ERROR ALGEBRAS

Error Algebras

By

WEI LEI, B.ENG.

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

© Copyright by Wei Lei, November 2007

MASTER OF SCIENCE (2007)	McMaster University
(Computing and Software)	Hamilton, Ontario

TITLE: **Error Algebras**

AUTHOR: Wei Lei, B.Eng. (Wuhan University of Technology, China)

SUPERVISOR: Dr. Jeffery I. Zucker

NUMBER OF PAGES: iv, 70

Abstract

In computations over many-sorted algebras, one typically encounters error cases, caused by attempting to evaluate an operation outside its domain (e.g. division by the integer 0; taking the square root of a negative integer; popping an empty stack). We present a method for systematically dealing with such error cases, namely the construction of an "error algebra" based on the original algebra. As an application of this method, we show that it provides a good semantics for (possibly improper) function tables.

Acknowledgements

First, I would like to express my sincere thanks and appreciation to my supervisor Dr. J. I. Zucker, for his thoughtful guidance, stimulating suggestions and constant encouragement throughout my study.

Also, I am grateful to the members of my committee, especially Dr. W. M. Farmer, for their careful review and valuable comments. Thanks to all my other professors, notably Dr. S. Qiao and Dr. P. Speissegger, for their help during my studies.

My thanks also go to Jian Xu, and all my other friends for their friendship, support and help.

Last but not least, I would like to express my gratitude to my parents. They are always ready to help in any way I need. I cannot thank them enough for their love and support.

Contents

Abstracts iii

Acknowledgements iv

1 Introduction 1

1.1 Background and objectives 1

1.2 Related work on error analysis 2

1.3 Overview 2

2 Basic Concepts 4

2.1 Basic algebraic concepts 4

2.2 Reducts and expansions 10

2.3 Standard signatures and algebras 10

2.4 Stacks over algebra of data 13

2.5 Terms over Σ: syntax and semantics 16

3 Error Algebras **19**

3.1 The error value ϵ: algebras A^ϵ of signature Σ^ϵ 19

3.2 Semantics of term over Σ^ϵ 28

3.3 Monotonicity 34

3.4 Error-consistency 37

4 Semantic of Improper Tables using Error Algebras **41**

4.1 Normal tables 43

4.1.1 Properness 46

4.1.2 Semantics of normal tables 49

4.2 Inverted tables 55

4.3 Transformations of tables 58

4.4 Inverting a normal table 59

4.5 Normalising an inverted Table 64

4.6 Comparison with the logic of Parnas 66

5 Conclusion and Future Work **69**

5.1 Conclusion 69

5.2 Future work 70

Chapter 1

Introduction

1.1 Background and objectives

In this thesis, we will develop a systematic method for handling error cases in computation over many-sorted algebras using *error algebras.* Desirable properties in computing with error cases are:

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

(2) *error-consistency*, which is a weaker condition than consistency.

We will apply this theory to the semantics of proper and improper function tables.

In particular, the type of booleans, has an error value $\mathbb{ee}$ as well as $\mathbb{tt}$ and $\mathbb{ff}$, leading to a 3-valued logic.

1.2 Related work on error analysis

The treatment of error values, and the related areas of definedness of terms and partial function, have received a great deal of attention, with various approaches. Good exposition of some of these approaches can be found in [Far90, Far95, Fef95, Jon06, KK94, Luo03, Par95, Par03, TZ88, Zhu03]

A strong motivation for at least some of these approaches is the investigation of new error and exception handling in software analysis.

1.3 Overview

Chapter 2 gives the fundamental definitions of many-sorted signatures Σ and Σ-algebras.

In Chapter 3, we introduce error algebras. Also, we discuss two important properties of such algebras: monotonicity and error-consistency.

In Chapter 4 we present a semantics for function tables using error algebras which extends the semantic theory of [Zuc96] by defining a uniform semantics for both proper and improper tables.

We consider both normal and inverted function tables, and show that the semantics, as well as the properties of properness, and improperness are preserved under the transformation between these two classes of tables. Finally, a comparison with the logic used by Parnas in [Par93] is given in this chapter.

Chapter 5 summaries the main results of the thesis and considers possible future work.

Chapter 2

Basic Concepts

We briefly introduce the basic concepts used in this thesis in this chapter, including many-sorted signatures Σ and Σ-*algebras*. Some examples are provided.

Most of the material and more details can be found in [TZ99, TZ00, TZ03].

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;

(2) $\boldsymbol{Func}(\Sigma)$ 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 $s_1, \ldots, s_m \in \textbf{\textit{Sort}}(\Sigma)$ are the *domain sorts* and $s \in \textbf{\textit{Sort}}(\Sigma)$ is the *range sort* of F. The *arity* of F is $m \geq 0$. The case $m = 0$ corresponds to *constant symbols*; we write $F : \to s$ in this case.

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

Definition 2.1.3 (Σ-algebras). A *Σ-algebra* $\underline{A}$ has:

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

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

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

The algebra $\underline{A}$ is *total* if F^A is total for each *Σ-function* symbol F. We write $\Sigma(\underline{A})$ for the signature of an algebra $\underline{A}$ [see Chapter 3].

In this thesis we assume:

Assumption 2.1.4 (Totality Assumption). All algebras are total.

Remark 2.1.5. Note that the existence of error output for certain input values of a function F^A does not imply partiality of F^A, or of $\underline{A}$.

