Semantics of Function Tables on the Reals

By
Sameena Hossain, B.Sc.

A Thesis
Submitted to the School of Graduate Studies
in partial fulfilment of the requirements for the degree of

Master of Science
Department of Computing and Software
McMaster University

MASTER OF SCIENCE (2013) McMaster University
(Computing and Software) Hamilton, Ontario

TITLE: **Semantics of Function Tables on the Reals**

AUTHOR: Sameena Hossain, B.Sc. (Indiana University of Pennsylvania, USA)

SUPERVISOR: Dr. Jeffery I. Zucker

NUMBER OF PAGES: iv, 60

Abstract

This thesis develops a formalism of function tables, inspired by the work of D. Parnas. It adapts that formalism so as to apply to topological partial algebras, involving continuous partial functions on the reals. In particular, it studies semantics-preserving transformations between two classes of tables: normal and inverted. This leads to a 3-valued logic different from that used by Wei Lei (2007) who investigated the application of function tables to "error algebras".

Acknowledgements

I would like to express my heartfelt gratitude and appreciation to my supervisor Dr. J. I. Zucker, for his cnstant support, guidance and encouragement throughout my graduate work.

I would also like to thank Dr. Alan Wassyng and Dr. Sanzheng Qiao, the members of my committee, for their helpful comments.

Also I would like to thank my family for their support and inspiration, specially my son Arwaad for his unconditional and unwavering love.

Contents

Chapter 1

Introduction

1.1 Background and objectives

In this thesis we present a semantics for *function tables* on reals. We primarily adopt the method of tabular represenations developed by David Parnas and his collaborators and further extend and modify it following [TZ00] and [WL07].

This thesis introduces a systematic method for handling undefined values in computation over many-sorted algebras, by the use of *extended algebras*. In this thesis, we will extend the work of [TZ00] and [WL07], to deal with *partial functions*, particularly as it applies to reals. In developing this method, two desirable attributes of such algebras are of importance to us:

(1) *monotonicity*, which is a weaker condition than *strictness*, as will be discussed

in Chapter 3 (§3.2), and

(2) *continuity*, which ensures reliability of *outputs*, as will also be discussed in Chapter 3 (§3.3).

As a good framework for continuity, we will work mainly with tables based on *topological partial algebra* over the reals.

An important aspect of our work (as with [Zuc96, WL07]) is to consider *transformations* of function tables from one format to another (i.e. from normal to inverted, and vice versa), and to compare the situation here (with divergent values) to that in [TZ00] and [WL07] (with error values).

A significant issue here is the appropriate definition of *properness* of tables, which is found to differ in all these cases. Another issue here is the use of *3-valued logic*. It is found that whereas in dealing with error values [WL07] *strict* versions of the propositional operators ($\bigwedge$, $\bigvee$) are needed, here, *strong* versions of these operators are appropriate.

Our semantic theory will apply uniformly to *proper* and *improper* function tables.

1.2 Related work on partial functions

Partial functions and *undefinedness* have been areas of interest for a considerable amount of time. Some background information can be found in [Far90, Far95, TZ00].

Tabular representation play a very important role in the development of extended algebras. The method of tabular representations, developed by David Parnas and his collaborators, has been found to be very useful for the formal documentation and inspection of software systems. This thesis extends these methods used by Parnas.

This technique was first applied in the documentation for the revised flight software for the US Navy's A-7 aircraft in the late seventies [Hen80, HKP78]. Another large project which used tabular notation was the documentation of the shutdown systems of the Darlington Nuclear Power Generating Station in Ontario, Canada, required by the Atomic Energy Control Board of Canada for that station's licensing, in the late eighties [Par93, PAM91]. These two projects served both as testing grounds for the tabular method, and as incentives for its further development.

The tabular method is also useful in the documentation of simple programs, as demonstrated in [PMI92]. Some examples of its use in system documentation are given in [WT95]. A survey of the method is given in [JPZ96].

The method of tabular notations is, essentially, a useful and perspicuous method for defining functions on many-sorted algebras. In the course of the projects described above, many kinds of tables were developed, and were found to be useful. A systematic exposition of ten kinds of tabular expressions was given in [Par92].

In [WL07], this method was extended, following [TZ00], to deal with error values.

Here in this thesis we continue it further to deal with undefined values.

1.3 Overview

Chapter 2 discusses the basic concepts by giving the fundamental definitions of many-sorted signatures Σ, and Σ-algebras.

In Chapter 3, we introduce *extended algebras*. We discuss two desirable attributes of these algebras: *monotonicity* and *continuity*.

In Chapter 4, we develop a semantics of function tables using extended algebras which

(1) like [WL07], extends the semantic theory of [Zuc96] by defining a uniform semantics for proper and improper tables, and

(2) develops this further to deal with *partial functions* and *divergent values* unlike [WL07], which dealt with total functions and *error* values. We consider two kinds of tables: *normal* and *inverted*, and transformations between them, and point out the differences from the treatment in [WL07].

Chapter 5 summarizes the main results, and considers possible future work.

In conclusion, let us consider what has been accomplished here.

The main motivation for this thesis was to develop a theory of tabular notations applied to *partial topological algebras* (typically over the reals), with *undefined values*, semantics incorporating a suitable *three-valued logic*, and appropriate transformation rules between normal and inverted tables. These turn out to be quite different from the corresponding concepts and rules appropriate for (total) *error algebras* with *error values* [WL07]. It should be noted that the practical applicability of this investigation is not (as yet) clear. However it is, we feel, an interesting exercise in computation theory. (We thank Dr. Wassyng for this observation.)

Chapter 2

Basic Concepts

In this Chapter, we introduce the basic concepts used in this thesis. The presentation of the chapter makes use of [WL07] (Chapter 2) except that we work with partial algebras instead of total algebras. We introduce many-sorted signatures Σ and Σ-algebras. Most of the material can be found in [TZ99, TZ00, TZ04, WL07].

2.1 Basic algebraic concepts

Definition 2.1.1 (Many-sorted signature Σ). A many-sorted signature Σ is a pair $\langle \boldsymbol{Sort}(\Sigma), \boldsymbol{Func}(\Sigma) \rangle$ where

(1) $\boldsymbol{Sort}(\Sigma)$ is a finite set of sorts or basic types, written s, s', ...

(2) ***Func***(Σ) is a finite set of primitive (or basic) function symbols

$$F : s_1 \times \cdots \times s_m \to s \qquad (m \geq 0).$$

Each symbol F has a *type* $s_1 \times \cdots \times s_m \to s$, where $s_1, \ldots, s_m$ are the *domain sorts* and s is the *range sort* of F. The *arity* of F is $m \geq 0$. The case $m = 0$ corresponds to *constant symbols*; we write $c : \ \to s$ in this case.

Definition 2.1.2 (Product types over Σ). A Σ-*product type*, or a *product type* over Σ, has the form $u = s_1 \times \cdots \times s_m$ $(m \geq 0)$, for Σ-sorts $s_1, \ldots, s_m$. We write $u, v, w, \ldots$ for Σ-*product types.*

Definition 2.1.3 (Σ-algebras). A Σ-*algebra* A has:

(1) for each Σ-sort s, a non-empty set A_s, called *the carrier set* of sort s;

(2) for each Σ-*function* symbol $F : u \to s$, a (partial, possibly total) *function* $F^A : A^u \rightharpoonup A_s$ where u is the Σ-*product* type $s_1 \times \cdots \times s_m$, s is a Σ-*sort* and

$$A^u = A_{s_1} \times \cdots \times A_{s_m}.$$

Note. For $m = 0$, the meaning of the constant symbol $c : \ \to s$ is an element $c^A \in A_s$.

Example 2.1.4. We can present signatures Σ as:

signature	Σ
sorts	$s, \ldots$
functions	$F : s_1 \times \cdots \times s_n \to s$
	$\vdots$
end	

where $\textbf{\textit{Sort}}(\Sigma) = \{s, \ldots\}$

and $\textbf{\textit{Functions}}(\Sigma) = \{F : s_1 \times \cdots \times s_m \to s, \ldots\}$

We can then present a Σ-algebra A as:

algebra	A
carriers	$A_s \ (s \in sort(\Sigma))$
functions	$F : A_{s_1} \times \cdots \times A_{s_m} \to A_s$
	$\vdots$
end	

Remark 2.1.5. (Partial algebras). In general we assume that our algebras are partial, i.e. the basic functions are partial. It may happen that a particular algebra is total (i.e. all the basic functions are total).

Example 2.1.6. The algebra $\mathcal{B}$ of ***booleans*** has signature

signature	$\Sigma(\mathcal{B})$
sorts	bool
functions	$\mathsf{true},\ \mathsf{false} : \ \to \mathsf{bool}$,
	$\wedge, \vee : \mathsf{bool}^2 \to \mathsf{bool}$,
	$\neg : \mathsf{bool} \to \mathsf{bool}$
end	

Then the algebra $\mathcal{B}$ is:

algebra	$\mathcal{B}$
carriers	$\mathbb{B}$
functions	$\mathsf{tt},\ \mathsf{ff} : \ \to \mathbb{B}$,
	$\wedge_{\mathcal{B}}, \vee_{\mathcal{B}} : \mathbb{B}^2 \to \mathbb{B}$,
	$\neg_{\mathcal{B}} : \mathbb{B} \to \mathbb{B}$
end	

where $\mathsf{true}^{\mathsf{B}} = \mathsf{tt}$, $\mathsf{false}^{\mathsf{B}} = \mathsf{ff}$, and the standard boolean operations have their usual meaning.

In future, for a Σ-algebra A, we will display the algebra A itself from which its signature Σ can be inferred.

