer Applications
Dublin City University
Dublin, Ireland

Abstract

The development of computer languages or software artefacts from basic concepts to the final product is usually a process starting with an abstract model of a key concept and extending this by adding more detailed functionality for extended structural definitions. We will present a refinement approach for the stepwise development of algebraic models. In each step we either add new elements to a model or refine the properties of existing ones. The process of refining elements such that properties of the original element are preserved is called superposition. We will present a categorical framework for refining algebraic structures. Algebras can be used to model a variety of concepts and objects. Language semantics and formal methods are two application areas which use models represented in terms of algebras.

1 Introduction

The development of computer languages or software artefacts from basic concepts to the final product is usually a process of starting with an abstract model of a key concept and extending this by adding more detailed functionality for extended structural definitions. We propose a layered, stepwise development method for algebraic models. Each new layer either adds new elements to a model or refines the properties of existing ones. Since addition of new elements is a straightforward operation, we address the refinement or redefinition of elements here. The process of redefining elements such that properties for the original element are preserved, shall be called superposition.

We present a categorical framework for refining algebraic structures. Algebras are used to model a variety of concepts and objects. Language semantics [11,12] and formal methods [10] are two application areas which use models represented in terms of algebras. Our approach generalises other extension

[1] Email:cpahl@compapp.dcu.ie

and refinement techniques such as the VDM refinement notion, see [5,6]. Software component technology in another possible application area, where our framework can be used as an adaptation technique in order to re-use a library component in a slightly different context. Our main objective is to obtain a framework for superposition which can be used in the definition of development methodologies for language design or software development. Our framework supports the idea of modularity in design by introducing concepts for a stepwise development in layers. Applying the superposition operator discharges automatically all proof obligations concering property preservation. We will present a framework which allows a language or software designer to create a library of superposition operators for various applications.

An incremental strategy starts with a core model. Elements in a new layer are defined in terms of the layer below. Definitions of elements in the new layer shall superimpose definitions of the respective original elements. A particular problem of this superposition is the preservation of properties of the original elements. We identify two kinds of elements in models: types and functions. We define both and explain notions of property preservation for superpositions of these kinds of elements (Section 2 and 3). A set of constructs for refining these elements is introduced. We argue that the standard notion for structure preservation, the homomorphism, is too restrictive. A more flexible notion is sought. More abstract, observationally oriented notions of property-preservation based on quotients, subobjects and characteristic functions are developed. We investigate how the two forms of property-preserving superpositions interfere. We are going to present an algebraic framework which provides concepts for lifting types and functions such that superposition of original elements by lifted elements with preservation of properties is possible. Elements (types and functions) are transformed to adapt to new structures. Essentially, we define our abstract superposition framework in Section 4. Functions and types and their refinements are formalised as subcategories with corresponding functors. A superposition operator formalising property-preserving refinements of algebras is introduced. The compositionality of superposition is studied.

2 Function Preservation

A function $f\colon A \to B$ is a map from one domain to another. Functions are characterised by some *observable behaviour*, which allows them to be distinguished from other functions with the same domain and codomain. The idea of observable behaviour is essential in our approach. In general, we distinguish two ways in which functions are given: extensionally and intensionally. Extensionally means that functions are given in terms of their input/output behaviour. We follow the intensional view here, distinguishing functions based on some notion of behaviour observation. Our framework centres on the preservation of properties in extensions of algebraic structures. These properties are

characterised in terms of observations on function behaviour.

The function $f\colon A \to B$ – called the **base function** – where A, B are types shall be lifted to $\mathcal{T}f\colon \mathcal{T}A \to \mathcal{T}B$ – called the **lifted function** – where $\mathcal{T}$ maps types (objects) and functions (maps) such that

- properties of A, B are preserved, called **type preservation** – type properties are specified by a type predicate,
- properties of f are preserved, called **function behaviour preservation** – function behaviour is to be preserved by $\mathcal{T}f\colon \mathcal{T}A \to \mathcal{T}B$.

The lifting $\mathcal{T}f$ **superimposes** the definition of f. The mapping $\mathcal{T}$ on maps is constrained: if $f\colon A \to B$ is a map, then $\mathcal{T}f\colon \mathcal{T}A \to \mathcal{T}B$ is a map. This shall be illustrated by a small example.

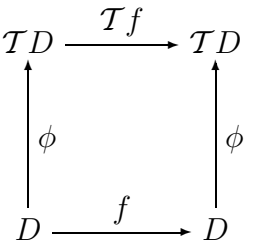
Example 2.1 Let *sqr* denote the usual squaring function $n \mapsto n^2$. Let $A = B = \mathcal{Z}$. Define $\mathcal{T}\mathcal{Z} = \mathcal{Q}$. Then, $\mathcal{T}sqr$ is the lifted squaring function. If we define equivalence classes on $\mathcal{Q}$ - classes of rational values that are mapped to the same integer value - then we expect $\mathcal{T}sqr$ on these classes and *sqr* to show the same behaviour. The equivalence is the observation criterion here.

By introducing two objects A and B we would distinguish two types of domains. However, we shall postpone the introduction of types for domains for some time and work with an untyped universe for the time being. Functions shall be maps on a domain D, e.g. $f\colon D \to D$.

The category of sets shall be the underlying default category. Whenever the term 'domain' is used, the reader can think of sets unless stated otherwise.

2.1 A Notion of Function Preservation

Let us assume an object lifting (a type operator) $\mathcal{T}\colon D \mapsto \mathcal{T}D$ and a domain mapping $\phi\colon D \to \mathcal{T}D$. A straightforward way to define function preservation of $f\colon D \to D$ by a lifted function $\mathcal{T}f\colon \mathcal{T}D \to \mathcal{T}D$ would be based on the commutativity of the following diagram:



(i.e. $\mathcal{T}f \circ \phi = \phi \circ f$). Whenever f maps d to d', we expect $\mathcal{T}f$ to map $\phi \circ d$ to $\phi \circ d'$. The map ϕ preserves a property, here structural information. It preserves the structure f in $\mathcal{T}f$. We could represent this in a category of endomaps for a given base category C. Objects are domains D with endomaps f, maps are C-maps ϕ such that $\phi \circ f = \mathcal{T}f \circ \phi$. The objects are structured,

the structure is imposed by maps f or $\mathcal{T}f$. The map ϕ is equivariant; it **preserves structure** if the equation is satisfied.

An example shall show that this definition is too restrictive for our framework and that a more relaxed notion of observational preservation is needed.

Example 2.2 Consider $(\_ \times D)\colon D \to D \times D$ as the definition for $\mathcal{T}$. Assume that ϕ maps d to $\langle d, d_1\rangle$ and d' to $\langle d', d_2\rangle$. $\mathcal{T}f$ shall be defined as $f \times \mathbf{1}_D$. In this case, the diagram does not commute since $\langle d', d_1\rangle$ is not equal to $\langle d', d_2\rangle$ if f maps d to d', but $\mathcal{T}f$ preserves the behaviour of f in its first component.