Example 2.1.6. The algebra of ***booleans*** has signature

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

Then the algebra $\mathcal{B}$ has the carrier $\mathbb{B} = \{\mathsf{tt}, \mathsf{ff}\}$ of sort bool, and so

$$\mathcal{B} = (\mathbb{B};\ \mathsf{tt},\ \mathsf{ff},\ \wedge,\ \vee,\ \neg)$$

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

Example 2.1.7. The algebra $\mathcal{W}(A)$ over a set $\underline{A}$ (an "alphabet") has signature

signature	$\Sigma(\mathcal{W})$
sorts	letter, word
functions	sing : letter $\to$ word,
	concat : word$^2 \to$ word,
	$\langle\ \rangle$: $\to$ word
end	

and so

$$\mathcal{W}(A) = (A, A^*;\ \mathsf{sing}^W,\ \mathsf{concat}^W,\ \langle\,\rangle^W).$$

Example 2.1.8. Algebras of ***naturals***:

(1) The algebra $\mathcal{N}_0$ of ***naturals*** has signature

signature	$\Sigma(\mathcal{N}_0)$
sorts	nat
functions	$0 : \to \mathsf{nat}$,
	$\mathsf{suc} : \mathsf{nat} \to \mathsf{nat}$
end	

The algebra $\mathcal{N}_0$ consists of the carrier $\mathbb{N}=\{0, 1, 2, \ldots\}$ of sort of nat, the zero constant $0^{\mathsf{N}} : \to \mathbb{N}$, and the successor function $\mathsf{suc}^{\mathsf{N}} : \mathbb{N} \to \mathbb{N}$, and so

$$\mathcal{N}_0 = (\mathbb{N};\ 0^N,\ \mathsf{suc}^N)$$

(2) The expanded algebra $\mathcal{N}$ of ***naturals*** has signature

signature	$\Sigma(\mathcal{N})$
import	$\mathcal{N}_0$
functions	$+, \times : \mathsf{nat}^2 \to \mathsf{nat}$
end	

The algebra $\mathcal{N}$ is expanded from the algebra $\mathcal{N}_0$ by adding functions $+ : \mathbb{N}^2 \to \mathbb{N}$ and $\times : \mathbb{N}^2 \to \mathbb{N}$:

$$\mathcal{N} = (\mathcal{N}_0;\ +^N,\ \times^N)$$

Example 2.1.9. We can also form algebras expanding $\mathcal{N}$ such as

$$\mathcal{N}' = (\mathcal{N},\ +^N, \times^N, \mathsf{pred}^N,\ \mathsf{div}^N, \mathsf{minus}^N,\ \mathsf{sqrt}^N)$$

Since the algebras are total (by the Totality Assumption), in order to define the terms such as $\mathsf{pred}^N(0)$ or $\mathsf{div}^N(2,0)$ we have to use *default values*:

$$\begin{aligned} \mathsf{pred}^N(0) &= 0 \\ \mathsf{div}^N(m,0) &= 0 \quad \text{for all } m \in \mathbb{N} \\ \mathsf{minus}^N(m,n) &= 0 \quad \text{for } m < n \end{aligned}$$

Example 2.1.10. The algebra $\mathcal{Z}$ of ***integers***:

signature	$\Sigma(\mathcal{Z})$
sorts	int
functions	$0,\ 1 : \to \mathsf{int}$,
	$+,\ \times : \mathsf{int}^2 \to \mathsf{int}$,
	$\mathsf{minus} : \mathsf{int}^2 \to \mathsf{int}$
	$\dots$
end	

The algebra $\mathcal{Z}$ consists of the carrier $\mathbb{Z}=\{\ldots, -2, -1, 0, 1, 2, \ldots\}$ of sort of int:

$$\mathcal{Z} = (\mathbb{Z};\ 0,\ 1,\ +^{Z},\ \times^{Z},\ \mathsf{minus}^{Z},\ \ldots).$$

Now pred and minus have *natural* (non-default) total definitions. But now we have new problems with functions such as:

$$\mathsf{sqrt}^{Z} : \mathsf{int} \to \mathsf{int}$$

$$\text{or } \mathsf{div}^{Z} : \mathsf{int}^2 \to \mathsf{int} \qquad \text{(integer division)}$$

For now, we again use *default values*:

$$\mathsf{sqrt}^{Z}(m) = \begin{cases} 0 & \text{for } m < 0 \\ n & \text{where } n^2 \le m < (n+1)^2 \text{ for } m \ge 0 \end{cases}$$

$$\mathsf{div}^{Z}(m, 0) = 0 \ \text{ for all } m \in \mathbb{Z}.$$

Example 2.1.11. The ring $\mathcal{R}_0$ of ***reals*** has signature

```
signature  Σ(R₀)

sorts      real

functions  0, 1 : → real,

           +, × : real² → real,

           − : real → real

end
```

and so $\mathcal{R}_0 = (\mathbb{R};\ 0,\ 1,\ +,\ -,\ \times)$.

2.2 Reducts and expansions

Definition 2.2.1. Let Σ and Σ' be signatures.

(1) $\Sigma \subseteq \Sigma'$ if and only if $\boldsymbol{Sort}(\Sigma) \subseteq \boldsymbol{Sort}(\Sigma')$ and $\boldsymbol{Func}(\Sigma) \subseteq \boldsymbol{Func}(\Sigma')$.