Example 2.1.7. The algebra $\mathcal{Z}_0$ of ***integers***:

algebra	$\mathcal{Z}_0$
carrier	$\mathbb{Z}$
functions	$0,\ 1\!: \ \to \mathbb{Z}$,
	$+, \times : \mathbb{Z}^2 \to \mathbb{Z}$,
	$- : \mathbb{Z}^2 \to \mathbb{Z}$
	...
end	

where the signature $\Sigma(\mathcal{Z})$ has sort int.

Example 2.1.8. The standard algebra $\mathcal{Z}$ of ***integers***:

algebra	$\mathcal{Z}$
import	$\mathcal{Z}_0, \mathcal{B}$
functions	$\mathsf{eq}^{\mathsf{Z}} : \mathbb{Z}^2 \to \mathbb{B}$,
	$\mathsf{less}^{\mathsf{Z}} : \mathbb{Z}^2 \to \mathbb{B}$
end	

(“Standardness” is defined in §2.3.)

We will use infix ‘$=$’ and ‘$<$’ for the eq^{Z} and $\mathsf{less}^{\mathsf{Z}}$ functions.

Example 2.1.9. The ring $\mathcal{R}_0$ of ***reals*** is

algebra	$\mathcal{R}_0$
carrier	$\mathbb{R}$
functions	$0, 1 : \to \mathbb{R}$,
	$+, \times : \mathbb{R}^2 \to \mathbb{R}$
end	

Example 2.1.10. The field $\mathcal{R}_1$ of ***reals*** is

algebra	$\mathcal{R}_1$
import	$\mathcal{R}_0$
functions	$\mathsf{inv} : \mathbb{R} \to \mathbb{R}$
end	

where for all $x \in \mathbb{R}$

$$\mathsf{inv}(x) \simeq \begin{cases} 1/x & \text{if } x \neq 0 \\ \uparrow & \text{otherwise} \end{cases}$$

where "$\uparrow$" denotes ***divergence***.

Notes.

(1) Here '$\simeq$' is "Kleene equality" which means that the two sides both converge to the same value , or both diverge.

(2) Since inv is a partial function, $\mathcal{R}_1$ is a *partial algebra*. The significance of this (related to continuity), will be discussed later (Remark 2.5.2).

2.2 Standard signatures and algebras

Definition 2.2.1 (Standard signatures). A signature Σ is *standard* if

(i) $\Sigma(\mathcal{B}) \subseteq \Sigma$, and

(ii) the function symbols of Σ include a *conditional*

$$\mathsf{if}_s : \mathsf{bool} \times s^2 \to s$$

for all sorts s of Σ other than bool, and an *equality operator*

$$\mathsf{eq}_s : s^2 \to \mathsf{bool}$$

for certain sorts s.

Definition 2.2.2 (Standard algebras). Given a standard signature Σ, a Σ-algebra A is a *standard algebra* if

(i) it is an expansion of $\mathcal{B}$, and

(ii) the conditionals and equality operators have their standard interpretation in A; i.e., for $b \in \mathbb{B}$ and $x, y \in A_s$,

$$\mathsf{if}_s(b, x, y) \simeq \begin{cases} x & \text{if } \mathrm{b} \downarrow \mathtt{tt} \\ y & \text{if } \mathrm{b} \downarrow \mathtt{ff} \\ \uparrow & \text{if } \mathrm{b} \uparrow \end{cases}$$

and eq_s also has its standard interpretation.

Remark 2.2.3. Any many-sorted signature Σ can be *standardized* to a standard signature Σ^B by adjoining the sort bool together with the standard boolean operations; and, correspondingly, any algebra A can be *standardized* to a standard algebra A^B by adjoining the algebra $\mathcal{B}$ and other boolean operators, e.g. the equality operator at certain sorts.

Assumption 2.2.4 (Standardness). We will assume our signatures and algebras are standard.

2.3 Continuous functions; Topological partial algebras

Definition 2.3.1 (Continuous function). Given two topological spaces X and Y, a *partial function* $f : X \rightharpoonup Y$ is continuous iff, for every open $V \subseteq Y$,

$$f^{-1}[V] =_{df} \{x \in X | x \in \boldsymbol{dom(f)} \text{ and } f(x) \in Y\}$$

is open in X.

Definition 2.3.2 (Topological partial Σ-algebra). A topological partial Σ-algebra is a partial Σ-algebra with topologies on the carriers such that each of the basic functions is continuous. Further, if the carriers $\mathbb{B}$ and/or $\mathbb{Z}$ are present, they have the discrete topology.

2.4 The algebra $\mathcal{R}$ of reals

Example 2.4.1 (Real algebra). An important many-sorted standard topological partial algebra for our purpose is the standard partial algebra of reals $\mathcal{R}$, with signature $\Sigma(\mathcal{R})$.

The algebra $\mathcal{R}$ is

algebra	$\mathcal{R}$
import	$\mathcal{R}_1, \mathcal{Z}, \mathcal{B}$
functions	eq^{R}, $\mathsf{less}^{\mathsf{R}}$: $\mathbb{R}^2 \to \mathbb{B}$,
end	

where eq^{R} and $\mathsf{less}^{\mathsf{R}}$ are *partial functions* defined by

$$\mathsf{eq}^{\mathsf{R}}(x,y) \simeq \begin{cases} \uparrow & \text{if } x = y \\ \mathsf{ff} & \text{if } x \neq y \end{cases}$$

$$\mathsf{less}^{\mathsf{R}}(x,y) \simeq \begin{cases} \mathsf{tt} & \text{if } x < y \\ \mathsf{ff} & \text{if } x > y \\ \uparrow & \text{if } x = y \end{cases}$$

Again, we will use infix '=' and '<' for these operators.

Remark 2.4.2. The reason for the partial defnitions of the functions $\mathsf{inv}^{\mathsf{R}}$, eq^{R} and $\mathsf{less}^{\mathsf{R}}$ is to ensure their continuity, since the total versions of these functions are not continuous [TZ99, TZ00]. This is discussed further in Chapter 3 (§3.3). This ensures that $\mathcal{R}$ is a *topological* partial algebra.

Note that $\mathcal{R}$ also contains eq^Z and less^Z which are total. These functions *are* continuous since $\mathcal{Z}$ has the discrete topology.

Remark 2.4.3. The standard algebra $\mathcal{R}$ (or some expansion of it) will be the main source of examples in this thesis.

2.5 Terms over Σ: syntax and semantics

Definition 2.5.1 (Variables).

(1) For each Σ-sort s, $\boldsymbol{Var}_s$ is a countable set of variables of sort $s : \mathtt{x}^s, \mathtt{y}^s, \ldots$

(2) $\boldsymbol{Var}(\Sigma) = \bigcup_{s \in \boldsymbol{Sort}(\Sigma)} \boldsymbol{Var}_s$

Definition 2.5.2 (Terms).

(1) The set $\boldsymbol{Tm}_s(\Sigma)$ of Σ-term of sort s is defined inductively by the clauses:

(a) $\boldsymbol{Var}_s(\Sigma) \subseteq \boldsymbol{Tm}_s(\Sigma)$.

(b) if $c : \to s$ is in $\boldsymbol{Func}(\Sigma)$ then $c \in \boldsymbol{Tm}_s(\Sigma)$.

(c) if $F : s_1 \times \cdots \times s_m \to s$ is in $\boldsymbol{Func}(\Sigma)$ and $t_i \in \boldsymbol{Tm}_{s_i}$ for $i = 1, \ldots, m$ then $F(t_1, \ldots, t_m) \in \boldsymbol{Tm}_s(\Sigma)$

(2) $\boldsymbol{Tm}(\Sigma) = \bigcup_{s \in \boldsymbol{Sort}(\Sigma)} \boldsymbol{Tm}_s(\Sigma)$

Note: In (1), clause (b) can be taken as a special case of clause (c), with $m = 0$.

Definition 2.5.3 (States over A). Let A be a Σ-algebra. A state over A is a family

$$\sigma = \langle \sigma_s \mid s \in \boldsymbol{Sort}(\Sigma) \rangle$$

of functions

$$\sigma_s : \boldsymbol{Var}_s \to A_s.$$

Definition 2.5.4 (Term evaluation). Each Σ-term t has a *value* $[\![t]\!]^A\sigma$ in A relative to state σ. The function

$$[\![t]\!]^A : \boldsymbol{State}(A) \to A_s$$

is defined by structural induction (or recursion) on t:

(a) $[\![\mathtt{x}_s]\!]^A\sigma = \sigma_s(\mathtt{x}^s)$.

(b) $[\![c]\!]^A\sigma = c^A$.

(c) $[\![F(t_1, \ldots, t_m)]\!]^A\sigma \simeq F^A([\![t_1]\!]^A\sigma, \ldots, [\![t_m]\!]^A\sigma)$

Note: if $t : s$ then $[\![t]\!]^A\sigma \in A_s$.

Notation 2.5.5. $\boldsymbol{Var}(t)$ is the set of variables occurring in t.

Notation 2.5.6. We write $\sigma(\mathtt{x}^s)$ for $\sigma_s(\mathtt{x}^s)$.

Definition 2.5.7. For any $M \subseteq \boldsymbol{Var}(\Sigma)$, and states σ and σ':

$$\sigma \approx \sigma' \text{ (rel } M) \quad \Longleftrightarrow \quad \sigma \restriction M = \sigma' \restriction M$$

i.e. σ and σ' *agree on* M.