Thus, we consider the above definition of function preservation as too restrictive. The given observability criterion for the example – consider the first component only – is sufficient for function preservation. A weaker notion of function preservation shall be introduced. For the given example, points of the product $D \times D$ can be considered as representing the same original element with respect to the lifting $(\_ \times D)$, if they correspond in their first component. Let us make precise what a point is. In the category of sets, a point x of a set X is a unique map $x\colon \mathbf{1} \to X$ where $\mathbf{1}$ is the terminal object, see [7] p.19. Functions are expected to preserve the first component for the given example. In a new layer, we expect additional constructs resulting in additional elements. Several of the extended elements might represent the same base element, i.e. are mapped back to the same base element.

Definition 2.3 An equivalence relation $\sim$ on $\mathcal{T}D$ shall be called a **representation relation** of D in $\mathcal{T}D$ if there is one equivalence class in $\mathcal{T}D/\!\sim$ for each point of D. The representation shall be called **faithful**, if the representation mapping $\phi^{\sim}\colon D \to \mathcal{T}D/\!\sim$ is monic; it shall be called **full**, if the mapping is epic.

Normally, we expect representations to be faithful, i.e. elements distinguishable in the basic layer should be distinguishable in the extension.

Definition 2.4 Let $\mathcal{T}$ be a lifting on domains and functions, and $\phi\colon D \to \mathcal{T}D$ a domain mapping. The lifting $\mathcal{T}f$ **preserves the function** f with respect to the representation $\sim$, if $\mathcal{T}f \circ \phi \circ d \;\sim\; \phi \circ f \circ d$ for any point d and $\mathcal{T}f$ **preserves the representation** of D, i.e. if $x_1 \sim x_2 \Rightarrow \mathcal{T}f \circ x_1 \sim \mathcal{T}f \circ x_2$ for $x_1 = \phi \circ d_1$ and $x_2 = \phi \circ d_2$ and $d_1, d_2\colon \mathbf{1} \to D$.

The second condition states that $\sim$ is a congruence on the D-relevant part of $\mathcal{T}D$. The represention relation can be seen as an observability criterion. Extended elements are observably equal, if they are equivalent, i.e. represent the same basic element in the extension.

2.2 Determine a Representation

Instead of determining the domain mapping ϕ first, we start with the representation relation on domains. The representation is made explicit in our approach, since it will be used as the main element in constructing all ingredients necessary to define an extended layer. This makes our approach different from those where an equivalence is implicitly defined via a retrieve operator [5,6]. Let $x_1, x_2 \colon X \to \mathcal{T}D$ be two maps (e.g. points of $\mathcal{T}D$) with codomain $\mathcal{T}D$ where X is the terminal $\mathbf{1}$. We can refine a relation $\sim^R$ on $\mathcal{T}D \times \mathcal{T}D$ by pairs (x_1, x_2) with $x_1, x_2 \colon X \to \mathcal{T}D$ that are mapped to equivalent values. The relation $\sim^R$ can be expanded to $\sim$, the transitive closure of $\sim^R$ which is the least equivalence relation containing $\sim^R$. Based on the equivalence $\sim$, we define a **quotient** for each domain:

$$\mathcal{T}D/\!\sim \;\; = \{S \subseteq \mathcal{T}D \mid y_1 \sim y_2 \text{ for all } y_1, y_2 \text{ in } S\}$$

$\mathcal{T}D$ is partitioned into equivalence classes, which together form the quotient.

Example 2.5 Let us look at products again. Two elements x_1, x_2 of $\mathcal{T}D = D \times D$ for $\mathcal{T} = (\_ \times D)$ can be considered equivalent, if their first components are equal: $x_1 \sim x_2$ if $p_1 \circ x_1 = p_1 \circ x_2$ for $x_1 = \langle p_1 \circ x_1, p_2 \circ x_1 \rangle$ and $x_2 = \langle p_1 \circ x_2, p_2 \circ x_2 \rangle$ where p_1 and p_2 are projections onto the first and second element, respectively. Then, x_1 and x_2 represent the same element of D, namely $p_1 \circ x_1$ (or $p_2 \circ x_2$).

We can relate $\mathcal{T}D$ and its quotient $\mathcal{T}D/\!\sim$ by an injection $\iota \colon \mathcal{T}D \hookrightarrow \mathcal{T}D/\!\sim$.

Proposition 2.6 *An injection $\iota \colon \mathcal{T}D \hookrightarrow \mathcal{T}D/\!\sim$ from any object into its quotient always exists*

Proof. See [2,3]. □

There is another property of quotients and their inclusion. A map $h \colon B \to C$ is a coequaliser of $f, g \colon A \to B$, if $h \circ f = h \circ g$ and for any map $k \colon B \to D$ for which $k \circ f = k \circ g$, there is is a unique map $l \colon C \to D$ such that $l \circ h = k$ (see [1] p.239). Coequalisers generalise equivalence relations.

Proposition 2.7 *An injection $\iota \colon \mathcal{T}D \hookrightarrow \mathcal{T}D/\!\sim$, which assigns an equivalence class for each point of $\mathcal{T}D$, is a coequaliser of points $x_1, x_2 \colon \mathbf{1} \to \mathcal{T}D$.*

Proof. Follows directly from Proposition 8.4.2 in [1]. □

Having the existence of the inclusion ι guaranteed, we might want to consider the inverse of ι. The resulting map is a choice operator, called δ, which assigns representatives for each equivalence class:

$$\mathcal{T}D \underset{\delta}{\overset{\iota}{\rightleftarrows}} \mathcal{T}D/\!\sim$$

Proposition 2.8 *Assume an injection* $\iota\colon \mathcal{T}D \hookrightarrow \mathcal{T}D/\sim$. *Then, a map* $\delta\colon \mathcal{T}D/\sim \to \mathcal{T}D$ *exists such that* ι *is a retraction of* δ, *i.e.* $\iota \circ \delta = 1_{\mathcal{T}D/\sim}$.

Proof. See [7] p.72/73. □

Based on a given equivalence on $\mathcal{T}D$ – which can be derived from a relation specification – the existence of maps ι and δ between the lifted domain $\mathcal{T}D$ and its quotient $\mathcal{T}D/\sim$ is guaranteed. These results will be useful in the construction of a function preserving extension, including the construction of the domain mapping $\phi\colon D \to \mathcal{T}D$.