(2) Suppose $\underline{A}$ is a Σ-algebra, $\underline{A}'$ is a Σ'-algebra and $\Sigma \subseteq \Sigma'$.

(a) The *Σ-reduct* $\underline{A}'|_\Sigma$ of $\underline{A}'$ is the algebra of signature Σ, consisting of the carriers of $\underline{A}'$ named by the sorts of Σ, and equipped with the functions of $\underline{A}'$ named by the function symbols of Σ.

(b) The *Σ'-algebra* $\underline{A}'$ is a *Σ'-expansion* of $\underline{A}$ if and only if $\underline{A}$ is the *Σ-reduct* of $\underline{A}'$.

Example 2.2.2.

$$\mathcal{N}_0 = \mathcal{N}|_{\Sigma(\mathcal{N}_0)}$$

$$\mathcal{R}_0 = \mathcal{R}|_{\Sigma(\mathcal{R}_0)}$$

2.3 Standard signatures and algebras

Definition 2.3.1 (Standard signatures). A signature Σ is *standard* if $\Sigma(\mathcal{B}) \subseteq \Sigma$, and 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.

For a standard signature Σ, a sort of Σ is called an *equality sort* if Σ includes an *equality operator*

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

Definition 2.3.2 (Standard algebras). Given a standard signature Σ, a Σ-algebra A is a *standard algebra* if (i) it is an expansion of $\mathcal{B}$, (ii) the conditional operator on each sort s has its standard interpretation in A; *i.e.*, for $b \in \mathbb{B}$ and $x, y \in A_s$,

$$\mathsf{if}_s^A(b, x, y) = \begin{cases} x & \text{if } b = \mathsf{tt} \\ y & \text{if } b = \mathsf{ff}; \end{cases}$$

and (iii) the operator eq_s is interpreted as a *identity* on each equality sort s.

Example 2.3.3. The algebra $\mathcal{Z}^B$ has signature $\Sigma(\mathcal{Z}^B)$.

```
signature   Σ(Z^B)
import      Z, B,
functions   eq_int, less_int: int^2 → bool,
            if_int: bool × int^2 → int
end
```

Then

$$\mathcal{Z}^B = (\mathcal{Z},\ \mathcal{B};\ \mathsf{eq}^Z,\ \mathsf{less}^Z,\ \mathsf{if}^Z)$$

where the standard operations (listed above) have their *standard interpretations* on $\mathbb{Z}$.

More generally: Given a signature Σ and a Σ-*algebra* $\underline{A}$, a *boolean expansion* of Σ is a signature $\Sigma^{\mathcal{B}}$ where

$$\begin{aligned}
\boldsymbol{Sort}(\Sigma^{\mathcal{B}}) &= \boldsymbol{Sort}(\Sigma) \cup \{\mathsf{bool}\} \\
\boldsymbol{Func}(\Sigma^{\mathcal{B}}) &= \boldsymbol{Func}(\Sigma) \cup \\
&\quad\ \boldsymbol{Func}(\Sigma(\mathcal{B})) \cup \\
&\quad\ \{(\mathsf{eq}_s : s^2 \to \mathsf{bool})_{s \in S}, \\
&\quad\quad (\mathsf{if}_s : \mathsf{bool} \times s^2 \to s)_{s \in \boldsymbol{Sort}(\Sigma)} \\
&\quad\ \}
\end{aligned}$$

where $S \subseteq \boldsymbol{Sort}(\Sigma)$ is the set of equality sorts of Σ.

The boolean expansion of $\underline{A}$ is the $\Sigma^{\mathcal{B}}$-algebra

$$\underline{A}^B = (\underline{A}, \mathcal{B}, (\mathsf{if}_s^A)_{s \in \boldsymbol{Sort}(\Sigma)}, (\mathsf{eq}_s^A)_{s \in S})$$

where

$$\mathsf{if}_s^A : \mathbb{B} \times A_s^2 \to A_s \quad (s \in \boldsymbol{Sort}(\Sigma))$$

$$\text{and}\ \ \mathsf{eq}_s^A : A_s^2 \to \mathbb{B} \qquad\qquad (s \in S)$$

Example 2.3.4. The *standard algebra of reals* $\mathcal{R}_0^{\mathcal{B}}$ is formed by standardizing the ring $\mathcal{R}_0$.

Note that real is not generally chosen to be an equality sort, since equality between two reals is not decidable.

Remark 2.3.5. Any many-sorted signature Σ can be *standardized* to a standard signature $\Sigma^{\mathcal{B}}$ by adjoining the sort bool together with the standard boolean operations; and, correspondingly, any algebra A can be *standardized* to a standard algebra $A^{\mathcal{B}}$ by adjoining the algebra $\mathcal{B}$ and other boolean operators, e.g. the equality operation at the equality sorts of $\Sigma^{\mathcal{B}}$.

Assumption 2.3.6 (Standardness). We will assume our signatures and algebras are standard.

Remark 2.3.7. The standard algebra $\mathcal{Z}^{\mathcal{B}}$ (or some expansion of it) will be the main source of examples later in this thesis, especially in Chapter 4.

