Regular Paper

Δ-extension of Algebraic Specification

KAZUKI YOSHIDA*, AKIHIKO OHSUGA*, MORIO NAGATA** and SHINICHI HONIDEN*

This paper introduces *implication* and a particular symbol Δ into Algebraic Specification. Implications are used in the specification as conditional equations to simplify writing axioms of a complicated requirement. With respect to Δ, there are two ways of using this symbol. (1) As a symbol to specifiy the partiality of an operation in axioms. Substituting Δ for the annoying description of an exceptional behavior is a particular example of this. (2) As a symbol to mean temporarily undefined in the correspondence of operations in stepwise refinement. This can make it easy to refine a part of the specification and validate it by term rewriting system.

Next, a new sound and complete calculus is defined to guarantee logical consistency between implication and Δ. We can use this calculus to verify the correctness of stepwise refinement.

1. Introduction

A new programming paradigm based on formal specifications is being paid attention to by researchers and engineers of software production. Algebraic Specification is one of the promising formal specification methods to specify abstract data types. Stepwise refinement is often applied to Algebraic Specification in order to realize abstract data types. One of the merits in Algebraic Specification is that the correctness of each refinement can be verified formally [5].

The authors are now conducting research on a methodology to specify the users' requirement by Algebraic Specification. When we specify a large-scale and complex requirement, it is fairly difficult to write axioms only by equations. This is because the output of an operation to be specified often varies according to the input value. (To specify the partiality of an operation or an exceptional behavior is an example of this.) From this viewpoint, the authors think that the notion of a conditional equation is needed to simplify writing axioms. Therefore, this paper introduces implication into Algebraic Specification. The conclusion of implication consists of one equation, and the premise is a conjunction of some equations to specify the condition under which the conclusion holds.

Futhermore, a particular symbol Δ is introduced into Algebraic Specification in this paper. There are two ways of using this symbol.

1. Δ can be used as a symbol to specify the partiality of an operation in axioms. This enables us to formally specify only the pure properties in the core part of the requirement by substituting Δ for the annoying description of an exceptional behavior.
2. In stepwise refinement (which is explained by an example in Chapter 2), Δ can also be used as a symbol to mean temporarily undefined in the correspondence of operations between two succesive steps. This can make it easy to refine a part of the specification and validate it by term rewriting system.

The authors think for both 1 and 2 above that specifying partiality and temporarily undefined explicitly by Δ with conditions about the domain of an operation will contribute to a clear visual review of the specification.

In accordance with the introduction of implication and Δ, a new sound and complete calculus is defined to guarantee logical consistency between them. This calculus can be used to verify the correctness of the correspondence between two specifications with implication and Δ in stepwise refinement. (A formal discussion on this problem is presented in [10].)

The following is a brief survey on relevant researches.

With respect to specifying partial data types, there are many other researches. Here are some explanations of them. In [3], the notion of order sort is introduced, and the way to give specifications systematically on the behaviors in executing undefined operations is researched. In [8], inference rules are defined to deal with both equations and formulas. Formulas express that the term in the argument is evaluable. We can see whether a term is defined or not from whether the formula of the term is deducible or not. In [1], notions of partial initial semantics and weakly terminal semantics are introduced for full abstractness, and sufficient, syntactically checkable conditions for each existence are given. Moreover, the relation between them is discussed.

Introduction of conditional equations into Algebraic

*Toshiba Corporation Systems & Software Engineering Laboratory, 70, Yanagi-cho, Saiwai-ku, Kawasaki, Kanagawa 210, Japan.

**Department of Administration Engineering, Faculty of Science and Technology, Keio University, 3-14-1 Hiyoshi, Yokohama, 223, Japan.

Specification can be seen in [4], etc.

Stepwise refinement of specification, and verification of the correctness are explained in [5], [6] and [9].

The theoretical foundation on introducing implication and Δ into Algebraic Specification is presented in the main part of this paper.

The contents in each section are explained briefly as follows.

In Section 2, we show an example of stepwise refinement of specification with implication and Δ, and the verification of its correctness.

In Section 3, partial many sorted algebras are extended to imaginary total many sorted algebras by adding a particular element Δ, which means undefined, and other concepts are extended as well.

In Section 4, the symbol Δ and implication are introduced into the equational language. A new calculus is defined for these extensions and its soundness and completeness are proved.

In Section 5, conclusions are made on this research and the prospects for further research are mentioned.

Some necessary propositions for the proof of Proposition 5 are proved in the Appendix.

2. Example

In order to realize abstract data types, stepwise refinement is ofter applied to Algebraic Specification. One of the merits in Algebraic Specification is that the correctness of each refinement can be verified formally [5].

This section explains the stepwise refinement of specification, and the verification of its correctness in the authors' proposed framework.

Let us consider a specification of screen editor as an example. Refinement and varification in this example are as follows.

• Step 1

At first, we roughly specify all axioms which must be satisfied by the screen editor. The following is one of them.

$$delete(insert(E, C))=E \tag{1}$$

Here,

insert : *editor*, *character*→*editor*

(an operation which inserts one character into the screen editor)

delete : *editor*→*editor*.

(an operation which deletes one character from the screen editor)

So, the axiom (1) represents the general property of the screen editor.

• Step 2

Now, we decide to realize the screen editor by a pair of buffer and cursor-place.

Sorts in Step 1 are transformed as follows.

character↦*character*

editor↦*list*×(*nat*×*nat*)

(This correspondence can be imagined if we consider that *list* is a list of character strings which represents buffer, and that *nat*×*nat* represents a cursor-place by the row and column numbers.)

The following operations are prepared for this realization.

insert' : *string*, *nat*, *character*→*string*

(an operation which inserts one character between the N_{th} and $N+1_{\text{th}}$ characters of a string)

delete' : *string*, *nat*→*string*

(an operation which deletes the $N-1_{\text{th}}$ character from a string)

retrieve : *list*, *nat*→*string*

(an operation which retrieves the N_{th} element from a list)

overwrite : *list*, *nat*, *string*→*list*

(an operation which overwrites the N_{th} element of a list by a string)

The following axioms hold between these operations.

$$<(|L|, N)=true \text{ imply } overwrite(L, N, S)=\Delta \tag{2}$$

$$<(|L|, N)=true \text{ imply } retrieve(L, N)=\Delta \tag{3}$$

$$<(|L|, N)=false \text{ imply } retrieve(overwrite(L, N, S), N)=S \tag{4}$$

$$eq(C, \text{“}\langle CR\rangle\text{”})=false \text{ imply } delete'(insert'(S, N, C), N+1)=S \tag{5}$$

$$<(|L|, N)=false \text{ imply } overwrite(overwrite(L, N, S), N, retrieve(L, N))=L \tag{6}$$

Δ is a symbol to mean undefined.
$|L|$ returns the length of list L.
It is assumed that $<$ and *eq* are functions which have already been defined. $<$ returns *true* when the first argument is less than the second argument, otherwise *false*. *eq* returns *true* when the first argument is equal to the second argument, otherwise *false*.

Then, we consider the correspondence of operations between Step 1 and Step 2 on the basis of the requirement. At first, for each operation of Step 1, the partitioning of the input values is specified according to the difference in the output. For example, there is generally a difference in the output of *insert* according to whether the element of *character* in the input is $\langle CR\rangle$ or not.