Example 2.9 For products, we can define the choice operator $\delta\colon \mathcal{T}D/\sim \to \mathcal{T}D$ by $\delta\colon [(d, d_0)]_\sim \mapsto (d, d_0)$ and the injection $\iota\colon \mathcal{T}D \hookrightarrow \mathcal{T}D/\sim$ by $\iota\colon (d, d') \mapsto [(d, d_0)]_\sim$ for all $d'\colon D$, where d_0 is any fixed element of D.

2.3 Constructing a Representation Mapping

The quotient captures what has to be preserved in a function lifting. We are going to construct a *representation mapping* $\phi^\sim\colon D \to \mathcal{T}D/\sim$ for a *domain mapping* $\phi\colon D \to \mathcal{T}D$. Based on $\sim$, the map $\phi^\sim$ shall associate an equivalence class to each point of D.

Definition 2.10 A **representation mapping** $\phi^\sim\colon D \to \mathcal{T}D/\sim$ is called

- **faithful**, if it is monic (i.e. $d_1 \neq d_2 \Rightarrow \phi^\sim \circ d_1 \neq \phi^\sim \circ d_2$ for any $d_1, d_2\colon D$),
- **full**, if it is epic (i.e. $t_1 \circ \phi^\sim = t_2 \circ \phi^\sim \Rightarrow t_1 = t_2$ for $t_1, t_2\colon \mathcal{T}D/\sim \to X$).

Normally, we expect $\phi^\sim$ to be faithful, since it guarantees that distinguishable points of the base domain are distinguishable when mapped into the lifted domain. In general, the map $\phi^\sim$ will not be full, but if that is the case we get isomorphsm between the basic domain and the quotient of the extension.

With the results obtained so far, such as existence of inclusion and choice, we can now define the **domain mapping** ϕ.

Definition 2.11 The **domain mapping** ϕ shall be defined by $\phi := \delta \circ \phi^\sim$.

The user specifies the representation relation $\sim$ and the representation map $\phi^\sim$ based on the lifting $\mathcal{T}$. The rest can be derived. The elements $\mathcal{T}$, $\sim$ and $\phi^\sim$ are the basic ingredients of a function preserving extension.

Definition 2.12 The **lifting triple** $\langle \mathcal{T}, \sim, \phi^\sim \rangle$ consists of a lifting operator $\mathcal{T}$, a representation relation $\sim$ and a representation mapping $\phi^\sim$.

It should be noted here that $\mathcal{T}$ is not generally a functor. The lifting triple can not expected to be a monad - a confusion might occur since some authors use the name triple for monads. Triples are different from monads here.

Example 2.13 For products, we define $\phi^{\sim}: d \mapsto [(d, d_0)]_{\sim}$ for any $d_0: D$ and derive $\phi: d \mapsto (d, d_0)$. For products, we have a full and faithful presentation.

2.4 Construct a Function Lifting

Given an object lifting $\mathcal{T}$ on domains, a representation $\sim$ and a representation mapping $\phi^{\sim}$, we have constructed the domain mapping ϕ. The remaining construct to be defined is the function lifting. Lifting $\mathcal{T}f$ preserves the function f, if $[\mathcal{T}f] \circ \phi^{\sim} = \phi^{\sim} \circ f$, where $[\mathcal{T}f]$ is defined by $[\mathcal{T}f] \circ [x]_{\sim} = [x']_{\sim}$ if $\mathcal{T}f \circ x = x'$ for points x and x' of $\mathcal{T}D$. Given a representation relation $\sim$, a lifting $\mathcal{T}f: \mathcal{T}D \to \mathcal{T}D$ has also to satisfy the substitution property of congruences $x_1 \sim x_2 \Rightarrow \mathcal{T}f \circ x_1 \sim \mathcal{T}f \circ x_2$ for $x_1, x_2: \mathcal{T}D$ with $x_1 = \phi \circ d_1$ and $x_2 = \phi \circ d_2$ and $d_1, d_2: D$.

Definition 2.14 Let x be a point of $\mathcal{T}D$. A lifted function $\mathcal{T}f$ for function f which preserves the behaviour of f is defined as follows:

$$\mathcal{T}f \circ x = \begin{cases} \phi \circ f \circ d \text{ if } x \sim \phi \circ d \text{ for some point } d \text{ of } D \\ y \qquad\qquad \text{otherwise, where } y \text{ is any point of } \mathcal{T}D \end{cases}$$

This defines $\mathcal{T}f$ based on f for all points of $\mathcal{T}D$. If $\phi^{\sim}$ is full, we can simplify the definition, i.e. $\mathcal{T}f \circ x = \phi \circ f \circ d$ for $x \sim \phi \circ d$ for some point d of D.

Proposition 2.15 *The lifting $\mathcal{T}f$ based on f as defined in Def. 2.14 is function preserving.*

Proof. Let $f \circ d = d'$ for any $d: D$. Then $\mathcal{T}f \circ x = \phi \circ f \circ d = \phi \circ d'$ for arguments x which are equivalent to $\phi \circ d$. Thus, we have $\mathcal{T}f \circ \phi = \phi \circ f$. This is even equality instead of equivalence. This is obviously a congruence on the D-relevant part, i.e. satisfies the substitution property $x_1 \sim x_2 \Rightarrow \mathcal{T}f \circ x_1 \sim \mathcal{T}f \circ x_2$ for any $x_1, x_2: \mathcal{T}D$ with $x_1 = \phi \circ d_1$ and $x_2 = \phi \circ d_2$ and $d_1, d_2: D$. □

Example 2.16 For products, we have the case that $\phi^{\sim}$ is full, thus we define $\mathcal{T}f \circ (p_1 \circ x, p_2 \circ x) = \phi \circ f \circ a$ for $x \sim \phi \circ p_1 \circ x$ for some point $p_1 \circ x$ of D.

3 Type Preservation

A domain can be constrained by a type predicate. We consider types as explicit objects, we also consider the predicate as a truth-valued map which includes or excludes elements from the domain. Our approach to representing types will use slice categories, see [1] p.35. However, we also look at monoid actions and types in Section 3.2 in order to introduce an alternative.

3.1 Parts and Characteristic Function

There is a duality between parts (or subobjects) of an object, related by an inclusion $\iota\colon S \hookrightarrow D$ where S is a part of D (denoted $S \subseteq D$) and a characteristic function $\chi_S\colon D \to \Omega$. The characteristic function determines whether an element of X is a part (or a subobject) or not. Ω is a truth-value object – a standard way of defining Ω is $2 = \{true, false\}$ for the Boolean topos **Set**. The truth-value object is unique to its own topos [7] p.348. In more structured categories Ω might have more structure for the truth value. Let Γ be a set of types. In the category of sets we can state: for any $x\colon \Gamma \to X$, x is included in the part (S, ι) of X iff $\chi_S(x) = true_\Gamma$ for $\chi_S\colon X \to 2$ and $true_\Gamma\colon \Gamma \to 1 \to 2$. We can combine the two constructs:

$$S \xhookrightarrow{\iota} D \xrightarrow{\chi_S} \Omega$$

In general, Ω is a truth object if for any object X; maps $\chi_S\colon X \to \Omega$ are natural bijections of $\iota\colon S \hookrightarrow X$, i.e. for each subobject $S \subseteq X$ there is exactly one $\chi_S\colon X \to \Omega$ (see [7]).