2.4 Stacks over algebra of data

Consider a standard algebra of data $\mathcal{D}$:

$$\mathcal{D}^{\mathcal{B}} = (\mathbb{D};\ \mathcal{B},\ F_1^D, \ldots,\ F_k^D,\ \mathsf{if}_{\mathsf{data}}^D,\ \mathsf{eq}_{\mathsf{data}}^D)$$

of signature Σ where

$$\begin{aligned}
\textbf{\textit{Sort}}(\Sigma) &= \{\mathsf{data}, \mathsf{bool}\} \\
\textbf{\textit{Func}}(\Sigma) &= \textbf{\textit{Func}}(\Sigma)(\mathcal{D}) \cup \\
&\quad \textbf{\textit{Func}}(\Sigma(\mathcal{B})) \cup \\
&\quad \{\mathsf{eq}_{\mathsf{data}} : \mathsf{data}^2 \rightarrow \mathsf{bool}, \\
&\quad\ \ \mathsf{if}_{\mathsf{data}} : \mathsf{bool} \times \mathsf{data}^2 \rightarrow \mathsf{data} \\
&\quad \}
\end{aligned}$$

Then, Σ^{stk} is the *stack signature over* Σ where

$$\begin{aligned}
\textbf{\textit{Sort}}(\Sigma^{\mathsf{stk}}) &= \{\mathsf{data}, \mathsf{bool}, \mathsf{stk}\} \\
\textbf{\textit{Func}}(\Sigma^{\mathsf{stk}}) &= \textbf{\textit{Func}}(\Sigma) \cup \\
&\quad \{\mathsf{empty} : \mathsf{stk}, \\
&\quad\ \ \mathsf{push} : \mathsf{data} \times \mathsf{stk} \rightarrow \mathsf{stk} \\
&\quad\ \ {}^*\mathsf{pop} : \mathsf{stk} \rightarrow \mathsf{stk} \\
&\quad\ \ {}^*\mathsf{top} : \mathsf{stk} \rightarrow \mathsf{data} \\
&\quad\ \ \mathsf{isempty} : \mathsf{stk} \rightarrow \mathsf{bool} \\
&\quad\ \ \mathsf{eq}_{stk} : \mathsf{stk}^2 \rightarrow \mathsf{bool} \\
&\quad\ \ \mathsf{if}_{stk} : \mathsf{bool} \times \mathsf{stk}^2 \rightarrow \mathsf{stk} \\
&\quad \}
\end{aligned}$$

$\mathcal{D}^{\mathsf{stk}}$ is the Σ^{stk} expansion of $\mathcal{D}$ where the carrier of sort stk is

$$\mathbb{S} = \mathcal{D}_{\mathsf{stk}} = \text{set of all stacks of data}$$

and all stack operations which are listed above have their usual interpretations.

Remarks 2.4.1.

(1) $\mathcal{D}^{\mathsf{stk}}$ is a standard algebra, and Σ^{stk} includes $\mathsf{eq}_{\mathsf{stk}}$, derived from $\mathsf{eq}_{\mathsf{data}}$.

(2) How should we define pop(empty) and top(empty)? For now, we again use

default values:

$$\begin{aligned} \mathsf{pop}(\mathsf{empty}) &= \mathsf{empty} \\ \mathsf{top}(\mathsf{empty}) &= ? \end{aligned}$$

We must assume there is a *default data item*, that is, a default element of $\mathbb{D}$.

For example

$$\text{For } \mathbb{D} = \begin{cases} \mathbb{B} & \text{take } \mathsf{tt} \text{ (or } \mathsf{ff}) \\ \mathbb{N} & \text{take } 0 \\ \mathbb{Z} & \text{take } 0 \\ A^* & \text{take } \langle\rangle \\ \mathbb{S} & \text{take } \mathsf{empty} \end{cases}$$

More generally we make the following assumption on Σ:

Assumption 2.4.2 (Instantiation). For each sort s of Σ, there is a *closed term* in Σ. Using these *closed terms*, δ_s, as *default values* we can systematically extend all functions in Σ to be *total* on all Σ-algebras.

Discussion 2.4.3 (Default values). Extending the domain of functions by default values is neither an esthetically nor computationally satisfactory. The problem is that default values hide errors. In the next chapter we will introduce a better idea: error algebras.

2.5 Terms over Σ: syntax and semantics

Definition 2.5.1 (Variables).

(1) For each $s \in \boldsymbol{Sort}(\Sigma)$, $\boldsymbol{Var}_s$ is a countable set of variables of sort s : $\mathrm{x}^s, \mathrm{y}^s, \ldots$

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

(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) is a special case of clause (c), with $m = 0$.

Definition 2.5.3 (States over $\underline{A}$). Let $\underline{A}$ be a Σ-algebra. A state over $\underline{A}$ is a family

$$\sigma = (\sigma_s)_{s \in \boldsymbol{Sort}(\Sigma)}$$

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 $\underline{A}$ relative to state σ. The function

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

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

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

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

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

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

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

Notation 2.5.6. We write $\sigma(\mathrm{x}^s)$ for $\sigma_s(\mathrm{x}^s)$ where $\sigma = (\sigma_s)_{s\in \boldsymbol{Sort}(\Sigma)}$

Definition 2.5.7. For $M \subseteq \boldsymbol{Var}(\Sigma)$:

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

i.e. σ and σ' *agree on* M.