This is written by using implications as follows.

$eq(C, \text{``}\langle CR\rangle\text{''})=true$ imply
$insert(E, C)=\Delta$

$eq(C, \text{``}\langle CR\rangle\text{''})=\neq false$ imply
$insert(E, C)=\Delta$

Δ is a temporary symbol to avoid specifying in detail.

Then, for each case partitioned above, the operations of Step 1 are expressed by those of Step 2 according to the requirement. But when the requirement for some case is vague, we can postpone specifying the case while we go into partial verification and further refinement only with a partial correspondence of the operations. As long as Δ remains in the correspondence of operations, we will be able to notice by visual review that the specification is incomplete, even when we are in a further refinement step.

With respect to the correspondence of operations, for example, let E be a vector $[L, Ro, Co]$, then

$eq(C, \text{``}\langle CR\rangle\text{''})=false$ imply
$insert([L, Ro, Co], C)$
$=[overwrite(L, Ro, insert'(retrieve(L, Ro), Co, C)), Ro, Co+1]$.

Moreover,

$eq(Co, 1)=false$ imply
$delete([L, Ro, Co])$
$=[overwrite(L, Ro, delete'(retrieve(L, Ro), Co)), Ro, Co-1]$.

We call this consideration of correspondence of sorts and operations between two specifications as in the above *refinement*.

• **Verification**

We verify that the above refinement is correct by a calculus which is newly defined in accordance with the introduction of implication and Δ. (Refer to Section 4 in which this calculus is defined.) That is to say, one of the necessary conditions for the correctness of refinement is that all the transformed axioms of Step 1 according to the correspondence are derivable from axioms of Step 2 by this calculus [10].

For example, (1) is expressed by the operations of Step 2 as follows.

$eq(C, \text{``}\langle CR\rangle\text{''})=false$ imply
$[overwrite(L', Ro, delete'(retrieve(L', Ro), Co+1)), Ro, Co]$
$=[L, Ro, Co]$
(Let L' be $overwrite(L, Ro, insert'(retrieve(L, Ro), Co, C))$.)

This can be derived from axioms (4), (5) and (6)

There is another condition for correctness, but this will not be argued here. (A formal discussion on verification is presented in [10].)

• **Step 3**

Operations such as *retrieve*, *overwrite*, *insert'*, and *delete'* defined in Step 2 are refined by *head* and *tail* which are primitive operations on the list. We omit the detail because the procedure is the same as in Step 2.

When a specification is refined step by step, the correctness of the final result with respect to the initial specification is guaranteed by verifying the correctness between each two successive steps [5].

3. Many Sorted Algebra

In this section, several notions of many sorted algebra are extended in accordance with the introduction of Δ. This extension is indispensable because Δ is a particular element which means undefined.

Fundamental terms *S-set*, *S-function* (*S-mapping*), *S-sorted system of variables*, and *assignment* used throughout this paper are defined in [4].

Furthermore, *signature*, *Σ-algebra* and *partial Σ-algebra* are defined in the same way as in [4]. $ALG_t(\Sigma)$ and $ALG(\Sigma)$ denote the class of all Σ-algebras and partial Σ-algebras, respectively.

For any $A\in ALG(\Sigma)$, *Δ-extension* A' is defined as follows. Δ is an element which means undefined.

Definition 1 Let $\Sigma=(S, \alpha:\Omega\to S^*\times S)$ be any signature and $A=((A_s|s\in S), (\sigma^A|\sigma\in\Omega))\in ALG(\Sigma)$.

Then A', defined as follows, is called the *Δ-extension* of A.

(1) For every $s\in S$,
$$A'_s=A_s\cup\{\Delta_s^{A'}\}.$$

(2) For every $\sigma\in\Omega(\sigma:\lambda\to s)$, ($\lambda$ is an empty string.)
$$\sigma^{A'}=\sigma^A.$$
(It is always assumed that σ is defined in A.)

(3) For every $\sigma\in\Omega(\sigma:w\to s)$ and $a_w\in A'^1_w$,
$$\sigma^{A'}(a_w)=\begin{cases}\sigma^A(a_w) & (\text{if } a_w\in dom\,\sigma^A)\\ \Delta_s^{A'} & (\text{otherwise}).\end{cases}$$ □

This is a strict extension, that is, for any $i\in\{1, \ldots, n\}=dom\ w$,

$$\sigma^{A'}(a_{w(1)}, \ldots, \Delta^{A'}_{w(i)}, \ldots, a_{w(n)})=\Delta_s^{A'}.$$

The following proposition holds with respect to a Δ-extension.

Proposition 1 For every $A\in ALG(\Sigma)$, the Δ-extension A' of A is a Σ-algebra.

(Proof) Let $A'=((A'_s|s\in S), (\sigma^{A'}|\sigma\in\Omega))$ be the Δ-extension of A.

For every $\sigma\in\Omega(\sigma:w\to s)$, its fundamental operation $\sigma^{A'}$ defined on an S-set $(A'_s|s\in S)$ is, for any $a_w\in A'_w$,

[1] Originally, w denotes a string of n sorts. (n is a natural number.) But sometimes we consider w to be a function from $\{1, \ldots, n\}$ to S. A'_w denotes $A'_{w(1)}\times\cdots\times A'_{w(n)}$.

$$\sigma^{A'}(a_w)=\sigma^A(a_w) \quad (\text{if } a_w\in dom\,\sigma^A)$$
$$\Delta_s^A \quad (\text{otherwise}).$$

$\Delta_s^A\in A_s'$ and $\sigma^A(a_w)\in A_s\subseteq A_s$, so

$$\sigma^{A'}: A_w'\to A_s'$$

and trivially this function is total.

Similarly, when $w=\lambda$, in particular.

Therefore, A' is a Σ-algebra. □

According to Proposition 1, for any $A\in ALG(\Sigma)$, its Δ-extension A' belongs to $ALG_t(\Sigma)$. Therefore we can make a Δ-extension recursively because $ALG_t(\Sigma)$ is a subset of $ALG(\Sigma)$. But we prohibit this by the next definition.

Definition 2 Let $A\in ALG(\Sigma)$ and A' be the Δ-extension of A.

Then, the Δ-extension of A' is A' itself. □

Let $ALG'(\Sigma)$ denote the set of all Δ-extensions made from every partial Σ-algebra in accordance with Definition 1 and Definition 2. After this, this set, $ALG'(\Sigma)$, is always considered in the discussion of semantics.

Σ-homomorphism is defined between two arbitrary Σ-algebras of $ALG'(\Sigma)$.

Definition 3 For A, $B\in ALG'(\Sigma)$, an S-function $f=(F_s: A_s\to B_s | s\in S)$ is said to be a *Σ-homomorphism* from A to B, denoted by $f: A\to B$, if it satisfies the following conditions:

(1) For every $s\in S$,

$$f_s(\Delta_s^A)=\Delta_s^B.$$

(2) For every $\sigma\in\Omega(\sigma:\lambda\to s)$,

$$f_s(\sigma^A)=\sigma^B.$$

(3) For every $\sigma\in\Omega(\sigma:w\to s)$, any $a_w\in A_w$,

$$f_s(\sigma^A(a_w))=\sigma^B(f_w(a_w)).$$

A Σ-homomorphism $f:A\to B$ is called a *Σ-isomorphism*, if there is a Σ-homomorphism $g:B\to A$ with

$$f_s\circ g_s=Id_{A_s},\quad g_s\circ f_s=Id_{B_s} \quad \text{for every } s\in S.$$

Id_{A_s} denotes the identity mapping on A_s.