Lemma 2.5.8 (Functionality Lemma for terms)**.** *For any* Σ*-term* t*:*

$$\sigma \approx \sigma' \ (rel\ \boldsymbol{Var}(t)) \quad \Longrightarrow \quad [\![t]\!]^A\sigma = [\![t]\!]^A\sigma'$$

Proof. By structural induction on t. □

Notation 2.5.9. $\boldsymbol{CT}(\Sigma)$ is the set of closed Σ-terms (where t is *closed* if $\boldsymbol{Var}(t) = \emptyset$).

Corollary 2.5.10. If t is closed then $[\![t]\!]\sigma$ is independent of σ.

Proof. By the Functionality Lemma for terms. □

So if t is closed we can write $[\![t]\!]^A =_{df} [\![t]\!]^A\sigma$ for all σ.

Remarks 2.5.11.

(a) We write $[\![t]\!]^A\sigma\downarrow$ to mean that evaluation of $[\![t]\!]^A\sigma$ *halts*, or *converges*, and $[\![t]\!]^A\sigma\downarrow a$ to mean that evaluation of $[\![t]\!]^A\sigma$ converges to a value a.

(b) We write $[\![t]\!]^A\sigma\uparrow$ to mean that evaluation of $[\![t]\!]^A\sigma$ *diverges*.

Chapter 3

Extended Algebras and their Semantics

In this chapter we will discuss extended algebras. We will also discuss two desirable attributes of such algebras that are useful for the purpose of this thesis, ***monotonicity*** and ***continuity***. The second of these, continuity, was introduced in Chapter 2 (Definition 2.3.1).

3.1 Extended algebras: the undefined value '$\uparrow$'

In working with a partial Σ-algebra A, we must also consider what we will call "extended semantics"- that is, how the basic Σ-functions and Σ-terms will behave with divergent inputs. This is relevant when we want to compose partial functions.

It is convenient to think of the partial basic Σ-functions F^A as being defined on an extended Σ-algebra $A^{\uparrow}$ with carrier sets

$$A_s^{\uparrow} =_{df} A_s \bigcup \{\uparrow\}$$

and with basic function semantics

$$F^{A^{\uparrow}} : A_{s_1}^{\uparrow} \times \cdots \times A_{s_m}^{\uparrow} \rightharpoonup A_s^{\uparrow}.$$

This is reminiscent of the construction of error algebras A^{ϵ} in [WL07], in which the carriers A_s are extended to $A_s \bigcup \{\epsilon\}$, where ϵ is an "error value" of sort s.

Definition 3.1.1 (Strict and consistent extensions). For each Σ-function symbol $F : u \rightharpoonup s$, we say:

(1) $F^{A^{\uparrow}}$ is ***strict*** over A if for all $a_i \in A_{s_i}$ $(i = 1, \ldots, m,\ i \neq k)$,

$$F^{A^{\uparrow}}(a_1, \ldots, \uparrow, \ldots, a_m) \uparrow$$

i.e. $F^{A^{\uparrow}}(a_1, \ldots, a_m)$ has divergent output if any argument is $\uparrow$; and

(2) $F^{A^{\uparrow}}$ is ***consistent*** over A if it extends F^A, *i.e.* $F^{A^{\uparrow}} \restriction A = F^A$.

Definition 3.1.2 (Basic extended signature and algebra). . Given a Σ-algebra

$$A = (A_{s_1}, \ldots, A_{s_{k-1}}, \mathbb{B}, F_1^A, \ldots, F_n^A),$$

let $A^{\uparrow}$ be the algebra

$$(A^{\uparrow}_{s_1}, \ldots, A^{\uparrow}_{s_{k-1}}, \mathbb{B}^{\uparrow}; F_1^{A^{\uparrow}}, \ldots, F_n^{A^{\uparrow}}, (\uparrow_s)_{s \in \boldsymbol{Sort}(\Sigma)})$$

of signature $\Sigma^{\uparrow}$ where

$$\begin{aligned} \boldsymbol{Sort}(\Sigma^{\uparrow}) &= \boldsymbol{Sort}(\Sigma) \\ \boldsymbol{Func}(\Sigma^{\uparrow}) &= \boldsymbol{Func}(\Sigma) \cup \{(\uparrow_s)_{s \in \boldsymbol{Sort}(\Sigma)}\}, \end{aligned}$$

and for all sorts $s : \uparrow_s^A = \uparrow$, and for $i = 1, \ldots, n$, $F_i^{A^{\uparrow}}$ is the *strict*, *consistent* extension of F_i^A.

We call

(1) $\Sigma^{\uparrow}$ the ***basic extended signature*** over Σ;

(2) $A^{\uparrow}$ the ***basic extended algebra*** over A.

Example 3.1.3 (Basic extended algebra based on $\mathcal{B}$). Consider the algebras:

$$\mathcal{B} = (\mathbb{B};\ \mathsf{tt}, \mathsf{ff}, \mathsf{and}, \mathsf{or}, \mathsf{not})$$

$$\mathcal{B}^{\uparrow} = (\mathbb{B}^{\uparrow};\ \mathsf{tt}, \mathsf{ff}, \uparrow, \mathsf{and}, \mathsf{or}, \mathsf{not})$$

where $\mathbb{B}^{\uparrow} = \{\mathsf{tt}, \mathsf{ff}, \uparrow\}$. The logical operators and, or *and* not which extend the regular boolean operators *strictly* and *consistently*, give rise to *a weak 3-valued logic*.

Example 3.1.4 (Other extended algebras on $\mathcal{B}$). The strict 3-valued boolean operators have the following truth tables:

$\wedge$	$\mathbb{tt}$	$\mathbb{ff}$	$\uparrow$
$\mathbb{tt}$	$\mathbb{tt}$	$\mathbb{ff}$	$\uparrow$
$\mathbb{ff}$	$\mathbb{ff}$	$\mathbb{ff}$	$\uparrow$
$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$

Table 1: Strict 'and' ($\wedge$)

$\vee$	$\mathbb{tt}$	$\mathbb{ff}$	$\uparrow$
$\mathbb{tt}$	$\mathbb{tt}$	$\mathbb{tt}$	$\uparrow$
$\mathbb{ff}$	$\mathbb{tt}$	$\mathbb{ff}$	$\uparrow$
$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$

Table 2: Strict 'or' ($\vee$)

We can also define *strong* version of these:

$\triangle$	$\mathbb{tt}$	$\mathbb{ff}$	$\uparrow$
$\mathbb{tt}$	$\mathbb{tt}$	$\mathbb{ff}$	$\uparrow$
$\mathbb{ff}$	$\mathbb{ff}$	$\mathbb{ff}$	$\boldsymbol{\mathbb{ff}}$
$\uparrow$	$\uparrow$	$\boldsymbol{\mathbb{ff}}$	$\uparrow$

Table 3: AND (**Strong** 'and', $\triangle$)

$\triangledown$	tt	ff	$\uparrow$
tt	tt	tt	$\boldsymbol{\mathsf{tt}}$
ff	tt	ff	$\uparrow$
$\uparrow$	$\boldsymbol{\mathsf{tt}}$	$\uparrow$	$\uparrow$

Table 4: OR **(Strong** 'or', $\triangledown$**)**

Note that all these operators are *commutative*.

Discussion 3.1.5 (Other non-strict boolean operators). Consider the statements:

(1) $\mathtt{x} \neq 0$ and $(1 \text{ div } \mathtt{x}) > 0$

(2) $\mathtt{x} = 0$ or $(1 \text{ div } \mathtt{x}) > 0$

Suppose $\mathtt{x} = 0$ (*i.e.* evaluated at σ with $\sigma(\mathtt{x}) = 0$). We may very well want:

- statement (1) to evaluate to ff; and
- statement (2) to evaluate to tt.

But strict operators would (in both cases) evaluate to $\uparrow$.

A good solution is to use cand ("conditional and") and cor ("conditional or"). These operators evaluate conjunctions and disjunctions *from the left*:

$\bigwedge^c$	tt	ff	$\uparrow$
tt	tt	ff	$\uparrow$
ff	ff	ff	$\boldsymbol{\mathsf{ff}}$
$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$

Table 5: cand **(conditional** 'and', $\bigwedge^c$**)**

$\bigvee^c$	tt	ff	$\uparrow$
tt	tt	tt	$\boldsymbol{\mathsf{tt}}$
ff	tt	ff	$\uparrow$
$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$

Table 6: cor **(conditional** 'or', $\bigwedge^c$**)**

Remarks 3.1.6.

(1) Unlike *strict* and *strong* 'and', and 'or', cand and cor are *not* commutative. Nevertheless these operators are computationally meaningful. In functional programming languages, such as SML, they are called 'andalso' and 'orelse' respectively.

(2) We can also *add* operators cand and cor to the algebra

$$\mathcal{B} = (\mathbb{B}; \mathsf{tt}, \mathsf{ff}, \mathsf{and}, \mathsf{or}, \mathsf{not})$$

which is then extended *consistently* but *not strictly* (cf. Definition 3.1.1) to the extended algebra:

$$\mathcal{B}^{\uparrow} = (\mathbb{B}^{\uparrow}; \mathsf{tt}, \mathsf{ff}, \uparrow, \mathsf{and}, \mathsf{or}, \mathsf{not}, \mathsf{cand}, \mathsf{cor})$$

Remarks 3.1.7. Consider the *standardized* data algebra

$$A = \mathcal{D}^{\mathcal{B}} = (\mathbb{D}, \mathbb{B}; \ldots, \mathsf{tt}, \mathsf{ff}, \wedge, \vee, \neg, \mathsf{eq}^{D}, \mathsf{if}^{D})$$

and the basic extended algebra over A

$$A^{\uparrow} = \mathcal{D}^{\mathcal{B}^{\uparrow}} = (\mathbb{D}^{\uparrow}, \mathbb{B}^{\uparrow}; \ldots, \mathsf{tt}, \mathsf{ff}, \wedge, \vee, \neg, \mathsf{eq}^{D,\uparrow}, \mathsf{if}^{D,\uparrow}, \uparrow^{D})$$

Now, consider the interpretation of equality 'eq' and the conditional 'if' in $A^{\uparrow}$:

(1) ***Equality vs identity***: The function $\mathsf{eq}^{A^{\uparrow}}$ *extends* eq^{A} by *strictness* ("*weak equality*" on $A^{\uparrow}$) so

$$\mathsf{eq}^{A^{\uparrow}}(x, \uparrow) \;\simeq\; \uparrow$$

for all "values" of x, including $\uparrow$.