All types we are going to introduce are subobjects of the domain D. Due to the strict typing approach of category theory, an element of a subobject cannot be an element of another object at the same time.

Definition 3.1 The **category of subobjects (or types) of** D, abbreviated C/D, in a category C can be defined as follows:

- inclusion maps $\alpha\colon A \hookrightarrow D$, $\beta\colon B \hookrightarrow D$, ... are objects
- a map from $\alpha\colon A \hookrightarrow D$ to $\beta\colon B \hookrightarrow D$ is a C-map $f\colon A \to B$ such that $\beta \circ f = \alpha$.

It follows that C/D is indeed a category, see [7]. If f exists, it is unique. This means that there is at most one map between two objects. In that case, we indicate $A \subseteq_D B$ (A is included in B over D). Due to the uniqueness of f, C/D is a preordered set (poset). If additionally a map $g\colon B \to A$ exists such that $\alpha \circ g = \beta$, then $\alpha\colon A \to D$ and $\beta\colon B \to D$ are isomorphic objects, denoted $A \cong B$. If $\alpha\colon A \to D$, $x\colon \Gamma \to D$, and $x \in A$ and $A \subseteq_D B$, then $x \in B$.

The need to relate or combine different predicates might arise in our superposition approach. Suppose $\chi_{\mathcal{T}A}$, the characteristic function for an extended type, is constructed from χ_A, the characteristic function for a basic type. It might be necessary to introduce another predicate on $\mathcal{T}A$ resulting in a combination, weakening or strengthening of $\chi_{\mathcal{T}A}$. The category of parts is the framework to explore relations between characteristic functions.

3.2 Actions and Types

Before continuing with type extensions, we shall be look at an alternative way of dealing with types. We follow [1] p.64ff here closely. Consider the monoid $(F, \circ, \mathbf{1})$ where F is the set of functions and $\circ$ is the composition. Define a mapping $\alpha\colon F \times D \to D$ by $\alpha(f, d) = f \circ d$ for $f\colon F$ and $d\colon D$. We shall write $f(d)$ for $f \circ d$ in the remainder of this section. domains.

Proposition 3.2 *The map α is a monoid action.*

Proof. $\alpha(\mathbf{1}, s) = s$ and $\alpha(f \circ g, s) = \alpha(f, \alpha(g, s)) = \alpha(f, g(d)) = f(g(s))$. □

Types are introduced for domains through a type function $type\colon D \to \Gamma$. Let Γ be a set of types. Each $t \in \Gamma$ is defined by $\{d \in D \mid type(d) = t\}$. Domain and codomain of functions are specified by $input\colon F \to \Gamma$ and $output\colon F \to \Gamma$. We expect $input(f_2) = output(f_1)$ for a composite $f_2 \circ f_1$ to be well-defined. We assume $\mathbf{1}_t \in F$ for each t with $input(\mathbf{1}_t) = output(\mathbf{1}_t)$ as the identity. We can define a **typed universe**, represented by a category C^Γ where elements of Γ are the objects and elements f of F with $input(f)$ and $output(f)$ as domain and codomain, respectively, are the maps. The category C^Γ is well-defined. We write as an abbreviation $f\colon A \to B$ for $f\colon D \to D$, $input(f) = A$ and $output(f) = B$. We consider elements of the type set Γ as objects. Soon, we will see that these type objects are subobjects of the domain D.

3.3 Extending a Type

We define type preservation first, and then consider how to construct type preserving extensions. A lifting triple $\langle \mathcal{T}, \sim, \phi^\sim \rangle$ shall be assumed for this discussion.

Definition 3.3 If a domain D is lifted to a domain $\mathcal{T}D$, then the **type** A constraining D is **preserved**, if the following diagram commutes:

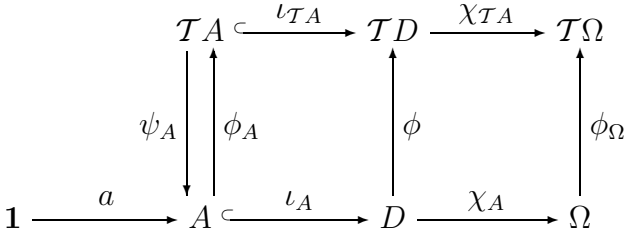
$$\begin{array}{ccc} \mathcal{T}A & \xrightarrow{\iota_{\mathcal{T}A}} & \mathcal{T}D \\ \uparrow{\scriptstyle \phi_A} & & \uparrow{\scriptstyle \phi} \\ A & \xrightarrow{\iota_A} & D \end{array}$$

The mappings ι_A, $\iota_{\mathcal{T}A}$ are inclusions, ϕ maps from D to $\mathcal{T}D$, ϕ_A maps from type A to $\mathcal{T}A$.

A type is characterised by a subobject, e.g. $\mathcal{T}A$, and its inclusion, e.g. $\iota_{\mathcal{T}A}$. The diagram in the previous definition formalises type preservation: $(\mathcal{T}A, \iota_{\mathcal{T}A})$ preserves (A, ι_A). Due to the duality of concepts, types can also be represented by characteristic functions. We can construct a subobject based on a given characteristic function, and vice versa. Here is the alternative definition.

Definition 3.4 The characteristic function $\chi_{\mathcal{T}A}$ **preserves the type** χ_A, if $\chi_{\mathcal{T}A} \circ \phi = \phi_\Omega \circ \chi_A$.

The following diagram is an elaboration of the above one with other constructs discussed.



It should be remembered here that there is a unique truth value object **2** that we have introduced earlier on.

3.4 Constructing a Type Preserving Lifting

After defining type preserving extensions, we now look at how to construct such an extension. We look at characteristic functions here – keeping in mind that we can construct the corresponding subobject inclusions at any time. Let us assume a characteristic function $\chi_A \colon D \to \Omega$, as well as maps $\phi_A \colon A \to \mathcal{T}A$, $\phi_\Omega \colon \Omega \to \mathcal{T}\Omega$ and a retraction $\psi_A \colon \mathcal{T}A \to A$ such that $\psi_A \circ \phi_A = 1_A$.