A Σ-homomorphism $f:A\to B$ is called *surjective* (*injective*) if $f_s:A_s\to B_s$ is surjective (injective) for every $s\in S$. □

We can say that a Σ-isomorphism is a bijective Σ-homomorphism.

The next proposition holds with respect to a Σ-homomorphism. This fact is used in the proof of later propositions.

Proposition 2 Let $\Sigma=(S, \alpha:\Omega\to S^*\times S)$ be any signature and A, B, $C\in ALG'(\Sigma)$.

A composition of two Σ-homomorphisms $f:A\to B$ and $g:B\to C$ is defined, denoted by

$$g\circ f: A\to C$$

as $g_s\circ f_s(a)=g_s(f_s(a))$ for every $s\in S$ and any $a\in A_s$.

Then, this composition is again a Σ-homomorphism. Moreover, this is associative.

(Proof) At first, we show that $g\circ f$ is a Σ-homomorphism.

(1) For every $s\in S$,

$$g_s\circ f_s(\Delta_s^A)=g_s(f_s(\Delta_s^A))=g_s(\Delta_s^B)=\Delta_s^C.$$

(2) For every $\sigma\in\Omega(\sigma:\lambda\to s)$,

$$g_s\circ f_s(\sigma^A)=g_s(f_s(\sigma^A))=g_s(\sigma^B)=\sigma^C.$$

(3) For every $\sigma\in\Omega$ $(\sigma:w\to s)$ and any $(a_1, \ldots, a_n)\in A_w$,

$$g_s\circ f_s(\sigma^A(a_1, \ldots, a_n))$$
$$=g_s(f_s(\sigma^A(a_1, \ldots, a_n)))$$
$$=g_s(\sigma^B(f_{w(1)}(a_1), \ldots, f_{w(n)}(a_n)))$$

($w(i)$ denotes the i_{th} element of sorts string w.)

$$=\sigma^C(g_{w(1)}(f_{w(1)}(a_1)), \ldots, g_{w(n)}(f_{w(n)}(a_n)))$$
$$=\sigma^C(g_{w(1)}\circ f_{w(1)}(a_1), \ldots, g_{w(n)}\circ f_{w(n)}(a_n)).$$

From (1), (2), (3) above, we can conclude that $g\circ f$ is a Σ-homomorphism.

Associativity follows from the fact that a Σ-homomorphism is a function. □

A *Σ-term* is defined as follows in accordance with the introduction of Δ.

Definition 4 Let $\Sigma=(S, \alpha:\Omega\to S^*\times S)$ be any signature and $v:X\to S$ be an S-sorted system of variables. Then,

$$T(\Sigma, X)=(T(\Sigma, X)_s | s\in S)$$

in an S-set which is formed according to the following 1 and 2.

1. $T^*(\Sigma, X)$ is the smallest S-set which is defined inductively by using X and Ω as follows.

(1) For every $x\in X$,

$$x\in T^*(\Sigma, X)_{v(x)}.$$

(2) For every $\sigma\in\Omega(\sigma:\lambda\to s)$,

$$\sigma\in T^*(\Sigma, X)_s.$$

(3) For every $\sigma\in\Omega(\sigma:w\to s)$, if $(t_1, \ldots, t_n)\in T^*(\Sigma, X)_w$ then

$$\sigma(t_1, \ldots, t_n)\in T^*(\Sigma, X)_s.$$

2. For every $s\in S$,

$$T(\Sigma, X)_s=T^*(\Sigma, X)_s\cup\{\Delta_s\}.$$

Δ_s is called a *symbol of undefined*.

An element of $T(\Sigma, X)$ is called a *Σ-term* on v. □

In the above definition, especially when $X=\phi$, the S-set is denoted by $T(\Sigma)$, and an element of $T(\Sigma)$ is called a *free Σ-term*.

Σ-term algebra is defined using $T(\Sigma, X)$ above.

Definition 5 Let $\Sigma=(S, \alpha: \Omega\to S^*\times S)$ be any signature and $v:X\to S$ be an S-sorted system of variables. Then Σ-algebra A, defined as follows, is called a *Σ-term algebra* on v.

(1) For every $s\in S$,

$$A_s = T(\Sigma, X)_s.$$

(2) For every $\sigma \in \Omega$ $(\sigma:\lambda \to s)$,

$$\sigma^A = \sigma.$$

(3) For every $\sigma \in \Omega$ $(\sigma:w \to s)$ and any $(t_1, \ldots, t_n) \in A_w$,

$$\sigma^A(t_1, \ldots, t_n) = \Delta_s \quad \text{if} \quad (\exists t_i \in A_{s'})(t_i = \Delta_{s'})$$
$$\sigma(t_1, \ldots, t_n) \quad \text{otherwise.}$$

This Σ-term algebra A is denoted by $T(\Sigma, X)$ (the same symbol as the set of Σ-terms). □

It is needless to say that $T(\Sigma, X) \in ALG'(\Sigma)$.

In the above definition, especially when $X = \phi$, $T(\Sigma, \phi)$ is called a *free Σ-term algebra* denoted by $T(\Sigma)$, which has a very important role in considering the semantics of specifications.

By the way, properties of all Σ-terms will be proved by considering two cases.

(1) Do the properties hold for Δ?

(2) Do the properties hold for all Σ-terms except Δ?

Specifically, a *structural induction* can be used to prove (2).

4. Equational Logic

Ordinary equational logic is extended in this section, which is an important part of Algebraic Specification. That is, implications are introduced into Algebraic Specification, which are used as conditional equations. Accordingly, a new calculus is defined which copes with implications and Δ, and its soundness and completeness are proved, so that logical consistency between implications and Δ is guaranteed.

At first, language symbols will be defined.

Definition 6 Let $\Sigma = (S, \alpha:\Omega \to S^* \times S)$ be any signature and X be a set of variables. A set of language symbols, denoted by $L(\Sigma, X)$, consists of the following.

(1) all variable symbols $x \in X$
(2) undefinition symbol Δ_s for every $s \in S$
(3) all constant symbols $\sigma \in \Omega$
(4) all operation symbols $\sigma \in \Omega$
(5) parenthesis ( )
(6) a comma ,
(7) an equality symbol $\approx$
(8) an implication symbol $\supset$ □

Next, *equation* and *(elementary) implication* of $L(\Sigma, X)$ are defined by using $T(\Sigma, X)$ in the same way as in [4]. However, the semantics of these formulas are changed in accordance with the introduction of Δ. This will be defined later.

By the way, equational logic can be interpreted as a many sorted algebra, because it is a first order logic whose logical symbols and predicate symbols except equality are excluded. So, *extended assignment* are defined on the basis of *assignment*. With respect to assignments, those which assign Δ to a varialbe are left out of consideration. Let $v:X \to S$ be an S-sorted system of variables. The set of assignments of elements in an S-set $A' = (A_s \cup \{\Delta_s^A\} \mid s \in S)$ to v is denoted by A'_v.