Lemma 2.5.8 (Coincidence Lemma). *For any* Σ*-term* t*:*

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

Proof. By structural induction on t. □

Definition 2.5.9 (Closed terms over Σ).

(1) t is *closed* if $\boldsymbol{Var}(t) = \emptyset$

(2) $\boldsymbol{CT}(\Sigma)$ is the set of all *closed* Σ-terms.

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

So if t is closed we can write:

$$[\![t]\!]^A = [\![t]\!]^A\sigma \quad \text{for all } \sigma.$$

Chapter 3

Error Algebras

In this chapter, we will introduce error algebras. Two important properties of such algebras, monotonicity and error-consistency, are discussed.

Some contents are adapted from [TZ88].

3.1 The error value ϵ: algebras A^ϵ of signature Σ^ϵ

Given a standard Σ-algebra

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

let ϵ be a new object or symbol, representing an "error value". For each sort s, let

$$A_s^\epsilon = A_s \cup \{\epsilon\}$$

In particular, $\mathbb{B}^\epsilon = \{\mathtt{tt}, \mathtt{ff}, \epsilon\}$, producing a three-valued logic.

For each $F : u \to s$ in $\boldsymbol{Func}(\Sigma)$, if $u = s_1 \times \cdots \times s_m$, define

$$A^{u,\epsilon} = A^{\epsilon}_{s_1} \times \cdots \times A^{\epsilon}_{s_m},$$

let

$$\underline{A}^{\epsilon} = (A^{\epsilon}_{s_1}, \ldots, F^{A,\epsilon}, \ldots, \epsilon_s, \ldots),$$

and for each $F \in \boldsymbol{Func}(\Sigma)_{u \to s}$, let

$$F^{A,\epsilon} = F^{A^{\epsilon}} : A^{u,\epsilon} \to A^{\epsilon}_{s}$$

be some extension of $F^A : A^u \to A_s$.

Definition 3.1.1 (Strict and consistent extensions). For each $F \in \boldsymbol{Func}(\Sigma)_{u \to s}$, we say:

(1) $F^{A,\epsilon}$ is *strict* over A if $F^{A,\epsilon}(a_1, \ldots, \text{€}, \ldots, a_m) = \text{€}$

(*i.e.* $F^{A,\epsilon}(a_1, \ldots, a_m)$ has value € if any argument is €); and

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

Definition 3.1.2 (Basic error signature and algebra). Let $\underline{A}^{\epsilon}$ be the algebra

$$(A^{\epsilon}_{s_1}, \ldots, A^{\epsilon}_{s_{k-1}}, \mathbb{B}^{\epsilon}; F^{A,\epsilon}_1, \ldots, F^{A,\epsilon}_n, (\text{€}_s)_{s \in \boldsymbol{Sort}(\Sigma)})$$

of signature Σ^{ϵ} where

$$\begin{aligned} \boldsymbol{Sort}(\Sigma^{\epsilon}) &= \boldsymbol{Sort}(\Sigma) \\ \boldsymbol{Func}(\Sigma^{\epsilon}) &= \boldsymbol{Func}(\Sigma) \cup \{(\mathsf{error}_s)_{s \in \boldsymbol{Sort}(\Sigma)}\} \end{aligned}$$

and for all sorts s : $\mathsf{error}_s^A = \epsilon\!\epsilon$

We call

(1) Σ^ϵ the ***basic error signature*** over Σ;

(2) A^ϵ the ***basic error algebra*** over $\underline{A}$.

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

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

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

The logical operators and^ϵ, or^ϵ and not^ϵ which extend and, or and not strictly and consistently, give rise to *weak 3-valued logic* [Kle52, §64].

Example 3.1.4 (Other error algebras on $\mathcal{B}$). The strict or weak 3-valued operators [Kle52] have the following truth tables (read rows before columns):

and	tt	ff	ϵϵ
tt	tt	ff	ϵϵ
ff	ff	ff	ϵϵ
ϵϵ	ϵϵ	ϵϵ	ϵϵ

Table 1: Strict 'and'

or	tt	ff	ee
tt	tt	tt	ee
ff	tt	ff	ee
ee	ee	ee	ee

Table 2: Strict 'or'

We can also define strong (non-strict) versions of there:

strong-and	tt	ff	ee
tt	tt	ff	ee
ff	ff	ff	ff
ee	ee	ff	ee

Table 3: Strong 'and'

strong-or	tt	ff	ee
tt	tt	tt	tt
ff	tt	ff	ee
ee	tt	ee	ee

Table 4: Strong 'or'

Remarks 3.1.5.

(1) All these operators are *commutative.*

(2) We could also define a weak 3-valued *implication*, as well as a strong version:

strong-imp	tt	ff	ee
tt	tt	ff	ee
ff	tt	tt	tt
ee	tt	ee	ee

Table 5: Strong 'imply'

Discussion 3.1.6 (Non-strict semantics). Consider statements:

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

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

Suppose $\mathrm{x} = 0$ (*i.e.* evaluate at σ with $\sigma(\mathrm{x}) = 0$). 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 ee.

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

cand	tt	ff	ee
tt	tt	ff	ee
ff	ff	ff	ff
ee	ee	ee	ee

Table 6: conditional 'and'

cor	tt	ff	ee
tt	tt	tt	tt
ff	tt	ff	ee
ee	ee	ee	ee

Table 7: conditional 'or'

Remarks 3.1.7.