Definition 3.5 Define the lifted characteristic function $\chi_{\mathcal{T}A} \colon \mathcal{T}D \to \mathcal{T}\Omega$ by:

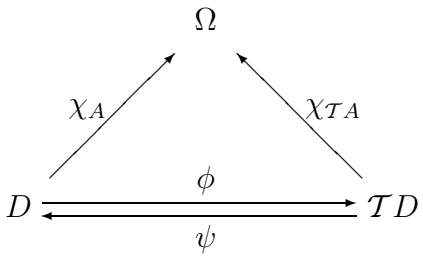
$$\chi_{\mathcal{T}A} \circ \iota_{\mathcal{T}A} \circ x = \begin{cases} \phi_\Omega \circ \chi_A \circ \iota_A \circ \psi \circ x & \text{for all } x = \phi_A \circ a \text{ for some point } a \colon A \\ false & \text{otherwise} \end{cases}$$

We will soon investigate in more detail what ψ is and when it exists.

Proposition 3.6 *The mapping $\chi_{\mathcal{T}A}$ is a well-defined type preserving extension of χ_A.*

Proof. A diagram for Def. 3.4 commutes for the given definition. □

Type preservation can be looked at what is called a determination problem, see [7]. The truth-value object is unique for a category, i.e. $\mathcal{T}\Omega = \Omega$. We have now the following determination problem:



where the lifted characteristic function $\chi_{\mathcal{T}A}$ for χ_A is sought.

Proposition 3.7 *If the map ϕ has a retraction ψ – i.e. if $\psi \circ \phi = 1_A$ – then the above diagram commutes, i.e. $\chi_{\mathcal{T}A} = \chi_A \circ \psi$.*

Proof. [7] p.72f. □

The retraction can be used to construct a solution for the determination problem. We would like to know when such a retraction exists.

Proposition 3.8 *A retraction ψ for ϕ exists, if ϕ is monic, i.e. $\phi \circ d_1 = \phi \circ d_2 \Rightarrow d_1 = d_2$ for any two points d_1, d_2 of D.*

Proof. We can define $\psi\colon \mathcal{T}D \to D$ by $\psi \circ x = d_1$ for $\phi \circ d_1 = x$ (injectivity). ψ is a retraction if $\psi \circ \phi \circ d_1 = d_1$. This is true due to the definition of ψ. □

The two propositions guarantee that our standard type preserving lifting according to Definition 3.5 for characteristic functions always works.

3.5 Type and Function Preservation

The combination of both forms of preservation, type and function preservation, shall result in a function lifting which respects types. The main problem is that the representation is introduced on the original domain, not on its constrained forms, the types. We carry out two investigations. Firstly, we consider the integration of the representation into the characteristic function definition. Secondly, we integrate types into function preservation.

We extend the definition of $\chi_{\mathcal{T}A}$ to an extended form $\chi^{\sim}_{\mathcal{T}A}$ which also respects the representation $\sim$.

Definition 3.9 The extended lifting $\chi^{\sim}_{\mathcal{T}A}$ for characteristic function lifting $\chi_{\mathcal{T}A}$ is defined by:

$$\chi^{\sim}_{\mathcal{T}A} \circ \iota_{\mathcal{T}A} \circ x = \begin{cases} \chi_{\mathcal{T}A} \circ \iota_{\mathcal{T}A} \circ x & \text{if } x \sim \phi \circ a \text{ for some point } a\colon A \\ false & \text{otherwise} \end{cases}$$

The map $\chi^{\sim}_{\mathcal{T}A}$ yields the same result for all points $x\colon \mathcal{T}A$ which are equivalent to some $\phi \circ a$. The extended form $\chi^{\sim}_{\mathcal{T}A}$ subsumes the standard construction $\chi_{\mathcal{T}A}$.

Let us now consider so-called type-compliant functions for a domain D.

Definition 3.10 A function $f\colon A \to B$ is called **type-compliant** with the characteristic functions $\chi_A\colon D \to \Omega$ and $\chi_B\colon D \to \Omega$, if whenever $\chi_A \circ \iota_A \circ a = true$ for some point a of A, then $\chi_B \circ \iota_B \circ f \circ a = true$.

The lifted function $\mathcal{T}f$ should preserve the function f, but $\mathcal{T}f$ should also respect the type constraints, i.e. should be type-compliant, if f is so.

Proposition 3.11 *Assume the maps* $\phi_A\colon A \to \mathcal{T}A$, $\phi_B\colon B \to \mathcal{T}B$, $f\colon A \to B$, $\mathcal{T}f\colon \mathcal{T}A \to \mathcal{T}B$ *and corresponding characteristic functions* $\chi_A\colon D \to \Omega$, $\chi_B\colon D \to \Omega$, $\chi^{\sim}_{\mathcal{T}A}\colon \mathcal{T}D \to \mathcal{T}\Omega$ *and* $\chi^{\sim}_{\mathcal{T}B}\colon \mathcal{T}D \to \mathcal{T}\Omega$ *for a given domain* D. *If*

(i) $\mathcal{T}f$ *preserves function* f,

(ii) (a) $\chi^{\sim}_{\mathcal{T}A}$ *preserves type* χ_A *and* (b) $\chi^{\sim}_{\mathcal{T}B}$ *preserves type* χ_B,

(iii) f *is compliant with* χ_A *and* χ_B,

then, $\mathcal{T}f$ *is compliant with* $\chi^{\sim}_{\mathcal{T}A}$ *and* $\chi^{\sim}_{\mathcal{T}B}$.

Proof. We assume $\chi_A \circ \iota_A \circ a \Rightarrow \chi^{\sim}_{\mathcal{T}A} \circ \iota_{\mathcal{T}A} \circ \phi_A \circ a$ (2a) and $\chi_B \circ \iota_B \circ f \circ a \Rightarrow \chi^{\sim}_{\mathcal{T}B} \circ \iota_{\mathcal{T}B} \circ \phi_B \circ f \circ a$ (2b). We have to show: if $\chi_A \circ \iota_A \circ a \Rightarrow \chi_B \circ \iota_B \circ f \circ a$ (3), then $\chi^{\sim}_{\mathcal{T}A} \circ \iota_{\mathcal{T}A} \circ \phi_A \circ a \Rightarrow \chi^{\sim}_{\mathcal{T}B} \circ \iota_{\mathcal{T}B} \circ \mathcal{T}f \circ \phi_A \circ a$. From $\chi_A \circ \iota_A \circ a$ we get $\chi^{\sim}_{\mathcal{T}A} \circ \iota_{\mathcal{T}A} \circ \phi_A \circ a$ via (2a) and $\chi_B \circ \iota_B \circ f \circ a$ via (3), from the latter also $\chi^{\sim}_{\mathcal{T}B} \circ \iota_{\mathcal{T}B} \circ \phi_B \circ f \circ a$ via (2b). We know that $\mathcal{T}f \circ \iota_{\mathcal{T}A} \circ \phi_A \circ a \sim \chi_B \circ \iota_B \circ f \circ a$ since $\mathcal{T}f$ preserves the function f via (1). $\chi^{\sim}_{\mathcal{T}B}$ works as an equalizer on the equivalent values: $\chi^{\sim}_{\mathcal{T}B} \circ \mathcal{T}f \circ \iota_{\mathcal{T}A} \circ \phi_A \circ a = \chi^{\sim}_{\mathcal{T}B} \circ \chi_B \circ \iota_{\mathcal{T}B} \circ f \circ a$. □

Since there is only one truth-value object $\Omega = \mathcal{T}\Omega$, we have $\chi_{\mathcal{T}A} = \chi_A \circ \psi$ where ψ is a retraction of ϕ. If $\phi^{\sim}$ is a full representation mapping, we can define $\chi_{\mathcal{T}A} = \chi_A \circ \psi^{\sim} \circ \iota$ where $\psi^{\sim}$ is a retraction of $\phi^{\sim}$. The map ψ is not needed in this construction, we can construct via the quotient.