On the other hand, the identity function ("strong equality") on $A^{\uparrow}$ has the form:

$$\mathsf{id}^{A^{\uparrow}} : (\mathbb{D}^{\uparrow})^2 \to \mathbb{B}$$

where

$$\mathsf{id}^{A^{\uparrow}}(x, \uparrow) = \begin{cases} \mathsf{tt} & \text{if } x \simeq \uparrow \\ \mathsf{ff} & \text{otherwise} \end{cases}$$

Note that $\mathsf{id}^{A^\uparrow}$ is a *non-strict* extension of eq^A. Also, eq^A is more meaningful computationally than $\mathsf{id}^{A^\uparrow}$.

(2) ***Conditional***: Note that $\mathsf{if}^{A^\uparrow}$ *extends* if^A by *strictness*. But it is *not* a "conditional operation" on $\mathcal{D}^\uparrow$ (as usually understood), since e.g.:

$$\mathsf{if}^{A^\uparrow}(\mathsf{tt}, d, \uparrow) \ \simeq \ \uparrow \quad (\text{not } d)$$

This can be called a "*weak conditional*" on $A^\uparrow$.

We could also adjoin a *non-strict* (or "strong") *conditional* to $A^\uparrow$:

$$\mathsf{if}_{\mathsf{ns}} : \mathbb{B}^\uparrow \times (\mathbb{D}^\uparrow)^2 \rightharpoonup \mathbb{D}^\uparrow$$

where

$$\mathsf{if}_{\mathsf{ns}}(b, x, y) \simeq \begin{cases} x & \text{if } b \simeq \mathsf{tt} \\ y & \text{if } b \simeq \mathsf{ff} \\ \uparrow & \text{if } b \simeq \uparrow \end{cases}$$

This is a *non-strict* extension of if with the standard meaning for the conditional, thus:

$$\mathsf{if}_{\mathsf{ns}}(\mathsf{tt}, d, \uparrow) \downarrow \ d \quad (\text{not } \uparrow).$$

Note that $A^\uparrow$ is not (strictly speaking) standard, according to our definition (2.2.2) since it contains $\mathcal{B}^\uparrow$ rather than $\mathcal{B}$. However, for practical purposes, we can treat it as a standard algebra.

Remark 3.1.8. We will assume from now on, that our standard extended algebras $A^{\uparrow}$ contain the ***nonstrict conditional*** $\mathsf{if}_{\mathsf{ns}}$, rather than its strict counterpart.

Example 3.1.9. Consider

$$\mathcal{R} = (\mathbb{R}, \mathbb{Q}, \mathbb{Z}, \mathbb{B}; \ldots, \mathsf{eq}_Q, \mathsf{eq}_Z)$$

Equality would (or should) be available on $\mathbb{Q}$, $\mathbb{Z}$ (and $\mathbb{B}$) but not $\mathbb{R}$, since equality on the reals is not computable. (But see Remark 2.4.2 and §3.5.).

3.2 Non-strict semantics; Monotonicity

In general, the extended semantics for a Σ-function F^A is determined simply by ***strictness*** (see §3.1).

However, as we have seen above, there are some important exceptions in our standard algebras: for example, the boolean operations cand, cor, AND, OR and the conditional operator $\mathsf{if}_{\mathsf{ns}}$.

The semantics for the above operations, although not strict, are all *monotonic* (in the extended semantics), as we now explain.

Definition 3.2.1 (Monotonicity). A partial function

$$f : A^{\uparrow} \times B \rightharpoonup C$$

is *monotonic* if, for any $b \in B$, if

$$f(\uparrow, b) \downarrow c \text{ (say)},$$

then also, for any $a \in A$,

$$f(a, b) \downarrow c.$$

In other words, if for some divergent input '$\uparrow$', the output *converges* to c, then replacing '$\uparrow$' by any element of A, will give the *same convergent* output.

Note that for functions, strictness trivially implies monotonicity. However as the following lemma shows, monotonicity is a more useful property than strictness for partial algebras.

Definition 3.2.2. An extended algebra is *monotonic* if all its basic functions are *monotonic*.

Lemma 3.2.3. *The boolean operations* cand $(\bigwedge^c)$, cor $(\bigvee^c)$, AND $(\triangle)$, OR $(\bigtriangledown)$, *and the conditional operator* $\mathsf{if}_{\mathsf{ns}}$, *are all monotonic.*

Proof. This is clear by checking their semantic definitions. $\square$

Remark 3.2.4. It follows from the above lemma that the standard algebras $\mathcal{B}^{\uparrow}$, $\mathcal{Z}^{\uparrow}$ and $\mathcal{R}^{\uparrow}$ are also *monotonic.*

We introduce partial equality. Let $A^{\uparrow} = (A_s^{\uparrow}, \dots)$ be a $\Sigma^{\uparrow}$ algebra.

Definition 3.2.5. (a) For $a, b \in A_s^{\uparrow}$, $a \underset{\sim}{\sqsubset} b$ iff $a = b$ *or* $a = \uparrow$.

(b) For $M \subseteq \boldsymbol{Var}(\Sigma^{\uparrow})$,

$$\sigma \underset{\sim}{\sqsubset} \sigma' \text{ (rel } M) \quad (\text{“}\sigma \text{ is extended by } \sigma' \text{ relative to } M\text{”})$$

iff for all $\mathtt{x} \in M$, $\sigma(\mathtt{x}) \underset{\sim}{\sqsubset} \sigma'(\mathtt{x})$

Thus $\uparrow_s$ is the *minimal element* of $A_s^{\uparrow}$, and the state $\sigma_{\uparrow}$, where $\sigma_{\uparrow}(\mathtt{x}) = \uparrow$ for all $\mathtt{x}$, is the minimal element of $State(A^{\uparrow})$.

Proposition 3.2.6. $\sigma \approx \sigma'$ (rel M) $\iff$ $\sigma \underset{\sim}{\sqsubset} \sigma'$ (rel M) *and* $\sigma' \underset{\sim}{\sqsubset} \sigma$ (rel M)

Proof. Clear from Definitions 2.5.7 and 3.2.5. □

Remarks 3.2.7.

(1) The relation “$\underset{\sim}{\sqsubset}$ (rel M)” (for fixed M) is a *pre-partial order* on $\boldsymbol{State}(\underline{A}^{\uparrow})$, *i.e.* it is *transitive* and *reflexive* (but not *anti-symmetric*).

(2) The relation “$\approx$ (rel M)” is the corresponding *equivalence relation* on $\boldsymbol{State}(A^{\uparrow})$.

(3) The $\underset{\sim}{\sqsubset}$ $-$*minimal* states (rel M) are those which are *totally unspecified* on M.

Theorem 1 (Monotonicity for $\boldsymbol{Tm}(\Sigma)$). *Suppose $A^{\uparrow}$ is monotonic. Then for all $t \in \boldsymbol{Tm}(\Sigma)$, and $\sigma, \sigma' \in \boldsymbol{State}(A^{\uparrow})$:*

$$\sigma \underset{\sim}{\sqsubset} \sigma' \ (rel\ \boldsymbol{Var}(t)) \implies [\![t]\!]\sigma \underset{\sim}{\sqsubset} [\![t]\!]\sigma'$$

Proof. By structural induction on t. □

Remark 3.2.8. As a corollary we get: if $A^{\uparrow}$ is monotonic, then

$$\sigma \approx \sigma'(rel\ \boldsymbol{Var}) \implies [\![t]\!]\sigma \simeq [\![t]\!]\sigma'$$

but we know this already from the Functionality Lemma for terms (Lemma 2.5.8) without the assumption of monotonicity.

3.3 Continuity; Its significance for computing functions on $\mathcal{R}$

In this section we will discuss the continuity of the operational semantics of partial algebras. We will see the advantage of our "algebraic" approach, since these functions are built up from simpler functions using composition, thus preserving continuity. We begin with a standard result.

Lemma 3.3.1 (Basic lemma on continuity). *The composition of continuous functions is continuous.*

Recalling Theorem 1 (in §3.2), we have:

Theorem 2 (Continuity of term functions). *For $t \in Tm(\Sigma)$, the function*

$$[\![t]\!]^A : \ State(A) \ \rightharpoonup \ A^{\uparrow}$$

is continuous.

Proof. Using Definition 2.3.1, Lemma 3.3.1 and structural induction on t.

$\square$

Discussion 3.3.2. The importance of continuity in relation to computation is that it provides *stability* and *reliability* in connection with readings of input and output values. This can be best seen by considering a "rudimentary" (1-dimensional) normal table, which defines a (total) step function $f : \mathbb{R} \to \mathbb{R}$:

$\mathtt{x} \leq 0$	0
$\mathtt{x} > 0$	1
H^1	G

Table 7

(This is an example of a normal table. Normal tables will be defined in Chapter 4.)