(1) 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 could *add* operators cand or 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* to:

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

Remarks 3.1.8. Consider the data algebra

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

and the basic algebra over A

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

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

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

$$\mathsf{eq}^{A,\epsilon}(x, \text{œ}) = \begin{cases} \text{œ (not tt)} & \text{if } x = \text{œ} \\ \text{œ (not ff)} & \text{otherwise} \end{cases}$$

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

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

where

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

Note that id^{A^ϵ} is a *non-strict* extension of eq^A.

(2) ***Conditional***: Note that $\mathsf{if}^{A,\epsilon}$ *extends* if^{A} by *strictness*. But it is *not* a conditional operation on $\mathcal{D}^\epsilon$:

$$\mathsf{if}^{A,\epsilon}(\mathsf{tt}, d, \epsilon\!\!\epsilon) = \epsilon\!\!\epsilon \quad (\text{not } d)$$

This is a "*weak conditional*" on $\underline{A}^\epsilon$.

We could *adjoin* a *non-strict* (or "strong") *conditional* to $\underline{A}^\epsilon$:

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

where

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

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

$$\mathsf{if}_{\mathsf{ns}}(\mathsf{tt}, d, \epsilon) = d \quad (\text{not } \epsilon\!\!\epsilon).$$

(3) $\underline{A}^\epsilon$ is not (quite) standard, even if $\underline{A}$ is, since:

(a) $\underline{A}^\epsilon$ contains $\mathcal{B}^\epsilon$ instead of $\mathcal{B}$; and

(b) $\underline{A}^\epsilon$ has $\mathsf{if}^{A,\epsilon}$ and $\mathsf{eq}^{A,\epsilon}$ as *strict* extensions of if^{A} and eq^{A}, *i.e.* weak conditional and equality, not the standard interpretations of these symbols on $\underline{A}^\epsilon$.

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.

Now consider the algebra:

$$\mathcal{R}^\epsilon = (\mathbb{R}^\epsilon, \mathbb{Q}^\epsilon, \mathbb{Z}^\epsilon, \mathbb{B}^\epsilon; \text{æ}^R, \text{æ}^Q, \text{æ}^Z, \mathsf{eq}^\epsilon_Q, \mathsf{eq}^\epsilon_Z)$$

This does not have eq^ϵ on $\mathcal{R}^\epsilon$. However we assume we can still distinguish between "real" reals and æ^R, so we add the predicate

$$\mathsf{is\text{-}error}^{R^\epsilon} : \mathbb{R}^\epsilon \to \mathbb{B}^\epsilon$$

where

$$\mathsf{is\text{-}error}^{R^\epsilon}(x) = \begin{cases} \mathsf{tt} & \text{if } x = \text{æ}^R \\ \mathsf{ff} & \text{otherwise} \end{cases}$$

(as well as error_s on the other sorts s)

Remark 3.1.10 (Other operators in $\underline{A}^\epsilon$). Over the basic error algebra $\underline{A}^\epsilon$ we can define operators such as

- cand or cor from $\mathsf{if}_{\mathsf{ns}}$;

- id from eq, is-error and if_{ns};
- is-error from id.

Definition 3.1.11 (Augmented error signature and algebra). Let Σ^ϵ and $\underline{A}^\epsilon$ be the basic error signature and algebra over Σ and $\underline{A}$. The *augmented error signature* $\Sigma^{\epsilon,a}$ and *augmented error* algebra $\underline{A}^{\epsilon,a}$ are formed by adding is-error and if_{ns} for all sorts s of Σ.

Definition 3.1.12 (Strict and consistent error signature and algebra). Let $\underline{A}$ be a Σ-algebra and $\underline{A}^\epsilon$ an error algebra over $\underline{A}$.

(1) $\underline{A}^\epsilon$ is *strict* over $\underline{A}$ iff for all $F \in \boldsymbol{Func}(\Sigma)$, $F^{A,\epsilon}$ is strict over $\underline{A}$.

(2) $\underline{A}^\epsilon$ is *consistent* over $\underline{A}$ iff for all $F \in \boldsymbol{Func}(\Sigma)$, $F^{A,\epsilon}$ is consistent over $\underline{A}$.

Remarks 3.1.13.

(1) The *basic error algebra* over $\underline{A}$ is *consistent* and *strict* over $\underline{A}$.

(2) So is the *augmented error algebra.*

3.2 Semantics of term over Σ^ϵ

Note that $\boldsymbol{State}(\underline{A}) \subset \boldsymbol{State}(\underline{A}^\epsilon)$.

Definition 3.2.1 (States over $\underline{A}^\epsilon$ represented over $\underline{A}$).

For $\sigma \in \boldsymbol{State}(\underline{A}^\epsilon)$, define $\sigma^A \in State(\underline{A})$ by

$$\sigma^A(\mathrm{x}^s) = \begin{cases} \sigma(\mathrm{x}^s) & \text{if } \sigma(\mathrm{x}) \neq \epsilon \\ \delta_s^A & \text{if } \sigma(\mathrm{x}) = \epsilon \end{cases}$$

where δ_s is the default value which exists from the *Instantiation Assumption* 2.4.2. (*I.e.*, we replace error values with default values of the same sort.)