4 Superposition

We shall now attempt to summarise the previous results and define a comprehensive superposition operator which shall guarantee type and function preservation for lifted types and functions.

4.1 Basic Categories and Functors for Type and Function Lifting

Let $\mathcal{T}$ be a lifting – technically an endomap on a given category, i.e. it maps certain objects (domains and types) to the corresponding lifted objects. It respects the function typing, i.e. $f\colon A \to B$ is mapped to $\mathcal{T}f\colon \mathcal{T}A \to \mathcal{T}B$. However, $\mathcal{T}$ might not be a functor. All elements participating in property-preserving liftings (based on the lifting triple and derived constructs) shall be collected in a construct called the superposition category.

Definition 4.1 A **superposition category** E for lifting triple $\langle \mathcal{T}, \sim, \phi^{\sim} \rangle$ and a given base category C shall contain the following elements:

- Objects (all objects are C-objects):
  - domain D, types $A, B, \ldots$, a truth-value object Ω,
  - extensions $\mathcal{T}_1 X, \mathcal{T}_2 X, \ldots$ where X is type, domain or truth-value object or extension, and $\mathcal{T}_1, \mathcal{T}_2, \ldots$ are mappings on objects,

  - quotients $\mathcal{T}X/\sim_1, \mathcal{T}X/\sim_2, \ldots$ for extensions $\mathcal{T}X$ of domains and types with respect to representations $\sim_1, \sim_2, \ldots$.
- Maps (all maps are C-maps):
  - identities $\mathbf{1}_X : X \to X$ on all objects X,
  - functions on domains, types, and quotients $f : X \to X$ where X is domain, type or quotient,
  - characteristic functions $\chi_X : D \to \Omega$ from domains or types to truth-value objects,
  - maps between a domain and quotient, between type and domain, between domain and extended domain, between domain and quotient of extension, between types.

We assume a single domain D of values, similar to a universe domain in some type systems.

Proposition 4.2 *The superposition category E is well-defined.*

Proof. Identity and composition are those defined for C. The category E is a subcategory of the base category C. □

Based on the category E, which comprises all elements needed in our approach, we define two major subcategories of E which will capture function preservation, viz. E^f, and type preservation, viz. E^t, in isolation. Additionally, two functors expressing the extensions will be defined on these subcategories.

4.1.1 Functions

We are going to define a subcategory of the superposition category which captures the concepts of function preservation in a categorical setting.

Definition 4.3 The **category of functions** E^f for the superposition category E shall contain the following (all elements are taken from category E):

- Objects: domain D, types $A, B, \ldots$, extensions $\mathcal{T}A, \mathcal{T}B, \ldots$ and quotients $\mathcal{T}A/\sim, \mathcal{T}B/\sim, \ldots$ for any $\mathcal{T}$ and $\sim$.
- Maps: functions on domains, types, quotients and identities on each object.

Proposition 4.4 *Category E^f is well-defined and forms a full subcategory of extension category E.*

Proof. Since composition and identity are elements of E, E^f is well-defined. The set of objects is a subset of objects of E. Maps of E^f also form a subset of maps of E. Thus, E^f is a subcategory of E. For each typed set of functions (hom-set) of E, the whole set forms the corresponding hom-set in E^f. Thus, we have a full subcategory. □

We can construct an inclusion functor $\Upsilon^f : E^f \hookrightarrow E$ mapping elements of E^f into E. The functor Υ^f is a monomorphism. The category E^f is the

structural framework on which a functor $\mathcal{T}^f$ describes function preserving liftings.

Definition 4.5 The endofunctor $\mathcal{T}^f$ on E^f, called the **function preservation functor**, shall be defined as follows:

- $\mathcal{T}^f$ maps the domain D to the quotient of the lifted object $\mathcal{T}D/\sim$.
- $\mathcal{T}^f$ maps a function $f\colon D \to D$ to $[\mathcal{T}f]\colon \mathcal{T}D/\sim \to \mathcal{T}D/\sim$ such that function behaviour is preserved.

Proposition 4.6 *The functor $\mathcal{T}^f$ is well-defined.*

Proof. Identity and composition have to be considered. $\mathcal{T}^f(\mathbf{1}_D) = \mathbf{1}_{\mathcal{T}D/\sim}$ due to function preservation $\phi^\sim \circ [\mathbf{1}_D] = [\mathbf{1}_{\mathcal{T}D}] \circ \phi^\sim$ and $\mathcal{T}^f(g \circ f) = \mathcal{T}^f(g) \circ \mathcal{T}^f(f)$ due to composability of functions in E^f. □

A functor is faithful, if the induced mapping on each hom-set is injective. $\mathcal{T}^f$ satisfies this property.

Proposition 4.7 *The functor $\mathcal{T}^f$ is faithful, if the underlying representation mapping $\phi^\sim$ is faithful.*

Proof. Injectivity is the key requirement in the definition of a faithful representation mapping $\phi^\sim$ on which $\mathcal{T}^f$ is based for each $f\colon A \to B$. Suppose f maps a to b_1 and g maps a to b_2 for $g\colon A \to B$. f and g are distinguishable. If b_1 and b_2 are distinguishable, so will be $\phi^\sim \circ b_2$ and $\phi^\sim \circ b_2$. Thus, $[\mathcal{T}f]$ and $[\mathcal{T}g]$ are distinguishable. □

However, $\mathcal{T}^f$ is not necessarily a full functor. It is normally also not unique – this is only the case if the representation mapping is full. In that case $\mathcal{T}^f$ is an isomorphism.

4.1.2 Types

Analogously to functions, we are going to define a category E^t and a functor $\mathcal{T}^t$ based on the superposition category E and a lifting triple $\langle \mathcal{T}, \sim, \phi^\sim \rangle$ to capture type preservation. Category E^t satisfies properties similar to those of E^f.