Here the output of the table function f is 0 if the input $\mathtt{x}$ is ≤ 1, and 1 if $\mathtt{x} > 1$. Thus f is *discontinuous* at 0.

Further, the evaluation of f near 0 is *unstable*, for suppose the input $\mathtt{x}$ is read as being (approximately) 0. Then the output is taken to be 0. However, if at a later time, the input is read as being (even slightly) > 0, then the output could be re-evaluated as 1. (Clearly this problem is not solved by redefining the table to have output 1 at 0.)

Later still the output could be read again as 0, and so on. This problem is bound up with the imperfections and variability of input measuring instruments.

The solution is to define f as a *partial* step function, thus:

`x` $<$ 0	0
`x` $>$ 0	1
H^1	G

Table 8

Now when the input value of `x` is read as being very close to 0 (and not clearly either < 0 or > 0), the output will be given as '$\uparrow$' (undefined).

If and when further readings clearly confirm the input to be either < 0 or > 0, then the output can be given accordingly (as 0 or 1 respectively).

Note that this "reassessment" of the input value will not *contradict* any previous output values, by the ***monotonicity*** property of our extended algebra (see §3.2).

To summarize the above discussion:

Reliability of output readings (as a function of input readings) depends on the ***continuity*** of the table functions: *small changes* in the input readings produce only *small changes* in the output readings. Continuity of these functions depends, in turn, on the ***partiality*** of (at least some of) the basic functions (such as equality and division on the reals): diverging when necessary, rather than having to converge to a (possibly unreliable) value. The "good behaviour" of divergence, and its replacement by convergence as a result of further measurements, depends in turn on the

monotonicity of the basic functions.

3.4 Basic algebraic results

From now on we assume that we are dealing with the topological partial algebra $\mathcal{R}$ (Remark 2.4.3).

The following results can be found, with proofs, in standard texts on algebra [Lan90, Wae64], real analysis [Roy66, Rud76], and constructive analysis [PER89, Wei00].

Let $\boldsymbol{E}$ be the equational calculus in the language $(0, 1, +, -, *)$, with the axioms for rings (i.e. that $(\mathbb{R}, 0, +, -)$ is a commutative group, $(\mathbb{R}, 1, *)$ is a commutative monoid, and $*$ is distributive over $+$). By "real term" we mean term of type real.

Definition 3.4.1 (Computational equivalence). Two real terms t_1, t_2 are *computationally equivalent* (written $t_1 \cong t_2$) iff $\boldsymbol{E} \vdash t_1 = t_2$.

Remarks 3.4.2. Any $\Sigma(\mathcal{R})$ *term* t of type real can be rewritten as a polynomial, which can be specified uniquely according to a prescribed ordering of variables, etc. [XFZ13].

Note that here polynomial expressions in "standard form" have integer coefficients, although the signature Σ does not have a data type int. The point is that our "polynomial notation" does not involve integers essentially. For example, the polynomial expression '$2x^2 \ -3x \ + \ 4$' stands for the Σ-term $x*x+x*x+(-x)+1+1+1$ (suitably parenthesized) of type real.

Remark 3.4.3. Computational equivalence of real terms is decidable, by transforming each term to its "canonical polynomial" and comparing them.

3.5 Modified semantics for equality of terms on $\mathcal{R}$

In order to motivate our semantics, consider the pair of real terms

$$t_1 \ \equiv \ \mathtt{x}, \quad t_2 \ \equiv \ 0 \tag{$*$}$$

Suppose $\sigma(\mathtt{x}) \ = \ 0$, i.e. $[\![t_1]\!]\sigma \ = \ 0$. Then we still define

$$[\![t_1 = t_2]\!]\sigma \ \uparrow \tag{$**$}$$

This follows from the definition of eq^{R} in §2.5, which is motivated by reasons of continuity (recall Remark 2.4.2, also see \$ 3.3), since for $\sigma(\mathtt{x})$ *approximately* but not *exactly* equal to 0,

$$[\![t_1 = t_2]\!]\sigma \ \downarrow \ \mathsf{ff}.$$

However for the pair of terms (*say*)

$$t_1 \ \equiv \ \mathtt{x}+\mathsf{2}, \quad t_2 \ \equiv \ \mathsf{1}+\mathtt{x}+\mathsf{1} \tag{$***$}$$

we have $[\![t_1]\!]\sigma = [\![t_2]\!]\sigma$, for all states σ, and to stipulate (**) here would be *counterintuitive*! Moreover, unlike the case (*), discontinuity is not an issue here, since for *all* σ, $[\![t_1]\!]\sigma$ and $[\![t_2]\!]\sigma$ are *exactly equal.* Note also that, here, unlike case (*), $t_1 \;\cong\; t_2$.

Similar considerations apply to boolean expression of inequality of real term

$t_1 < t_2$.

In view of the above considerations, we revise the definitions (in Example 2.4.1) of the partial functions eq^{R} and $\mathsf{less}^{\mathsf{R}}$ in $\mathcal{R}$, as in [XFZ13], to obtain:

$$[\![t_1 = t_2]\!]\sigma \;\simeq\; \begin{cases} \mathtt{tt} & \text{if} \;\; \mathrm{t}_1 \cong \mathrm{t}_2 \\ \uparrow & \text{if} \;\; [\![\mathrm{t}_1]\!]\sigma \;=\; [\![\mathrm{t}_2]\!]\sigma \;\; \text{but} \;\; \mathrm{t}_1 \;\ncong\; \mathrm{t}_2 \\ \mathtt{ff} & \text{if} \;\; [\![\mathrm{t}_1]\!]\sigma \;\neq\; [\![\mathrm{t}_2]\!]\sigma. \end{cases}$$

$$[\![t_1 < t_2]\!]\sigma \;\simeq\; \begin{cases} \mathtt{tt} & \text{if} \;\; \mathrm{t}_1 < \; \mathrm{t}_2 \\ \mathtt{ff} & \text{if} \;\; [\![\mathrm{t}_1]\!]\sigma \;>\; [\![\mathrm{t}_2]\!]\sigma \;\; \text{or} \;\; \mathrm{t}_1 \;\cong\; \mathrm{t}_2 \\ \uparrow & \text{if} \;\; [\![\mathrm{t}_1]\!]\sigma \;=\; [\![\mathrm{t}_2]\!]\sigma \;\; \text{but} \;\; \mathrm{t}_1 \;\ncong\; \mathrm{t}_2. \end{cases}$$

Note that, with the above modified semantic definitions, term functions are still continuous. The proof depends on the fact that the condition for the atomic formula '$t_1 = t_2$' to have an output of $\mathtt{tt}$ instead of $\uparrow$ (i.e. that $t_1 \cong t_2$) is *independent* of the state (similarly for term '$t_1 < t_2$'). Hence the continuity proof still holds.

Chapter 4

Semantics of Function Tables

In this Chapter we will present semantics for different types of ***function tables***. We will discuss ***proper*** and ***improper tables***; ***normal*** and ***inverted tables***; and finally ***transformations*** between normal and inverted tables.

4.1 Tables: Previous work and motivation

(a) **Proper and improper tables**

In [Zuc96] Zucker considered two kinds of tabular expressions: normal and inverted. He provided a semantics for both kinds of tables, and defined transformation between them which preserve the semantics. However, the semantics apply only to the unproblematic case of "proper" tables. The extension of the semantics to "improper" tables was left as an open problem.

(b) **Tables based on error algebras**

The theory of tables based on "error algebras" [WL07] deals systematically with an error value ϵ at all sorts and extends the semantic theory of [Zuc96] by defining a *uniform semantics* for proper and improper tables in the context of error algebras. (It will be seen that ***divergent values*** require a treatment different from ***error values***.)

The approach taken in [WL07] was not to divide tables into "proper" and "improper" subclasses (as in [Zuc96]) but to consider, for any table T at any particular state σ, whether T is proper or improper at σ. (The answer will vary, in general, with σ). It was also found necessary to broaden the concept of "properness" used in [Zuc96], to allow overlapping conditions where the output value agrees on the overlap.

In this thesis, we will be constructing tables with "divergent values" (denoted '$\uparrow$'), mainly on the algebra $\mathcal{R}$ of reals. This leads to different table function semantics, as will be discussed below. We will develop the work of [Zuc96] and [WL07] by considering *partial algebras*, with *divergent values*, and how to deal with them.

4.2 Normal Tables

We will define the class $\boldsymbol{Tab}_N(\Sigma)$ of *normal (function) tables over* Σ. Consider (for convenience) a 2-dimensional normal table [Par92, Zuc96].

Example 4.2.1 (A two dimensional normal table).

C_1^2	C_2^2	$\ldots$	C_l^2

H^2

C_1^1
C_2^1
$\vdots$
C_k^1

H^1

t_{11}	t_{12}	$\ldots$	t_{1l}
t_{21}	t_{22}	$\ldots$	t_{2l}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_{k1}	t_{k2}	$\ldots$	t_{kl}

G

Table 9

In Table 9, the headers H^1 and H^2 of table T contain *conditions* C_i^1 $(1 \leq i \leq k)$ and C_j^2 $(1 \leq j \leq l)$ respectively. These are boolean-valued expressions over Σ, extended e.g. by bounded quantifiers. The cells (i, j) of the grid G of T contain terms $t_{i,j}$, all of the same Σ-sort: the ***output sort*** of T.