Theorem 1. *Let $\underline{A}^\epsilon$ be an error algebra over $\underline{A}$, $t \in \boldsymbol{Tm}(\Sigma)$ and $\sigma \in \boldsymbol{State}(\underline{A}^\epsilon)$. Then*

(1) *If $\underline{A}^\epsilon$ is strict over $\underline{A}$, and there exists $\mathrm{x} \in \boldsymbol{Var}(t)$ such that $\sigma(\mathrm{x}) = \epsilon$, then*

$$[\![t]\!]^{A^\epsilon}\sigma = \epsilon$$

(2) *If $\underline{A}^\epsilon$ is consistent over $\underline{A}$, and for all $\mathrm{x} \in \boldsymbol{Var}(t)$, $\sigma(\mathrm{x}) \neq \epsilon$, then*

$$[\![t]\!]^{A^\epsilon}\sigma = [\![t]\!]^{A}\sigma^A \neq \epsilon$$

Proof.

(1) By structural induction on t.

(a) $t \equiv \mathrm{x}$:

By assumption, $\sigma(\mathrm{x}) = \epsilon$,

and so,

$$[\![\mathrm{x}]\!]^{A^\epsilon}\sigma = \sigma(\mathrm{x}) = \epsilon.$$

(b) $t \equiv F(t_1, \dots, t_m)$:

since there exists $\mathrm{x} \in \boldsymbol{Var}(t)$, $\sigma(\mathrm{x}) = \epsilon$ and $\boldsymbol{Var}(t) = \bigcup_{i=1}^{n} \boldsymbol{Var}(t_i)$,

we have, for some $i \in \{1, \dots, n\}$:

$$\mathrm{x} \in \boldsymbol{Var}(t_i) \text{ and } \sigma(\mathrm{x}) = \epsilon$$

By induction hypothesis:

$$[\![t]\!]^{A^\epsilon}\sigma = \epsilon$$

Hence,

$$[\![F^{A^\epsilon}(t_1, \dots, t_m)]\!]^{A^\epsilon}\sigma = F^{A^\epsilon}([\![(t_1))]\!]^{A^\epsilon}\sigma, \dots, [\![t_m]\!]^{A^\epsilon}\sigma) = \epsilon$$

since F^{A^ϵ} is strict.

(c) $t \equiv c$:

By definition of ***Terms*** (definition 2.5.2) this is a special case of clause (b), with $m = 0$. Thus,

$$[\![c]\!]^{A^\epsilon}\sigma = \epsilon.$$

(2) By structural induction on t:

(a) $t \equiv \mathrm{x}$:

Using the given fact that $\sigma(\mathrm{x}) \neq \epsilon$:

$$[\![\mathrm{x}]\!]^{A^\epsilon}\sigma = \sigma(\mathrm{x}) = [\![\mathrm{x}]\!]^{A}\sigma^{A}$$

(b) $t \equiv F(t_1, \ldots, t_n)$:

By definition

$$[\![F(t_1, \ldots, t_n)]\!]^{A^\epsilon}\sigma = F^{A,\epsilon}([\![t_1]\!]^{A^\epsilon}\sigma, \ldots, [\![t_n]\!]_{A^\epsilon}\sigma)$$

where by the induction hypothesis:

$$\forall i\ [\![t_i]\!]^{A^\epsilon}\sigma = [\![t_i]\!]^{A}\sigma^A \qquad 0 \le i \le n \qquad (*)$$

continuing on we have:

$$\begin{aligned} F^{A,\epsilon}([\![t_1]\!]^{A^\epsilon}\sigma, \ldots, [\![t_n]\!]^{A^\epsilon}\sigma) &= F^{A,\epsilon}([\![t_1]\!]^{A}\sigma^A, \ldots, [\![t_n]\!]^{A}\sigma^A) \ \ \textit{(by (*))} \\ &= F^{A}([\![t_1]\!]^{A}\sigma^A, \ldots, [\![t_n]\!]^{A}\sigma^A) \ \ (\underline{A}^\epsilon \textit{ is consistent}) \end{aligned}$$

(c) $t \equiv c$:

It is true since it is the special case of clause (b) with $n = 0$.

□

Remarks 3.2.2.

(1) Theorem 1(1) says that the semantics of $\underline{A}^\epsilon$ *extends* the semantics of $\underline{A}$ *strictly.*

i.e. errors propagate or *persist*: once ϵ occurs as value of a subterm it will *persist* as the value of the whole term.

(2) Theorem 1(2) says that the semantics of $\underline{A}^\epsilon$ *extends* the semantics of $\underline{A}$ *consistently.*

Discussion 3.2.3. So there are *two* possible reasons why $[\![t]\!]^{A^\epsilon}\sigma = \text{\ae}$:

(1) From Theorem 1(1): for some $\mathrm{x} \in \boldsymbol{Var}(t) : \sigma(\mathrm{x}) = \text{\ae}$; or

(2) From Theorem 1(2): $\underline{A}^\epsilon$ is *not consistent* over $\underline{A}$, so some function at some argument returns $\text{\ae}$ instead of a *default value.*

Corollary 3.2.4. If $\underline{A}^\epsilon$ is consistent over $\underline{A}$, $t \in \boldsymbol{Tm}(\Sigma)$ and $\sigma \in \boldsymbol{State}(\underline{A})$ then $[\![t]\!]^{A^\epsilon}\sigma = [\![t]\!]^{A}\sigma$

Proof. From Theorem 1(2), since if $\sigma \in \boldsymbol{State}(\underline{A})$, then $\sigma = \sigma^A$. □

Corollary 3.2.5. If A^ϵ is consistent over $\underline{A}$, and $t \in \boldsymbol{CT}(\Sigma)$ then

$$[\![t]\!]^{A^\epsilon} = [\![t]\!]^{A}.$$

Proof. Immediate from Theorem 1(2). □

Discussion 3.2.6. Theorem 1 works only assuming $\underline{A}^\epsilon$ is *consistent* and *strict.* How important or desirable are these properties?