Definition 7 Let $\Sigma = (S, \alpha:\Omega \to S^* \times S)$ be any signature, $v:X \to S$ be any S-sorted system of variables, and $A \in ALG'(\Sigma)$. Then for any $\mathbf{a} \in A_v$, *extended assignment* of $\mathbf{a}$ on A is a Σ-homomorphism $\mathbf{a}^{ex}:T(\Sigma, X) \to A$, defined as follows.

(1) For every $x \in X$,

$$\mathbf{a}^{ex}_{v(x)}(x) = a_{v(x)}(x).$$

(2) For every $s \in S$,

$$\mathbf{a}^{ex}_s(\Delta_s) = \Delta_s^A.$$

(3) For every $\sigma \in \Omega(\sigma:\lambda \to s)$.

$$\mathbf{a}^{ex}_s(\sigma) = \sigma^A.$$

(4) For every $\sigma \in \Omega(\sigma:w \to s)$ and any $t_w \in T(\Sigma, X)_w$,

$$\mathbf{a}^{ex}_s(\sigma(t_w)) = \sigma^A \mathbf{a}^{ex}_w(t_w).$$ □

Especially, when $X = \phi$ in the above definition, extended assignment is called *evaluation*.

The next two propositions concerned with the above two notions explain a universal property of $T(\Sigma, X)$ and $T(\Sigma)$.

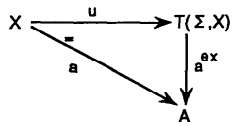
Proposition 3 $T(\Sigma, X)$ is a *free algebra* [2] over X in $ALG'(\Sigma)$.

(Proof)

1. It is clear that $T(\Sigma, X) \in ALG'(\Sigma)$.
2. At first, mapping $\mathbf{u}:X \to T(\Sigma, X)$ is defined as

$$\mathbf{u}(x) = x.$$

Then, it is shown that for any $A \in ALG'(\Sigma)$ and any $\mathbf{a} \in A_v(v:X \to s)$, its extended assignment $\mathbf{a}^{ex}$: $T(\Sigma, X) \to A$ is a unique Σ-homomorphism which makes the following diagram of mappings commutative, i.e. $\mathbf{a}^{ex} \circ \mathbf{u} = \mathbf{a}$.



The diagram in Proposition 3

It is clear from Definition 7 that $\mathbf{a}^{ex}$ is a Σ-homomorphism and $\mathbf{a} = \mathbf{a}^{ex} \circ \mathbf{u}$. So, it remains to show that $\mathbf{a}^{ex}$ is the only Σ-homomorphism. Assume that there is another Σ-homomorphism

$$h:T(\Sigma, X) \to A$$

which satisfies $h \circ \mathbf{u} = \mathbf{a}$. It can be shown by structural induction that $h = \mathbf{a}^{ex}$.

(1) For all constant symbols $\sigma \in T(\Sigma, X)$,

$$\mathbf{a}^{ex}(\sigma) = \sigma^A = h(\sigma).$$

For all variables $x \in T(\Sigma, X)$,

$$\mathbf{a}^{ex}(x) = a(x) = h(x).$$

For all undefinition symbols $\Delta_s \in T(\Sigma, X)$,

$$\mathbf{a}^{ex}(\Delta_s)=\Delta_s^A=h(\Delta_s).$$

(2) Assume that $\sigma(t_1, \ldots, t_n)\in T(\Sigma, X)$ and $\mathbf{a}^{ex}(t_i)=h(t_i)$ for $i=1, \ldots, n$, then

$$\begin{aligned}\mathbf{a}^{ex}(\sigma(t_1, \ldots, t_n)) &= \sigma^a(a^{ex}(t_1), \ldots, a^{ex}(t_n)) \\ &= \sigma^a(h(t_1), \ldots, h(t_n)) \\ &= h(\sigma(t_1, \ldots, t_n))\end{aligned}$$

From (1) and (2) above, $h=\mathbf{a}^{ex}$. This shows that $\mathbf{a}^{ex}$ is the only Σ-homomorphism. □

The above fact is used in the proof of later propositions.

The next proposition holds as a special case, $X=\phi$, in Proposition 3.

Proposition 4 $T(\Sigma)$ is a free algebra over ϕ in $ALG'(\Sigma)$. □

This proposition about $T(\Sigma)$ is important when the semantics of specifications is considered.

Next, *tautological* elementary implications will be defined before a discussion is made on a new calculus which can cope with implications and Δ.

Definition 8 Let $\Sigma=(S, \alpha:\Omega\to S^*\times S)$ be any signature. For every $s\in S$, let $v_s:\{x, y, z\}\to S$ be the S-sorted system of variables with $v_s(x)=v_s(y)=v_s(z)=s$. Then, the following elementary implications are called *tautological*:

REF_s: $(v_s;\ \phi\supset x\approx x)$,
$(v_s;\ \phi\supset\Delta_s\approx\Delta_s)$. ($\phi$ is an empty set)

SYM_s: $(v_s;x\approx y\supset y\approx x)$,
$(v_s;\ \Delta_s\approx x\supset x\approx\Delta_s)$,
$(v_s;\ x\approx\Delta_s\supset\Delta_s\approx x)$.

TRA_s: $(v_s;\ x\approx y,\ y\approx z\supset x\approx z)$,
$(v_s;\ \Delta_s\approx y,\ y\approx z\supset\Delta_s\approx z)$,
$(v_s;\ x\approx y,\ y\approx\Delta_s\supset x\approx\Delta_s)$.

For every $\sigma\in\Omega$ with $\sigma:w\to s$ (let w be a string of n sorts.) and every $j\in\{1, \ldots, n\}$, an S-sorted system of variables is defined

$$v_{\sigma,j}:\{x_1, \ldots, x_n, y, z\}\to S$$

by $v_{\sigma,j}(x_i)=w(i)$ for every $i\in\{1, \ldots, n\}$, and $v_{\sigma,j}(y)=w(j)$, $v_{\sigma,j}(z)=s$. Further tautological elementary implications are defined with this S-sorted system of variables:

$REP_{\sigma,j}$: $(v_{\sigma,j};\ x_j\approx y,\ z\approx\sigma(x_1, \ldots, x_n)\supset z\approx\sigma(x_1, \ldots, x_{j-1}, y, x_{j+1}, \ldots, x_n))$,
$(v_{\sigma,j};\ x_j\approx\Delta_{w(j)},\ z\approx\sigma(x_1, \ldots, x_n)\supset z\approx\Delta_s)$.

Finally,

$$\begin{aligned}TAUT(\Sigma) &= \{REF_s\mid s\in S\}\cup\{SYM_s\mid s\in S\}\cup\{TRA_s\mid s\in S\}\cup \\ &\quad \{REP_{\sigma,j}\mid\sigma\in\Omega,\ \sigma:w\to s,\ j=1, \ldots, n\}\end{aligned}$$

is called the set of *tautological* elementary implications of Σ. □

Strictness of Δ-extension is taken into account in the definition of $REP_{\sigma,j}$. Calculus which can cope with implications and Δ is defined by using $TAUT(\Sigma)$.