The value of T (at a given state) is the value of the cell determined by the conditions in the headers H^1 and H^2 which are evaluated to $\mathtt{tt}$ (at that state), assuming T is "proper"; i.e. assuming (for now) there is a *unique* i such that $[\![C_i^1]\!]\sigma \downarrow \mathtt{tt}$, and for all $i' \neq i$, $[\![C^1_{i'}]\!]\sigma \downarrow \mathsf{ff}$; and there is a *unique* j such that $[\![C^2_j]\!]\sigma \downarrow \mathsf{tt}$, and for all $j' \neq j$, $[\![C^2_{j'}]\!]\sigma \downarrow \mathsf{ff}$.

Later (Definition 4.3.2) we will give a different (more general, and more appropriate) definition of "properness", and we can call the above "strict properness".

Remarks 4.2.2. We cannot exclude improper tables at the syntactic level since

(1) properness of T depends on the state;

(2) properness (at all states) is not decidable in general.

Remarks 4.2.3. Here is a possible strategy for evaluating improper tables at a given state σ, assuming all headers have at least one condition which evaluates to tt: Take the leftmost (or topmost) condition which evaluates to tt (like the "case" statement in C). But this is unsatisfactory since

(1) the semantics is then dependent on the order of rows and columns, and hence would not be preserved by table transformations (from normal to inverted, and conversely; see below).

(2) The "leftmost" (or "topmost") cell in the header may give a divergent output and not allow evaluations of the other cells.

4.3 Properness of normal tables

We are looking for a condition on tables which will make their semantics unproblematic. Differing from the definition of properness in [Zuc96], we define "properness" as in [WL07] by allowing overlapping conditions, where the values agree on the overlap. There are also differences from the semantics in [WL07], as we will see below.

Definition 4.3.1 (Universality for headers over extended algebras). A tuple of conditions $(C_1, \ldots, C_n)$ is said to be *universal* at $\sigma \in \boldsymbol{State}(A^{\uparrow})$ if, for some i,

$$[\![C_i]\!]^{A^{\uparrow}}\sigma = \mathtt{tt}$$

Note that this allows the case that $[\![C_j]\!]^{A^{\uparrow}}\sigma \uparrow, \quad$ for some $j \neq i$.

Definition 4.3.2 (Proper normal table). T is *proper* at σ if

(i) all its headers are universal at σ, and

(ii) the value of a term t_{ij} at σ is the *same* for all (i, j) for which conditions C_i^1 in header H^1 and C_j^2 in header H^2 are ***true*** or ***divergent*** at σ.

Remarks 4.3.3.

(1) Condition (ii) says that the values agree on overlapping conditions that either evaluate to $\mathtt{tt}$ *or diverge*. This is different from the semantics in [WL07], where the presence of the error value '$\mathtt{ee}$' in any header renders the table improper (at that state).

(2) If the output sort of T is real, we must remember that equality between real terms is partial (cf. Remark 2.4.2 and §3.5), and so comparisons between such terms are not always possible. Condition (ii) must be interpreted as: the terms in cells associated with ***true or divergent*** conditions must be ***strongly equivalent***.

4.4 Semantics of normal tables

Definition 4.4.1. Let T be a normal table over Σ, and σ a state over T in A. Suppose T is proper at σ. Choose indices i, j for which the entries C_i^1 and C_j^2 evaluate to $\mathtt{tt}$ at σ. There is at least one such pair (i, j), since both headers are universal. Then the *meaning* of T at σ is

$$[\![T]\!]^A\sigma = [\![t_{ij}]\!]^A\sigma.$$

Note that by the *properness condition* (Definition 4.3.2), the value of $[\![t_{ij}]\!]\sigma$ does not depend on the choice of indices i, j for which $[\![C_i]\!]\sigma = [\![C_j]\!]\sigma = \mathtt{tt}$ or $\uparrow$.

Notation 4.4.2. For a $\mathsf{boolean}$ valued term C, at state σ, we write

$$\sigma \models C \text{ (“}\sigma\text{ satisfies C”) to mean : } [\![\mathrm{C}]\!]\sigma \downarrow \mathtt{tt}$$

.

Next we will define table functions relative to a list of variables.

Definition 4.4.3. A list $\bar{\mathtt{x}}$ of variables is said to *cover* T if it includes all of $\boldsymbol{Var}(T)$, i.e., if $\boldsymbol{Var}(T) \subseteq \bar{\mathtt{x}}$.

Definition 4.4.4 (Table function). Let $\bar{\mathtt{x}} \equiv (\mathtt{x}_1, \ldots, \mathtt{x}_m)$ be any list of variables which covers T, with $\mathtt{x}_i : s_i$ for $i = 1, \ldots, m$. Then relative to $\bar{\mathtt{x}}$, T *names* or *defines* a *table function*

$$f_{T,\bar{\mathtt{x}}} : \; s_1 \times \cdots \times s_m \to s$$

with interpretation on A

$$f^A_{T,\bar{\mathtt{x}}} : \; A_{s_1} \times \cdots \times A_{s_m} \rightharpoonup A_s$$

as follows. For all $a_1 \in A_{s_1}, \ldots, a_m \in A_{s_m}$, let σ be the state over A defined by $\sigma(\mathtt{x}_i) = a_i$ for $i = 1, \ldots, m$. Then

$$f^A_{T,\mathtt{x}}(a_1, \ldots, a_m) \;\simeq\; [\![T]\!]^A \sigma.$$

Note that this definition is independent of the choice of the state σ, by the Functionality Lemma for terms (Lemma 2.5.8).

Remarks 4.4.5.

(1) A term t_{ij} in the grid of T may very well diverge ($\uparrow$) at σ without causing T to be improper. This is analogous to the (non-strict) semantics for the conditional

$$\mathsf{if}_{\mathsf{ns}}(t^{\mathsf{bool}}, t^s_1, t^s_2)$$

(2) More interestingly, a condition in the header may evaluate to $\uparrow$ without causing T to be improper! This is justified by the monotonicity property of tables. This is unlike the definition of properness in [WL07], and points to a conceptual difference between ***error outputs*** ϵ and ***undefined outputs*** $\uparrow$.

We now extend the definition (4.4.1) of $[\![T]\!]^{A}\sigma$ to the case that T is improper at σ.

Definition 4.4.6 (**Semantics of normal tables over** Σ**).** Let T be a normal table over Σ, and σ a state over T in $A^{\uparrow}$. We define $[\![T]\!]^{A^{\uparrow}}\sigma$ as follows:

Case 1: T is proper at σ. Then $[\![T]\!]^{A^{\uparrow}}\sigma$ is as in Definition 4.4.1.

Case 2: T is improper at σ. Then $[\![T]\!]^{A^{\uparrow}}\sigma\ \uparrow$.

Theorem 3. *Let $A^{\uparrow}$ be an extended topological algebra which is monotonic. Let T be a normal table, with* $\boldsymbol{Var}(T) \subseteq \mathtt{x}$. *Then $f^{A^{\uparrow}}_{T,\mathtt{x}}$ is*

(1) monotonic, and

(2) continuous.

Proof.

(1) This follows easily from the monotonicity of term functions (Theorem 1), and the definition of properness of tables (Definition 4.3.2).

(2) To prove continuity of $f^{A^{\uparrow}}_{T,\mathtt{x}}$: Suppose $\overline{a} \in \boldsymbol{dom}(f^{A}_{T,\overline{\mathtt{x}}})$. Take σ *s.t.* $\sigma[\overline{\mathtt{x}}] = \overline{a}$.

Then T is proper at σ, and

$$f^{A^{\uparrow}}_{T,\overline{\mathtt{x}}}(\overline{a}) = [\![T]\!]^{A^{\uparrow}}\sigma.$$

Let $\overline{a}\,'$ be a tuple of inputs "near" $\overline{a}$, and take σ' *s.t.* $\sigma'[\overline{\mathtt{x}}] = \overline{a}\,'$. We consider two cases:

(i) The output sort of T is nat or bool.

Then (since these are discrete spaces) for $\overline{a}\,'$ sufficiently near $\overline{a}$, T' is also proper at σ', and the value of $[\![T]\!]^{A^\dagger}\sigma'$ is actually *the same as* $[\![T]\!]^{A^\dagger}\sigma$.

(ii) The output sort of T is real.

Then by the condition for properness of T in Remark 4.3.3(2), for $\overline{a}\,'$ sufficiently near $\overline{a}$, T' is also proper at σ', by strong equivalence of real valued terms in cells with overlapping true conditions, and the value of $[\![T]\!]^{A^\dagger}\sigma'$ is *close* to the value of $[\![T]\!]^{A^\dagger}\sigma$, by the *continuity* of *term functions* (Theorem 2).

$\square$

4.5 Inverted tables

In this section we consider the class $\boldsymbol{Tab}_I(\Sigma)$ of *inverted (function) tables over* Σ. Such a table T differs from a normal table in the following way (see Table 12).

(1) One of its headers H^1, is the *value header*. Instead of *conditions*, it contains *terms*, all of the same Σ-sort, the *output sort* of T. The other header H^2, the

condition header, contains conditions as before.

(2) The cells of T contain *conditions* instead of terms.

The idea (or operational semantics) for T is as follows. For a given state σ over T, search the condition header H^2 until you find a condition C_j which holds at σ. The index j determines a column. Search along this column for a cell (i, j) whose entry C_{ij} has the value $\mathtt{tt}$. The corresponding entry t_i in H^1 then gives the value of the function.

The desirability of this search always producing a unique value, leads to the following definition of properness for inverted tables. Let T be an inverted table as follows:

Example 4.5.1 (An inverted table)**.**

	C_1	$\dots$	C_j	$\dots$	C_l
t_1	C_{11}	$\dots$	C_{1j}	$\dots$	C_{1l}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_i	C_{i1}	$\dots$	C_{ij}	$\dots$	C_{il}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_k	C_{k1}	$\dots$	C_{kj}	$\dots$	C_{kl}

H^2

H^1 **Table 10** G

Definition 4.5.2 (Proper inverted tables). T is *proper* at σ iff

(1) For some $j, \sigma \models C_j$

(2) For all j s.t. $\sigma \models C_j$, there exits i s.t. $\sigma \models C_{ij}$.