Definition 4.8 The **category of types** E^t for the superposition category E shall contain the following elements:

- objects: domain D, types $A, B, \ldots$, extensions $\mathcal{T}A, \mathcal{T}B, \ldots$ and the truth-value objects.
- maps: characteristic functions and identities on each object.

Proposition 4.9 *Category E^t is well-defined and forms a full subcategory of the superposition category E.*

Proof. Analogously to Proposition 4.4. □

Analogously, we can construct an inclusion functor $\Upsilon^t\colon E^t \hookrightarrow E$, where Υ^t is a monomorphism. Let us now define the endofunctor $\mathcal{T}^t$ on E^t.

Definition 4.10 The endofunctor $\mathcal{T}^t$ on E^t, called the **type preservation functor**, shall be defined as follows:

- $\mathcal{T}^t$ maps the domain D to $\mathcal{T}D$ and the truth-value object Ω to $\mathcal{T}\Omega$.
- $\mathcal{T}^t$ maps a characteristic function $\chi_A\colon D \to \Omega$ to $\chi^{\sim}_{\mathcal{T}A}\colon \mathcal{T}D \to \mathcal{T}\Omega$ such that types are preserved. In particular, $\mathcal{T}^t$ maps identities $\mathbf{1}_X$ to $\mathbf{1}_{\mathcal{T}X}$ for domains.

The truth-value object is unique to the category, i.e. $\Omega = \mathcal{T}\Omega$. Remember, that applying Definition 3.9 to define $\chi^{\sim}_{\mathcal{T}D}$ guarantees type preserving function liftings. We do not have to look at this issue explicitly.

Proposition 4.11 *The functor $\mathcal{T}^t$ is well-defined.*

Proof. Identity and associativity of composition have to be considered. It holds $\mathcal{T}^t(\mathbf{1}_D) = \mathbf{1}_{\mathcal{T}^t D}$ by definition and $\mathcal{T}^t(g \circ f) = \mathcal{T}^t(g) \circ \mathcal{T}^t(f)$, since at least one of f, g has to be the identity (the characteristic map can only be composed with identities in this category). □

Proposition 4.12 *Functor $\mathcal{T}^t$ is faithful, if the underlying mapping ϕ is monic.*

Proof. Analogously to $\mathcal{T}^f$. □

The functor $\mathcal{T}^t$ is not necessarily a full functor. It is normally also not unique. This is only the case if it full (and faithful), and thus, an isomorphism.

4.1.3 Natural Transformations

Now, we are going to reformulate the mappings $\phi^{\sim}$ and ϕ on a higher level of abstraction as natural transformations between functors on the function category and the type category, respectively. We look at functions first. A natural transformation in the context of function preservation is a function ϕ that assigns a map $\phi_A\colon \mathbf{1}(A) \to \mathcal{T}(A)$ for each object A. We define for the remainder $\phi^f :\, = \phi^{\sim}$ and $\phi^t :\, = \phi$ in order to achieve a consistent style of naming when all constructs will finally be assembled.

Proposition 4.13 *Let $\mathcal{T}^f$ be an endofunctor and $\mathbf{1}^f$ the identity functor on*

E^f. Then, ϕ^f is a natural transformation from $\mathbf{1}^f$ to $\mathcal{T}^f$:

$$\begin{array}{ccc} \mathcal{T}^f A & \xrightarrow{[\mathcal{T}^f f]} & \mathcal{T}^f B \\ \uparrow{\scriptstyle \phi^f_A} & & \uparrow{\scriptstyle \phi^f_B} \\ \mathbf{1}^f A & \xrightarrow{\mathbf{1}^f f} & \mathbf{1}^f B \end{array}$$

Proof. The diagram commutes since $\mathcal{T}^f$ is an endofunctor on E^f, see Prop. 4.6. Commutativity is guaranteed due to function preservation. □

Proposition 4.14 *Let $\mathcal{T}^t$ be an endofunctor on E^t and $\mathbf{1}^t$ the identity functor on E^t. Then, ϕ^t is a natural transformation from $\mathbf{1}^t$ to $\mathcal{T}^t$:*

$$\begin{array}{ccc} \mathcal{T}^t D & \xrightarrow{\mathcal{T}^t \chi_{\mathcal{T}A}} & \mathcal{T}^t \Omega \\ \uparrow{\scriptstyle \phi^t_D} & & \uparrow{\scriptstyle \phi^t_\Omega} \\ \mathbf{1}^t D & \xrightarrow{\mathbf{1}^t \chi_A} & \mathbf{1}^t \Omega \end{array}$$

Proof. The diagram commutes since $\mathcal{T}^t$ is an endofunctor on E^t, see Prop. 4.11. Commutativity is guaranteed due to the type preservation requirement. □

4.2 The Superposition Operator

Finally, all constructs will be assembled together in order to form a type and function preserving superposition operator which, for a given model (a set of objects and maps) and a lifting triple, constructs a lifted model (another set of objects and maps) which superimposes the definition of the base one.

Definition 4.15 Let E be a superposition category for a lifting triple $\langle \mathcal{T}, \sim, \phi^{\sim} \rangle$ with full subcategories E^t (the type category) and E^f (the function category). The quintuple

$$\langle\ \mathcal{T}\colon E \to E,\ \mathcal{T}^t\colon E^t \to E^t,\ \phi^t\colon \mathbf{1}^t \to \mathcal{T}^t,\ \mathcal{T}^f\colon E^f \to E^f,\ \phi^f\colon \mathbf{1}^f \to \mathcal{T}^f\ \rangle$$

or $\langle \mathcal{T}, \mathcal{T}^t, \phi^t, \mathcal{T}^f, \phi^f \rangle$ is a **superposition**, if the following is satisfied:

(i) $\mathcal{T}$ is an endomap on E,

(ii) $\mathcal{T}^t$ is an endofunctor on E^t,

(iii) ϕ^t is a natural transformation from $\mathbf{1}^t$ to $\mathcal{T}^t$ on E^t,

(iv) $\mathcal{T}^f$ is an endofunctor on E^f,

(v) ϕ^f is a natural transformation from $\mathbf{1}^f$ to $\mathcal{T}^f$ on E^f.

The operator is well-defined, i.e. it guarantees type and behaviour preservation of functions $f\colon A \to B$ in liftings $\mathcal{T}f\colon \mathcal{T}A \to \mathcal{T}B$. Remember that type compliancy of the extension (which is caused by interaction between type and function properties) is guaranteed, if the basic function is type-compliant.