Definition 9 An equation $(v;\ t_1\approx t_2)$ with $v:X\to S$ is said to be *$\mathscr{A}$-derivable* from a finite set of equations $(v;\ G)$, denoted by

$$\mathscr{A}\vdash(v;\ G\supset t_1\approx t_2)$$

where $\mathscr{A}$ is any set of elementary implications, if it is derivable using only the rules of derivation **D1** and **D2** below:

D1 For every $(v;\ t\approx t')\in(v;\ G)$,

$$\mathscr{A}\vdash(v;\ G\supset t\approx t').$$

D2 If

(a) $(u;\ H\supset t\approx t')\in TAUT(\Sigma)\cup\mathscr{A}$ with $u:Y\to S$,

(b) there is an assignment $\mathbf{a}\in T(\Sigma, X)_u$, such that

$$\mathscr{A}\vdash(v;\ G\supset\mathbf{a}^{ex}(p)\approx\mathbf{a}^{ex}(p'))$$
for every $(u;\ p\approx p')\in(u;\ H)$,

it follows that

$$\mathscr{A}\vdash(v;\ G\supset\mathbf{a}^{ex}(t)\approx\mathbf{a}^{ex}(t')).$$

$IMP(\mathscr{A})$ denotes the set of all elementary implications $(v;\ G\supset t\approx t')$ with $\mathscr{A}\vdash(v;\ G\supset t\approx t')$. □

For any implication $(v;\ G\supset H)$,

$$\mathscr{A}\vdash(v;\ G\supset H)$$

is defined as

$$\mathscr{A}\vdash(v;\ G\supset t\approx t')$$

for every $(v;\ t\approx t')\in(v;\ H)$.

Now, the other part, semantics of the formulas, will be defined.

Definition 10 Let $(v;\ t\approx t')$ be an equation of $L(\Sigma, X)$ on $v:X\to S$ and $A\in ALG'(\Sigma)$. Then, an assignment $\mathbf{a}\in A_v$ is called a *solution* of $(v;\ t\approx t')$ in A, denoted by $(A, \mathbf{a})\models(v;\ t\approx t')$, if it satisfies the following conditions (assuming that $t, t'\in T(\Sigma, X)_s$):

1. If $t\neq\Delta_s$ and $t'\neq\Delta_s$,

$$\mathbf{a}^{ex}(t)\neq\Delta_s^A,\quad \mathbf{a}^{ex}(t')\neq\Delta_s^A,\quad \text{and}\quad \mathbf{a}^{ex}(t)=\mathbf{a}^{ex}(t').$$

2. If $t=\Delta_s$ or $t'=\Delta_s$,

$$\mathbf{a}^{ex}(t)=\mathbf{a}^{ex}(t')=\Delta_s^A.$$

A set of solutions of $(v;\ t\approx t')$ in A is denoted by $A_{(v;t\approx t')}$.

Let $(v;\ G)$ be a set of equations of $L(\Sigma, X)$ on $v:X\to S$. Then, a set of solutions of $(v;\ G)$ in A, denoted by $A_{(v;G)}$, is defined as

$$\{\mathbf{a}\in A_v\mid \text{for all } (v;\ t\approx t')\in(v;\ G),\ (A, \mathbf{a})\models(v;\ t\approx t')\}.$$

□

Definition 11 Let $(v;\ G\supset t\approx t')$ be an elementary implication of $L(\Sigma, X)$ on $v:X\to S$ and $A\in ALG'(\Sigma)$. If $A_{(v;G)}\subseteq A_{(v;t\approx t')}$, then $(v;\ G\supset t\approx t')$ is said to *be satisfied* (or *hold*) in A, denoted by

$$A \models (v;\ G \supset t \approx t').$$

Let $(v;\ G \supset H)$ be an implication of $L(\Sigma, X)$ on $v:X \to S$. If $A_{(v;\ G)} \subseteq A_{(v;H)}$, then $(v:\ G \supset H)$ is said to *be satisfied* (or *hold*) in A, denoted by

$$A \models (v;\ G \supset H).$$

Let $\mathscr{A}$ be a set of (elementary) implications of Σ and ξ an (elementary) implication of Σ. If ξ is satisfied by all Δ-extensions which satisfy $\mathscr{A}$, then ξ is called a *consequence* of $\mathscr{A}$, denoted by $\mathscr{A} \models \xi$. □

Then, the calculus in Definition 9 is proved to be sound and complete by using induction on the representation of $IMP(\mathscr{A})$ and some propositions in the Appendix.

Proposition 5 Let $\Sigma = (S, \alpha:\Omega \to S^* \times S)$ be any signature, $\mathscr{A}$ any set of elementary implications of Σ, and $(v;\ G \supset H)$ any implication of $L(\Sigma, X)$ on $v:X \to S$. Then,

[1] $\mathscr{A} \vdash (v;\ G \supset H)$ implies $\mathscr{A} \models (v;\ G \supset H)$.

[2] $\mathscr{A} \models (v;\ G \supset H)$ implies $\mathscr{A} \vdash (v; G \supset H)$.

(Proof)

[1] First, it will be shown that $\mathscr{A}$-derivability is sound for elementary implications.

The proof is based on the inductive representation

$$IMP(\mathscr{A}) = \bigcup_{i=0}^{\infty} IMP_i(\mathscr{A}). \quad \text{(See Appendix)}$$

(1) Let $(r;\ G \supset t \approx t') \in IMP_0(\mathscr{A})$ with $r:Y \to S$. From the definition of $IMP_0(\mathscr{A})$, either $(r;\ t \approx t') \in (r;\ G)$ **(Case 1)**, or $t = t' = y \in Y$ **(Case 2)**.

Case 1 Because $A_{(r;G)} = \cap\{A_{(r;t \approx t')} \mid (r;\ t \approx t') \in (r;\ G)\}$,

$$A_{(r;G)} \subseteq A_{(r;t \approx t')}$$

holds for every $(r;\ t \approx t') \in (r;\ G)$.

Case 2 Because $(r;\ G)$ is a set of equations on $r:Y \to S$,

$$A_{(r;G)} \subseteq A_r = A_{(r;y \approx y)}$$

holds for every $y \in Y$.

(2) Now, it is assumed that this proposition is true for every elementary implication of $IMP_i(\mathscr{A})$ $(i \geq 0)$. If

$$(r;\ G \supset t \approx t') \in IMP_{i+1}(\mathscr{A})$$

then, there are

$$(u;\ H \supset m \approx m') \in TAUT(\Sigma) \cup \mathscr{A}$$

with $u:Z \to S$ and $\mathbf{t} \in T(\Sigma, Y)_u$

such that condition (b) of **D2** is satisfied with respect to $(r;\ G)$, that is,

$$(r;\ G \supset \mathbf{t}^{ex}(p) \approx \mathbf{t}^{ex}(p')) \in IMP_i(\mathscr{A})$$

holds for every $(u;\ p \approx p') \in (u;H)$, and $\mathbf{t}^{ex}(m) = t$, $\mathbf{t}^{ex}(m') = t'$. Now, let $\mathbf{a} \in A_{(r;G)}$. Then, we have to show that

$$\mathbf{a} \in A_{(r;t \approx t')}.$$

Consider an assignment $\mathbf{a}^{ex} \circ \mathbf{t}:Z \to A$. Then, for any $m \in T(\Sigma, Z)$,

$$(\mathbf{a}^{ex} \circ \mathbf{t})^{ex}(m) = \mathbf{a}^{ex}(\mathbf{t}^{ex}(m))$$

because $T(\Sigma, Z)$ is a free algebra.