(3) For all j s.t. $\sigma \models C_j$ or $[\![C_j]\!]\sigma \uparrow$, and all i s.t. $\sigma \models C_{ij}$ or $[\![C_{ij}]\!]\sigma \uparrow$, $[\![t_i]\!]\sigma$ has the same value $a \in A_s$ (where s is the output sort).

Remarks 4.5.3. (Cf. Remark 4.3.3(2)) For the case that the terms in H^1 are of sort real, we need the stronger condition that the term $[\![t_i]\!]\sigma$ in condition (3) above are *strongly equivalent*.

Example 4.5.4 (A proper inverted table). According to the definition found in [WL07], the table below is an example of a proper inverted table:

$y \geq 10$	$y < 10$

H^2

$x + y$
$x - y$
$y - x$

H^1

$x < 0$	$x < y$
$0 \leq x < y$	$y \leq x < 0$
$x \geq y$	$x \geq 0$

G

Table 11

Definition 4.5.5 (Semantics of inverted tables). Let T be an inverted table over Σ, and σ a state over T in A.

(1) Suppose T is *proper* at σ. Choose indices $j,\ i$ for which conditions (1), (2) and (3) of Definition 4.5.2 hold. Then the *meaning* of T at σ is

$$[\![T]\!]^A\sigma \;=\; [\![t_i]\!]^A\sigma$$

Note again that by the *properness condition* (Definition 4.5.2), the value of $[\![t_i]\!]^A\sigma$ does not depend on the actual choice of indices j and i for which conditions (1), (2) and (3) of Definition 4.5.2 hold.

(2) The extension of this defintion to improper tables is just as in Definition 4.4.6 for normal tables.

Definition 4.5.6 (Inverted table functions). This is defined exactly as for normal table functions (Defintion 4.4.4).

4.6 Transformations of tables

We are interested in transforming tables from normal to inverted and from inverted to normal, semantically equivalent tables which may be easier to work with. First we define the notion of *semantic equivalence of tables.*

Definition 4.6.1 (Semantic equivalence of tables over $A^{\uparrow}$). Two tables, T_1 and T_2 over $A^{\uparrow}$ are *semantically equivalent on* $A^{\uparrow}$ (written $T_1 \approx_{A^{\uparrow}} T_2$) iff for all states σ over $\boldsymbol{Var}(T_1, T_2)$ in $A^{\uparrow}$, $[\![T_1]\!]^{A^{\uparrow}}\sigma \simeq [\![T_2]\!]^{A^{\uparrow}}\sigma$.

Remark 4.6.2. Semantic equivalence is defined here not only as a relation between *proper tables* (as in [Zuc96]) but also for *improper tables*.

We will define *transformations*

$$\varphi: \ \tau \ \rightarrow \ \tau'$$

of tables from one class τ to another class τ'. These transformations must be effective (in the syntax) and also satisfy the following properties:

(1) For all σ, T is proper at σ iff $\varphi(T)$ is proper at σ,

(2) φ is *semantics preserving*, i.e. $\varphi(T) \approx T$.

If $\varphi(T) = T'$, then T' is called the *transform* of T under φ.

4.7 Inverting a normal table

Following [Zuc96], we consider two methods for transforming a normal table to a semantically equivalent inverted one.

In [WL07], the first inversion method is illustrated with a simple example as follows: Consider the case of a 2-dimensional 3×3 normal table T, given in Table 14.

Example 4.7.1 (A normal table)**.**

C^2_1	C^2_2	C^2_3

H^2

C^1_1
C^1_2
C^1_3

H^1

t_{11}	t_{12}	t_{13}
t_{21}	t_{22}	t_{23}
t_{31}	t_{32}	t_{33}

G

Table 12

T is "inverted along dimension 1" to produce an inverted table (Table 13) with condition header H^2 unchanged, and value header H^1, much bigger than the original, since the length of the value header in the new table has increased to the size of the original table, i.e. the number of cells in its grid.

The second method for inversion is appropriate for a normal table T in which the number of *distinct* terms (up to strong equivalence) in its grid is *small*. Suppose, e.g., the grid in Table 14 contains only 2 terms (up to strong equivalence), say t_1 and t_2, as shown in Table 16. According to Method 2, we invert T, also along dimension 1, to produce Table 17.

Example 4.7.2 (Inversion of Table 12: Method 1).

C_1^2	C_2^2	C_3^2

H^2

H^1	G		
t_{11}	C_1^1	false	false
t_{21}	C_2^1	false	false
t_{31}	C_3^1	false	false
t_{11}	false	C_1^1	false
t_{21}	false	C_2^1	false
t_{31}	false	C_3^1	false
t_{11}	false	false	C_1^1
t_{21}	false	false	C_2^1
t_{31}	false	false	C_3^1

Table 13

Example 4.7.3 (A special case of Table 12).

C_1^2	C_2^2	C_3^2

H^2

H^1	G		
C_1^1	t_1	t_1	t_2
C_2^1	t_2	t_1	t_2
C_3^1	t_1	t_2	t_2

Table 14

Example 4.7.4 (Inversion of Table 14: Method 2)**.**

C_1^2	C_2^2	C_3^2

H^2

t_1
t_2

$\tilde{\mathsf{H}}^1$

$C_1^1 \bigtriangledown C_3^1$	$C_1^1 \bigtriangledown C_2^1$	false
C_2^1	C_3^1	$C_1^1 \bigtriangledown C_2^1 \bigtriangledown C_3^1$

$\tilde{\mathsf{G}}$

Table 15

Note that ***strong disjunction*** ($\bigtriangledown$) is used here, in contrast to error algebras [WL07] where ***strict disjunction*** is used. This points to the conceptual distinction between error values and divergent values. Strict disjunction is used in [WL07], so as to not to hide error values, but *strong* disjunction is used here so as to incorporate divergent values in true conditions. (This difference is illustrated by Example 4.7.7 below.)

The following theorems holds for both inversion transformations considered in this Section.

Lemma 4.7.5. *Let T be a normal table, and $\widetilde{T}$ the inverted table obtained from T by Method 1 or 2. Then*

$$\widetilde{T} \textit{ is proper at } \sigma \iff T \textit{ is proper at } \sigma.$$

Proof. We show

(1) T is proper at $\sigma \implies \widetilde{T}$ is proper at σ;

(2) T is improper at $\sigma \implies \widetilde{T}$ is improper at σ.

(1) T is proper at σ.

The proof is similar as for Theorem 3 (1) in [Zuc96]. Note that we need the "strong" definition of disjunction to make this work in the case that some of the conditions diverge, unlike the error case in the error algebras [WL07], as noted above.

(2) T is improper at σ.

If H^2 is not universal in T (at some state σ), then the same header H^2 is not universal in $\widetilde{T}$. If H^1 is not universal in T, then all the columns in the grid of $\widetilde{T}$ will also not be universal.

If H^1 and H^2 in T are both universal (at σ) but lead to different values on the overlap, then these different values will also manifest themselves in the value header of $\widetilde{T}$.

$\square$

Remarks 4.7.6. Suppose the normal table T (Table 13) is proper but *not strictly proper*, e.g. if $\sigma \models C_1^2$ and $\sigma \models C_1^1$ and also $\sigma \models C_3^1$. Then the inverted table by Method 2 (Table 15) is still *strictly proper*. Hence Lemma **??** does not hold with "properness" replaced by "strict properness". This explains our more liberal definition of properness, following [WL07].

Theorem 4. *Suppose T is a normal table, and $\widetilde{T}$ is the inverted table obtained from T by Method 1 or Method 2. Then*

$$\widetilde{T} \approx_{A^{\uparrow}} T.$$

Proof. There are two cases.

(1) T is a proper at σ. Similar to Theorem 3 in [Zuc96, §8].

(2) T is improper at σ. Then by Lemma 4.7.5 the inverted table $\widetilde{T}$ is also improper. Thus we have,

$$[\![T]\!]^{A^{\uparrow}}\sigma \text{ and } [\![\widetilde{T}]\!]^{A^{\uparrow}}\sigma \text{ both diverge.}$$

□

Example 4.7.7 (Different forms of disjunction)**.** (See Example 4.7.4)

Let T be the following table:

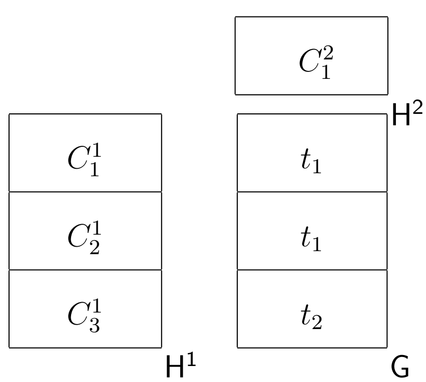

Table 16: T

Now by inverting T, we get $\widetilde{T}$ as follows:

		C_1^2
		H^2
t_1		$C_1^1 \vee C_2^1$
t_2		C_3^1
H^1		G

Table 17: $\widetilde{T}$

In order to show the contrast, suppose, at a particular state,

C_1^2 becomes $\mathbb{tt}$, C_1^1 becomes $\mathbb{ee}$ or $\uparrow$, C_2^1 becomes $\mathbb{tt}$ and C_3^1 becomes $\mathbb{ff}$

First, working with error values [WL07], T becomes

		$\mathbb{tt}$
		H^2
$\mathbb{ee}$		t_1
$\mathbb{tt}$		t_1
$\mathbb{ff}$		t_2
H^1		G

Table 18

And by inverting with *strict disjunction*, we get $\widetilde{T}$ which becomes

tt

H^2

t_1
t_2

H^1

$\mathsf{tt} \vee \epsilon\epsilon \rightarrow \epsilon\epsilon$
ff

G

Table 19

Both tables are seen to be improper, as expected.