Theorem 4.16 [Superposition Theorem] *Every type and function property of the underlying model is a property of the transformed model if the superposition operator is applied.*

Proof. Both transformations, represented by $(\mathcal{T}^t, \phi^t)$ for type liftings and $(\mathcal{T}^f, \phi^f)$ for function liftings preserve the respective properties, see Propositions 4.6, 4.11, 4.13 and 4.14. □

4.3 Laws of Superposition

After defining our main extension operator, the superposition, we are going to investigate some properties connected to this operator. One of the important questions addresses the compositionality of the operator. Let us summarise the context briefly. The underlying mappings $\mathcal{T}_1, \mathcal{T}_2, \ldots$ are in general not functors; associativity of composition is consequently not guaranteed. Composition of functors $\mathcal{T}_1^f$ and $\mathcal{T}_2^f$ on quotients and analogously for the type functor involving truth-value objects is not relevant. These constructs only constrain one particular step. Quotients, for instance, are a means to capture which behaviour has to be preserved for a single step; it is not relevant for a second lifting. Subject to composition are only mappings on types, e.g. from A to $\mathcal{T}_1 A$ and from $\mathcal{T}_1 A$ to $\mathcal{T}_2\mathcal{T}_1 A$, and on functions, e.g. f to $\mathcal{T}_1 f$ and $\mathcal{T}_1 f$ to $\mathcal{T}_2\mathcal{T}_1 f$. A result about the compositionality of extensions shall be formulated.

Definition 4.17 We construct a category $E^{\rightarrow}$ based on the superposition category E. Maps $\mathcal{T}_1, \ldots, \mathcal{T}_n$ shall be endomaps on E:

- *Objects* are functions, including basic ones $f\colon A \to B$ or $g\colon B \to C$, ..., and extended ones $\mathcal{T}_1 f\colon \mathcal{T}_1 A \to \mathcal{T}_1 B$ or $\mathcal{T}_2 g\colon \mathcal{T}_2 B \to \mathcal{T}_2 C$, ...
- *Maps* on objects are triples of maps $(\phi_A, \phi_B, \mathcal{T}_F)$ which map the object $f\colon A \to B$ to $\mathcal{T}f\colon \mathcal{T}A \to \mathcal{T}B$ with domain mappings $\phi_A\colon A \to \mathcal{T}A$, $\phi_B\colon B \to \mathcal{T}B$, and a function lifting $\mathcal{T}_F\colon F \to \mathcal{T}F$ (F is the set of basic functions, $\mathcal{T}F$ the set of extended functions, etc.) such that $\mathcal{T}A$ preserves type A, $\mathcal{T}B$ preserves type B and $\mathcal{T}f$ preserves function f,
- The *identity* is $(\mathbf{1}_A, \mathbf{1}_B, \mathbf{1}_F)$,
- The *composite* of two maps $(\phi_{\mathcal{T}A}, \phi_{\mathcal{T}B}, \phi_{\mathcal{T}F}) \circ (\phi_A, \phi_B, \phi_F)$ is defined elementwise by $(\phi_{\mathcal{T}A} \circ \phi_A, \phi_{\mathcal{T}B} \circ \phi_B, \phi_{\mathcal{T}F} \circ \phi_F)$.

Maps in the category $E^{\rightarrow}$ are property-preserving liftings of functions. The definition states that the composition of property-preserving lifting of functions is associative.

Proposition 4.18 *The category $E^{\rightarrow}$ is well-defined.*

Proof. Identity and associativity of composition have to be shown.

- Identity: $(\mathbf{1}_A, \mathbf{1}_B, \mathbf{1}_F)\colon (f\colon A \rightarrow B) \mapsto (f'\colon A \rightarrow B)$ is the identity since we have function preservation $\mathbf{1}_B \circ f = f' \circ \mathbf{1}_A$ of f in f'.
- Associativity: The maps ϕ_A, ϕ_B are E-maps. Their composition is associative. F is the set of functions in E. Composition for maps in F is associative. Thus, the composition of function liftings $T_1, \ldots, \mathcal{T}_n$ is also associative.

□

5 Discussion

Applying our superposition operator results in a model presented in layers, each superimposing the layer below. The layers are specified using superposition and (possibly) augmentation. Redefinition with property preservation is captured by superposition. In order to provide a useful tool kit, so-called superposition schemes need to be introduced - essentially a library of common superpositions which have been obtained by applying our concepts to language semantics. Details about this in an earlier work can be found in [12].

Our approach compares to the application of monads in language semantics in that modular extensions of existing semantic models by new features are sought. Unlike Moggi's work [8] for modular language semantics, we do not assume a particular form of semantics (programs having computations as their semantics), thus we can provide a more general framework. Another framework, which is similar to ours, is Hoare and Jifeng's approach to linking theories in their unified theory of programming [4]. Our framework attempts to provide a similar tool kit for relating algebras. Refinement calculi for software development are well established [9]. We have presented a refinement approach for the stepwise development of algebraic models.

In the future, we plan to apply this framework. A promising application area is component technology. Component technology aims at reusing software through component libraries. Often, matching of requirements with services provided by a library component does not succeed. Support for automatic and semi-automatic adaptation can solve this problem. Adapting functionality of a library component to extended structural requirements can be facilitated using the techniques presented here.

Acknowledgements

We are grateful to the anonymous reviewers who have helped to improve the paper considerably.

References

[1] M. Barr and C. Wells. *Category Theory for Computing Science*. Prentice Hall, 1995. (second edition).

[2] P.M. Cohn. *Universal Algebra*. Harper and Row Publishers, 1965.

[3] G. Grätzer. *Universal Algebra*. D. van Nostrand Company, 1968.

[4] C.A.R. Hoare and H. Jifeng. *Unified Theories of Programming*. Prentice Hall, 1998.

[5] C.B. Jones. Data Reification. In J.A. McDermid, editor, *The Theory and Practice of Refinement*. Butterworths, 1989.

[6] C.B. Jones. *Systematic Software Development with VDM*. Prentice Hall, 1990.

[7] F.W. Lawvere and S. Schanuel. *Conceptual Mathematics*. Cambridge University Press, 1998.

[8] E. Moggi. Notions of Computation and Monads. *Information and Computation*, 93:55–92, 1991.

[9] C. Morgan. *Programming from Specifications 2e*. Addison-Wesley, 1994.

[10] C. Pahl. A Modular Development of the Denotational RSL Concurrency Model. Technical Report IT-TR:1997-016, Department of Information Technology, Technical University of Denmark, 1997.

[11] C. Pahl. Modular, Behaviour Preserving Extensions of the Unix C-shell Interpreter Language. Technical Report IT-TR:1997-014, Department of Information Technology, Technical University of Denmark, 1997.

[12] C. Pahl. Modular Composition of Language Features through Language Extensions. In A. Butterfield and S. Flynn, editors, *Proc. 3rd Irish Workshop on Formal Methods, July 1999, Galway, Ireland*, Electronic Workshops in Computing. Springer-Verlag, 1999.