For every $(u;\ p \approx p') \in (u;\ H)$,

$$\mathbf{a} \in A_{(r;\mathbf{t}^{ex}(p) \approx \mathbf{t}^{ex}(p'))}$$

because

$$(r;\ G \supset \mathbf{t}^{ex}(p) \approx \mathbf{t}^{ex}(p')) \in IMP_i(\mathscr{A}).$$

Here, two cases will be considered.

Case 1 $p = \Delta$, or, $p' = \Delta$ (hereafter, the sort index is omitted)

$$t^{ex}(p) = \Delta, \quad \text{or,} \quad t^{ex}(p') = \Delta$$

Since $\mathbf{a} \in A_{(r;\mathbf{t}^{ex}(p) \approx \mathbf{t}^{ex}(p'))}$,

$$\mathbf{a}^{ex}(\mathbf{t}^{ex}(p)) = \mathbf{a}^{ex}(t^{ex}(p')) = \Delta^A,$$

that is,

$$(\mathbf{a}^{ex} \circ \mathbf{t})^{ex}(p) = (\mathbf{a}^{ex} \circ \mathbf{t})^{ex}(p') = \Delta^A.$$

So, $\mathbf{a}^{ex} \circ \mathbf{t}$ is a solution of $(u;\ p \approx p')$ in A.

Case 2 $p \neq \Delta$ and $p' \neq \Delta$

$$\mathbf{t}^{ex}(p) \neq \Delta \quad \text{and} \quad \mathbf{t}^{ex}(p') \neq \Delta.$$

Since $\mathbf{a} \in A_{(r;\mathbf{t}^{ex}(p) \approx \mathbf{t}^{ex}(p'))}$,

$$\mathbf{a}^{ex}(\mathbf{t}^{ex}(p)) \neq \Delta^A, \quad \mathbf{a}^{ex}(\mathbf{t}^{ex}(p')) \neq \Delta^A$$

and

$$\mathbf{a}^{ex}(\mathbf{t}^{ex}(p)) = \mathbf{a}^{ex}(\mathbf{t}^{ex}(p')).$$

We can rewrite these as follows:

$$(\mathbf{a}^{ex} \circ \mathbf{t})^{ex}(p) \neq \Delta^A, \quad (\mathbf{a}^{ex} \circ \mathbf{t})^{ex}(p') \neq \Delta^A$$

and

$$(\mathbf{a}^{ex} \circ \mathbf{t})^{ex}(p) = (\mathbf{a}^{ex} \circ \mathbf{t})^{ex}(p').$$

This means $\mathbf{a}^{ex} \circ \mathbf{t}$ is a solution of $(u;\ p \approx p')$ in A.

From **Case 1** and **Case 2** above, we can conclude that

$$\mathbf{a}^{ex} \circ \mathbf{t} \in A_{(u;H)}.$$

Since $(u;\ H \supset m \approx m') \in TAUT(\Sigma) \cup \mathscr{A}$,

$$\mathbf{a}^{ex} \circ \mathbf{t} \in A_{(u;m \approx m')}.$$

Here, two cases will be considered again.

Case 1 $m = \Delta$, or, $m' = \Delta$

$$t = \mathbf{t}^{ex}(m) = \Delta, \quad \text{or,} \quad t' = \mathbf{t}^{ex}(m') = \Delta$$

Since $\mathbf{a}^{ex}(\mathbf{t}^{ex}(m)) = \mathbf{a}^{ex}(\mathbf{t}^{ex}(m')) = \Delta^A$,

$$\mathbf{a}^{ex}(t) = \mathbf{a}^{ex}(t') = \Delta^A.$$

So, $\mathbf{a}$ is a solution of $(r;\ t \approx t')$ in A.

Case 2 $m \neq \Delta$ and $m' \neq \Delta$

$$t = \mathbf{t}^{ex}(m) \neq \Delta \quad \text{and} \quad t' = \mathbf{t}^{ex}(m') \neq \Delta$$

Since $\mathbf{a}^{ex} \circ \mathbf{t} \in A_{(u;m \approx m')}$,

$$\mathbf{a}^{ex}(t) = \mathbf{a}^{ex}(\mathbf{t}^{ex}(m)) \neq \Delta^A,$$

$$\mathbf{a}^{ex}(t') = \mathbf{a}^{ex}(\mathbf{t}^{ex}(m')) \neq \Delta^A,$$

and

$$\mathbf{a}^{ex}(t)=\mathbf{a}^{ex}(\mathbf{t}^{ex}(m))=\mathbf{a}^{ex}(\mathbf{t}^{ex}(m'))=\mathbf{a}^{ex}(t').$$

This means **a** is a solution of $(r; t\approx t')$ in A.
From **Case 1** and **Case 2** above, we can conclude that

$$\mathbf{a}\in A_{(r;t\approx t')}.$$

From (1) and (2) above, it has been proved that $\mathscr{A}$-derivability is sound for elementary implications.

Next, it will be shown that $\mathscr{A}$-derivability is sound for implications. Let $(v; G\supset H)$ with $v:X\to S$ be any implication with $\mathscr{A}\vdash(v; G\supset H)$, and let A be any *$\mathscr{A}$-algebra*. (*$\mathscr{A}$-algebra* means a Δ-extension which satisfies all elementary implications of $\mathscr{A}$.) We have to prove that

$$\mathbf{a}\in A_{(v;t\approx t')} \quad \text{results from} \quad \mathbf{a}\in A_{(v;G)}$$

for every $(v; t\approx t')\in(v; H)$. $\mathscr{A}\vdash(v; G\supset H)$ means

$$\mathscr{A}\vdash(v; G\supset t\approx t')$$

for every $(v; t\approx t')\in(v; H)$. The soundness for elementary implications guarantees

$$\mathscr{A}\models(v; G\supset t\approx t'),$$

so that,

$$\mathbf{a}\in A_{(v;G)}\subseteq A_{(v;t\approx t')}.$$

The above has proved that $\mathscr{A}$-derivability is sound for implications.

[2] Assume that $A_{(v;G)}\subseteq A_{(v;H)}$ for any $\mathscr{A}$-algebra A. Then,

$$F(\mathscr{A}, G, v)_{(v;G)}\subseteq F(\mathscr{A}, G, v)_{(v;H)}$$

because $F(\mathscr{A}, G, v)$ is an $\mathscr{A}$-algebra. (See Definition A.2 and Proposition A.4.)

Now let us define $\mathbf{i}\in T(\Sigma, X)_v$ with $v:X\to S$ by, for every $x\in X$,

$$\mathbf{i}(x)=x,$$

then, from Proposition A.3,

$$[\mathbf{i}, G]\in F(\mathscr{A}, G, v)_{(v;H)} \quad \text{iff} \quad \mathscr{A}\vdash(v; G\supset\mathbf{i}^{ex}(H)).$$

Since $[\mathbf{i}, G]\in F(\mathscr{A}, G, v)_{(v;G)}$,

$$[\mathbf{i}, G]\in F(\mathscr{A}, G, v)_{(v;H)}.$$

Moreover, since $\mathbf{i}(x)$ is an identity map on $T(\Sigma, X)$, we can conclude that

$$\mathscr{A}\vdash(v; G\supset H). \qquad \square$$

5. Conclusion

This paper has introduced implication and Δ into Algebraic Specification as a starting point to establish a methodology to specify the users' requirement by Algebraic Specification. Thanks to the introduction of implication, we can write a conditional equation directly in the specification from a case-oriented analysis. Moreover, Δ enables us to specify that operations are partial or temporarily undefined. These contribute to producing bugless specifications in large-scale and complex system developments.