Alternatively, working (as in this thesis) with ***divergent*** values, we have T as

tt

H^2

$\uparrow$
tt
ff

H^1

t_1
t_1
t_2

G

Table 20

And by inverting T now with ***strong disjunction***, we get, $\widetilde{T}$ which evaluates to

	$\mathtt{tt}$
t_1	$\mathtt{tt} \bigtriangledown \uparrow \rightarrow \mathtt{tt}$
t_2	$\mathtt{ff}$

H^2 H^1 G

Table 21

Both tables (20 and 21) are now proper, as we would expect from Lemma 4.7.5.

Hence, in both cases properness (or improperness) is preserved, as are the semantics.

4.8 Normalizing an inverted Table

The transformation of an inverted table to a normal one produces a one-dimensional table. The table presents a simpler view, with complex conditions inside the cells.

Adopting the example in [WL07], we consider the 2-dimensional 3×2 inverted table shown as Table 22, with value header H^1.

Example 4.8.1 (Two-dimensional table).

C_1^2	C_2^2

H^2

t_1
t_2
t_3

H^1

C_{11}	C_{12}
C_{21}	C_{22}
C_{31}	C_{32}

G

Table 22

This can be normalized to a 1-dimensional table, shown as Table 23.

Example 4.8.2 (Normalization of Table 22).

$(C_1^2 \bigtriangleup C_{11}) \bigtriangledown (C_2^2 \bigtriangleup C_{12})$
$(C_1^2 \bigtriangleup C_{21}) \bigtriangledown (C_2^2 \bigtriangleup C_{22})$
$(C_1^2 \bigtriangleup C_{31}) \bigtriangledown (C_2^2 \bigtriangleup C_{32})$

$\hat{\mathsf{H}}^1$

t_1
t_2
t_3

$\hat{\mathsf{G}}$

Table 23

Note again that ***strong conjunction*** ($\bigtriangleup$) and ***strong disjunction*** ($\bigtriangledown$) are used here, *not strict conjunction* and *strict disjunction*, as in [WL07], for error algebras. While combining H^2 conditions with the grid values, strong conjunction is used to eliminate all ff conditions and to obtain only tt or $\uparrow$ conditions. Strong disjunction is used so that none of the convergent values are hidden, and so that $\uparrow$ values can be incorporated in $\mathtt{tt}$ conditions, as discussed in Example 4.5.4.

Table 22 can also be normalized to Table 24, by "splitting strong disjunctions" in the conditions.

Example 4.8.3. Another normalization of Table 22 is as follows (following [Zuc96] and [WL07], but using strong disjunction):

$C_1^2 \vartriangle C_{11}$	t_1
$C_2^2 \vartriangle C_{12}$	t_1
$C_1^2 \vartriangle C_{21}$	t_2
$C_2^2 \vartriangle C_{22}$	t_2
$C_1^2 \vartriangle C_{31}$	t_3
$C_2^2 \vartriangle C_{32}$	t_3
H^1	G

Table 24

Lemma 4.8.4. *Let $\widehat{T}$ be the normal table obtained from T by the method of either Table 19 or Table 20. Then*

$$\widehat{T} \text{ is proper at } \sigma \iff T \text{ is proper at } \sigma.$$

Proof. By extending the method of Theorem 3(1) in [Zuc96] for proper tables, as in Lemma **??**(2).

□

Theorem 5. *Suppose T is an inverted table, and $\widehat{T}$ is the normal table obtained from T as above. Then:*

$$\widehat{T} \approx_{A\uparrow} T.$$

Proof. Similar to Theorem 4. □

Remark 4.8.5. Here also, we see that properness and improperness are both preserved, with our definition of properness (cf. Remark 4.7.6).

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis we have developed a systematic method for incorporating undefined values in computation over many-sorted algebras. As we have shown, by the use of *extended algebras*, undefined values can be handled effectively rather than being ignored/omitted as was done in the case of *error algebras*.

The main difference between our partial algebras with undefined values, and total algebras with error values [WL07] is the change of 3-valued logic, requiring strong, instead of strict, disjunction and conjunction (see Examples 4.8.2 and 4.8.3).

In computing with undefined values, the most desirable attributes for the partial functions are:

(1) *monotonicity*, which is a weaker condition than strictness (see §3.2), and

(2) *continuity*, which ensures reliability of outputs (see the discussion in §3.3).

We have applied this theory to the semantics of (proper and improper) function tables.

5.2 Future Work

There are many possible extensions of the work in this thesis; one of which is:

To develop a *single*, logically coherent, and not too complicated system, to incorporate both *error values* and *undefined values*.

Bibliography

[Bee85] M. Beeson. Foundations of Constructive Mathematics. Springer-Verlag. 1985.

[Far90] W. M. Farmer. A Partial Functions Version of Church's Simple Theory of Types. *Journal of Symbolic Logic*, 55:1269–91, 1990. Also MITRE Corporation technical report M88-52, 1988; revised 1990.

[Far95] W. M. Farmer. Reasoning About Partial Functions with the Aid of a Computer. *ERKENNTNIS: An International Journal of Analytic Philosophy*, 43:279–294, 1995.

[Fef95] S. Feferman. Definedness. *ERKENNTNIS: An International Journal of Analytic Philosophy*, 43:295–320, 1995.

[Hen80] K. L. Heninger. Specifying Software Requirements for Complex Systems: New Techniques and Their Apllication, *IEEE Transactions on Software*

Engineering, **SE-6**, 2–13, 1980.

[HKP78] K. L. Heninger, J. Kallander, D. L. Parnas, and J. E. Shore. Software Requirements for the A-7E Aircaraft. United States Naval Research Laboratory, Washington DC, NRL Memorandun Report 3876, 1978.

[Jon06] Cliff B. Jones. Reasoning about partial functions in the formal development of programs. *Electr. Notes Theor. Comput. Sci.*, 145:3–25, 2006.

[JPZ97] R. Janicki, D. L. Parnas, and J. I. Zucker. Tabular representations in relational documents. *Relational methods in computer science*, pages 184–196. Springer-Verlag New York, Inc. 1997.

[KK94] M. Kerber and M. Kohlhase. A mechanization of strong Kleene logic for partial functions. In A. Bundy, editor, *Automated Deduction—CADE-12*, volume 814 of *Lecture Notes in Computer Science*, pages 371–385. Springer-Verlag, 1994.

[Kle52] S. C. Kleene. *Introduction to metamathematics*. North-Holland, 1952.

[Luo03] L. Luo. Specifiability and Computability of Functions by Equations on Partial Algebras. Master's thesis, Dept. of Computing and Software, McMaster University, 2003. Technical Report CAS 03-07-JZ, Dept. of Computing and Software, McMaster University, April 2003.

[Par92] D. L. Parnas. Tabular representation of relations. Communications Research Laboratory, McMaster University, CRL Report 260, 1992.

[Par93] D. L. Parnas. Predicate logic for software engineering. *IEEE Transactions on Software Engineering*, 19:856–862, Springer-Verlag New York, Inc., 1993.

[Par95] D. L. Parnas. A Logic for Describing, Not Verifying, Software. *ERKENNTNIS: An International Journal of Analytic Philosophy*, 43:321–338, 1995.

[PAM91] D. L. Parnas, G. J. K. Asmis and J. Madey. Assessment of Safety-Critical Software in Nuclear Power Plants. *Nuclear Safety*, **32**, pages 189–198, 1991.

[PMI94] D. L. Parnas, J. Madey, and M. Iglewski. Formal documentation of well-structured program. *IEEE Transactions on Software Engineering*, 20:948-976, 1994.

[TZ88] J. V. Tucker and J. I. Zucker. Program Correctness over Abstract Data Types with Error-State Smantics. *IEEE Transactions on Software Engineering.* North-Holland, 1988.

[TZ00] J. V. Tucker and J. I. Zucker. Computable functions and semicomputable sets on many-sorted algebras. *Handbook of Logic in Computer Science* volume 5 section 1.2, pages 317–523, Oxford University Press, 2000.

[TZ04] J. V. Tucker and J. I. Zucker. Abstract versus concrete computation on metric partial algebras. *ACM Transactions on Computational Logic*, 2004.

[WL07] W. Lei. Error Algebras. *M.Sc. Thesis*, Department of Computing and Software, McMaster University, 2007.

[WT95] A. J. Wilder and J. V. Tucker. System Documentation Using Tables - a short course. Communications Research Laboratory, McMaster University, CRL Report 306, 1995.

[XFZ13] B. Xie, M. Q. Fu and J. I. Zucker. Characterizations of Semicomputable Sets of Real Numbers. To appear in *Journal of Logic and Algebraic Programming*, 2013.

[Zhu03] L. Zhu. Hoare logics for programming languages with partial functions and non-deterministic choice. Master's thesis, Dept. of Computing and Software, McMaster University, 2003. Technical Report CAS 06-04-JZ, Dept. of Computing and Software, McMaster University, April 2006.

[Zuc96] J. I. Zucker. Transformations of Normal and Inverted Function Tables. *Formal Aspects of Computing*, 8:679-705, 1996.