The results in this paper are briefly summarized as follows.

At first, the Δ-extension of the partial Σ-algebra was defined. It has been shown that a Δ-extended Σ-term algebra is a free algebra in the class of all Δ-extensions. This guarantees that there is only one extended assignment for any assignment.

Next, a set of tautological implications and derivation rules were newly defined. The strictness of Δ-extension was considered in this definition. In the end, it has been proved that the calculus is sound and complete, which is the main result of this research.

From the above results, it can be seen that there is no theoretical problem in using Δ in the calculus of implication.

As a final topic, the prospects for further research are considered.

Currently, research is being conducted on the verification of correctness in the stepwise refinement of the specification with implication and Δ, using the new calculus in this paper. Several useful propositions have been proved, which are presented in [10].

Formal error checking of specification units produced in stepwise refinement can be considered. For example, when the transformation of operations are specified, it is possible to check formally whether the partitioning of input values according to the requirement is exclusive or not, and moreover, exhaustive or not [7]. The authors are planning to implement these algorithms and carry out experiments in specifying real systems.

We can further consider the transformation algorithm from a direct execution system of this extended calculus into logic programs. If this algorithm is implemented, executable codes can be automatically generated from Algebraic Specification with implications and Δ. In this research, we can refer to the transformation algorithm from a regular term rewriting system into a parallel logic program which the authors have proposed in another paper [11].

Acknowledgements

The authors would like to thank TOSHIBA Systems and Software Engineering Laboratory Director Mr. Nishijima, and Senior Manager Mr. Ohfude for providing them with the opportunity to carry out this research. The authors also express their thanks to anonymous referees for valuable comments. In addition, the authors wish to thank Mr. Umibe for reviewing and giving valuable comments to the original English manuscript.

Reference

1. BROY, M. and WIRSING, M. Partial abstract types. *Acta Informatica*, **18** (1982), 47–64.
2. EHRIG, H. and MAHR, B. Fundamentals of Algebraic Specification 1, Equations and Initial Semantics. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo (1985).
3. GOGUEN, J. A. and WINKLER, T. Introducing OBJ3. Technical report, SRI International (1989).
4. HORST, R. Initial Computability, Algebraic Specifications, and Partial Algebras, Clarendon Press Oxford (1987).
5. INAGAKI, Y. and SAKABE, T. Abstract Data Types, *IPS Japan*, **25** (5) (1984), 491–501.
6. MAIBAUM, T. S. E., VELOSO, P. A. S. and SADLER, M. R. A Theory of Abstract Data Types for Program Development, *In Int. Joint Conf. on Theory and Practice of Software Development* (1985), 214–230.
7. NAGATA, M. An Approach to Construction of Functional Programs, *J. Inf. Process.* **5**(4) (1982), 231–238.
8. SAKABE, T., INAGAKI, Y. and HONDA, N. Specification of abstract data type with partially defined operations, *In 6th Int. Conf. on Softw. Eng.* (1982), 218–224.
9. TURSKI, W. M. and MAIBAUM, T. S. E. The Specification of Computer Programs, ADDISON-WESLEY PUBLISHING COMPANY (1987).
10. YOSHIDA, K. OHSUGA, A. and NAGATA, M. Stepwise Refinement and its Correctness in Δ-extension (in preparation).
11. YOSHIDA, K., OHSUGA, A., YAMAMOTO, J. and HONIDEN, S. A Program Transformation from Term Rewriting System into Parallel Logic Program (in preparation).

(Received February 4, 1991; revised September 5, 1991)

A Appendix

Another description of the set $IMP(\mathscr{A})$ will be given here. The rule of derivation **D2** is applicable only if there are any $\mathscr{A}$-derivable elementary implications satisfying (a) and (b). On the other hand, one can derive elementary implications without any prerequisites by means of **D1**. Therefore, we denote by

$$IMP_0(\mathscr{A})$$

the set of all those elementary implications that are $\mathscr{A}$-derivable only by means of **D1**. This means

$$IMP_0(\mathscr{A})=\{(v;\ G\supset t\approx t')\mid(v;\ t\approx t')\in G\}.$$

Since **D2** is the rule of derivation that can be applied iteratively, we set

$IMP_{i+1}(\mathscr{A})=IMP_i(\mathscr{A})$ joined with the set of all elementary implications that are $\mathscr{A}$-derivable by **D2** and $IMP_i(\mathscr{A})$.

Thus, $IMP_i(\mathscr{A})$ is the set of all elementary implications which are $\mathscr{A}$-derivable by i-fold application of the rule **D2**. This leads to

$$IMP(\mathscr{A})=\cup_{i=0}^{\infty} IMP_i(\mathscr{A}).$$

Proposition A.1 Let $\mathscr{A}$ be any set of elementary implications of Σ and $v:X\to S$ be an S-sorted system of variables. Then, for every $s\in S$ and every $r\in T(\Sigma, X)_s$,

$$\mathscr{A}\vdash(v;\ G\supset r\approx r).$$

(Proof) Two cases will be considered.

(1) $r\in T(\Sigma, X)_s-\{\Delta_s\}$

If we take $(v;\ \phi\supset x\approx x)\in TAUT(\Sigma)$ and $\mathbf{t}:\{x\}\to T(\Sigma, X)$ with $\mathbf{t}(x)=r$, then, the condition (b) of **D2** is satisfied with respect to $(v;\ G)$, so we obtain

$$\mathscr{A}\vdash(v;\ G\supset \mathbf{t}^{ex}(x)\approx \mathbf{t}^{ex}(x)) \quad \text{or} \quad \mathscr{A}\vdash(v;\ G\supset r\approx r).$$

(2) $r=\Delta_s$

If we take $(v;\ \phi\supset\Delta_s\approx\Delta_s)\in TAUT(\Sigma)$, then for every $\mathbf{t}\in T(\Sigma, X)_s$, the condition (b) of **D2** is satisfied with respect to $(v;\ G)$, so we obtain

$$\mathscr{A}\vdash(v;\ G\supset \mathbf{t}^{ex}(\Delta_s)\approx \mathbf{t}^{ex}(\Delta_s))$$

or

$$\mathscr{A}\vdash(v;\ G\supset\Delta_s\approx\Delta_s). \qquad \square$$

Definition A.2 Let $\Sigma=(S,\ \alpha:\Omega\to S^*\times S)$ be any signature, and $\mathscr{A}$ any set of elementary implications of Σ, $(v;\ G)$ any set of equations of $L\ (\Sigma, X)$ on $v:X\to S$. Then, $F(\mathscr{A}, G, v)\in ALG'(\Sigma)$ is defined as follows.

1. For every $s\in S$, we set

$$F^*(\mathscr{A}, G, v)_s=\{(t\in G)\mid t\in T(\Sigma, X)_s\}.$$

The next step is the construction of an S-equivalence

$$\equiv=(\equiv_s\mid s\in S)$$

in the S-set $F^*(\mathscr{A}, G, v)$ by, for any (t, G), $(t', G)\in F^*(\mathscr{A}, G, v)_s$,

$$(t, G)\equiv_s(t', G) \quad \text{iff} \quad \mathscr{A}\vdash(v;G\supset t\approx t').$$

The above binary relation $\equiv_s$ in $F^*(\mathscr{A}, G, v)$ is a congruence relation for every $s\in S$. (Reflexivity is proved by Proposition A.1. Symmetry, transitivity and congruence are clear from Definition 8 and Definition 9.)

2. We define

$$F(\mathscr{A}, G, v)_s=F^*(\mathscr{A}, G, v)/\equiv_s \quad \text{for every} \quad s\in S$$

and the elements of $F(\mathscr{A}, G, v)$, i.e., the equivalence classes, is denoted by the following.

(1) For every $s\in S$ and any $(t, G)\in F^*(\mathscr{A}, G, v)_s$,

$$[t, G]=\{(r, G)\mid(t, G)\equiv_s(r, G)\}.$$

(2) For any $(t_w, G)\in F^*(\mathscr{A}, G, v)_w$ with $t_w=(t_1, \ldots, t_n)$,

$$[t_w, G]=([t_1, G], \ldots, [t_n, G]).$$

(3) For every $s\in S$,

$$\Delta_s^{F(\mathscr{A}, G, v)}=[\Delta_s, G].$$

3. For every $\sigma\in\Omega(\sigma:\lambda\to\sigma)$,

$$\sigma^{F(\mathscr{A}, G, v)}=[\sigma, G].$$

4. For every $\sigma\in\Omega(\sigma:w\to s)$,

$$\sigma^{F(\mathscr{A}, G, v)}([t_w, G])=[\sigma(t_w), G].$$

According to this,

$$F(\mathscr{A}, G, v)=(F(\mathscr{A}, G, v)_s\mid s\in S). \qquad \square$$

Proposition A.3 Let $(r;\ H)$ be any non-empty set of

equations of $L(\Sigma, Y)$ on $r: Y \to S$. An assignment $\mathbf{f} \in F(\mathscr{A}, G, v)_r$ with $v: X \to S$, given by

$$\mathbf{f}(y) = [\mathbf{t}(y), G], \quad \mathbf{t} \in T(\Sigma, X)_r$$

for every $y \in Y$, is a solution of $(r; H)$ in $F(\mathscr{A}, G, v)$ if and only if

$$\mathscr{A} \vdash (v; G \supset \mathbf{t}^{ex}(H)).$$

$(\mathbf{t}^{ex}(H) = \{(v; \mathbf{t}^{ex}(p')) \approx \mathbf{t}^{ex}(p')) \mid (r; p \approx p') \in (r; H)\})$

(Proof)
(sufficiency)
Let $\mathbf{f} \in F(\mathscr{A}, G, v)_{(r;H)}$. Then

$$\mathbf{f}^{ex}(p) = \mathbf{f}^{ex}(p')$$

holds for every $(r; p \approx p') \in (r; H)$.

Because $T(\Sigma, Y)$ is a free algebra,

$$[\mathbf{t}^{ex}(p), G] = [\mathbf{t}^{ex}(p'), G].$$

Therefore,

$$\mathscr{A} \vdash (v; G \supset \mathbf{t}^{ex}(p) \approx \mathbf{t}^{ex}(p')).$$

This is identical with

$$\mathscr{A} \vdash (v; G \supset \mathbf{t}^{ex}(H)).$$

(necessity)
We assume

$$\mathscr{A} \vdash (v; G \supset \mathbf{t}^{ex}(H))$$

or

$$\mathscr{A} \vdash (v; G \supset \mathbf{t}^{ex}(p) \approx \mathbf{t}^{ex}(p'))$$

for every $(r; p \approx p') \in (r; H)$. Then,

$$[\mathbf{t}^{ex}(p), G] = [\mathbf{t}^{ex}(p'), G].$$

Here, two cases will be considered.

(1) $p = \Delta_s$ or $p' = \Delta_s$

It is trivial that

$$[\mathbf{t}, G] : Y \to F(\mathscr{A}, G, v)$$

is a solution of $(r; p \approx p')$.

(2) $p \neq \Delta_s$ and $p' \neq \Delta_s$

Then

$$\mathbf{t}^{ex}(p) \neq \Delta_s \quad \text{and} \quad \mathbf{t}^{ex}(p') \neq \Delta_s.$$

So

$$[\mathbf{t}^{ex}(p), G] \neq \Delta_s \quad \text{and} \quad [\mathbf{t}^{ex}(p'), G] \neq \Delta_s.$$

Moreover, since $[\mathbf{t}^{ex}(p), G] = [\mathbf{t}^{ex}(p'), G]$,

$$[\mathbf{t}, G] : Y \to F(\mathscr{A}, G, v)$$

is a solution of $(r; p \approx p')$.

From (1) and (2) above,

$$[\mathbf{t}, G] \in F(\mathscr{A}, G, v)_{(r;p \approx p')}.$$

Since the above can hold for every $(r; p \approx p') \in (r; H)$,

$$\mathbf{f} = [\mathbf{t}, G] \in F(\mathscr{A}, G, v)_{(r;H)}. \qquad \square$$

Proposition A.4 $F(\mathscr{A}, G, v)$ with $v: X \to S$ is an $\mathscr{A}$-algebra.

(Proof) Two cases will be considered.

(1) Let $(r; \phi \supset p \approx p')$ with $r: Y \to S$ be any elementary implication of $\mathscr{A}$ which premise is an empty set. For any $\mathbf{t} \in T(\Sigma, X)_r$, we can obtain

$$\mathscr{A} \vdash (v; G \supset \mathbf{t}^{ex}(p) \approx \mathbf{t}^{ex}(p'))$$

by **D2**. Therefore,

$$[\mathbf{t}^{ex}(p), G] = [\mathbf{t}^{ex}(p'), G]$$

and

$$[\mathbf{t}, G] \in F(\mathscr{A}, G, v)_{(r;p \approx p')}.$$

From this, we can conclude, for any $\mathbf{f} \in F(\mathscr{A}, G, v)_r$, $\mathbf{f} \in F(\mathscr{A}, G, v)_{(r;p \approx p')}$

because $\mathbf{f}$ is represented using $\mathbf{t}$, i.e. $\mathbf{f} = [\mathbf{t}, G]$.

(2) Let $(r; H \supset p \approx p')$ with $r: Y \to S$ be any elementary implication of $\mathscr{A}$ whose premise is a non-empty set and $\mathbf{f} \in F(\mathscr{A}, G, v)_{(r;H)}$. According to Proposition A.3, this means

$$\mathscr{A} \vdash (v; G \supset \mathbf{t}^{ex}(H))$$

where $\mathbf{t} \in T(\Sigma, X)_r$ with $\mathbf{f} = [\mathbf{t}, G]$. By **D2**, we can obtain

$$\mathscr{A} \vdash (v; G \supset \mathbf{t}^{ex}(p) \approx \mathbf{t}^{ex}(p')).$$

A further application of Proposition A.3 leads to

$$\mathbf{f} = [\mathbf{t}, G] \in F(\mathscr{A}, G, v)_{(r;p \approx p')}.$$

From (1) and (2) above, $F(\mathscr{A}, G, v)$ has been shown to satisfy any elementary implication of $\mathscr{A}$. $\square$