• ***Strictness***: This does not hide *errors*: "*errors* propagate". This formalizes the idea of 'GIGO'. But we may prefer operators such as cand, cor and $\mathsf{if}_{\mathsf{ns}}$ as (non-strict) extensions of and, or and if^A. We will return to this.

• ***Consistency***: Consider algebras containing functions with *default values*, for example:

(1) $\mathcal{N}' = (\mathbb{N},\ \mathbb{B};\ 0,\ \mathsf{suc},\ \mathsf{pred}, \ldots)$ where $\mathsf{pred}(0) = 0$.

Now define:

(2) $\mathcal{N}^\epsilon = (\mathbb{N}^\epsilon,\ \mathbb{B}^\epsilon;\ 0,\ \mathsf{suc}^\epsilon,\ \mathsf{pred}^\epsilon, \ldots)$ but with $\mathsf{pred}^\epsilon(0) = \epsilon$.

So now pred^ϵ is not a consistent extension of pred!

Similarly, one can have non-consistent but strict extensions to remove default values, with, for example,

div in $\mathcal{N}$ or $\mathcal{Z}$; or

pop and top in $\mathcal{D}^{\mathsf{stk}}$.

So *consistency* is not always desirable.

Remarks 3.2.7.

(1) From now on, we will evaluate Σ-terms and Σ^ϵ-terms over $\underline{A}^\epsilon$, that is, we let $\sigma_1,\ \sigma_2, \ldots$ range over $\boldsymbol{State}(\underline{A}^\epsilon)$, and let $[\![t]\!]\sigma$ mean $[\![t]\!]^{A^\epsilon}\sigma$ (even if $t \in \boldsymbol{Tm}(\Sigma)$).

(2) We can think of $\sigma(\mathrm{x}) = \epsilon$ as "σ is *unspecified* at x" or "x is *not yet initialized* in state σ".

Discussion 3.2.8. We are looking for (computationally meaningful) conditions on *error algebras*, weaker than

(1) *strictness*; and

(2) *consistency.*

For (1) we propose *monotonicity*; and for (2) we propose *error-consistency.* We discuss both of these in the following two sections.

3.3 Monotonicity

First define a simple *partial order* on each carrier A_s^ϵ of $\underline{A}^\epsilon$.

Definition 3.3.1. For all $x, y \in A_s^\epsilon$

$$x \sqsubseteq y \iff x = y \text{ or } x = \epsilon^s$$

Definition 3.3.2.

(1) Let $F : s_1 \times \cdots \times s_m \to s$ be in $\boldsymbol{Func}(\Sigma)$. Then $F^{A,\epsilon}$ is *monotonic* over $\underline{A}$ iff

$\forall x_1, y_1 \in A_{s_1}^\epsilon, \ldots, x_m, y_m \in A_{s_m}^\epsilon$:

$$x_1 \sqsubseteq y_1 \text{ and} \ldots \text{and } x_m \sqsubseteq y_m \implies F^{A,\epsilon}(x_1, \ldots, x_m) \sqsubseteq F^{A,\epsilon}(y_1, \ldots, y_m).$$

(2) $\underline{A}^\epsilon$ is *monotonic* over $\underline{A}$ iff for all $F \in \boldsymbol{Func}(\Sigma)$, $F^{A,\epsilon}$ is *monotonic* over $\underline{A}$.

Lemma 3.3.3. *Strictness* $\implies$ *monotonicity*

Proof. From definitions of strictness (Definition 3.1.1) and monotonicity (Definition 3.3.2). □

An equivalent characterization of monotonicity is:

Lemma 3.3.4. *If $F : u \to s$ is in $\textbf{\textit{Func}}(\Sigma)$, then $F^{A,\epsilon}$ is monotonic over $\underline{A}$ iff the following holds:*

for any $\bar{a} \in A^u$, where $\bar{a} \equiv (a_1, \ldots, \epsilon, \ldots, a_m)$ (including at least one ϵ),

if $F^{A,\epsilon}(\bar{a}) = y \neq \epsilon$, then replacing ϵ in $\bar{a}$ by any other argument will not change the output y.

Proof. Immediate from Definitions 3.3.1 and 3.3.2 (2). □

Discussion 3.3.5. The computational significance of *strictness* is that it does *not* "*hide errors*" in the input. *Monotonicity* is a more liberal concept than strictness — it may hide *errors* in the input, but only if they *do not affect* the output, that is, if they are *irrelevant* to output.

Example 3.3.6. The operators if, cand, cor, strong-and, strong-or are monotonic. But is-error and id are not.

Remark 3.3.7 (Semantics of bounded quantifiers). Consider *bounded quantification over integers.* Then interpret

$$\forall k_{a \le k \le b}\, P(k) \quad (a, b \in \mathbb{Z})$$

as

$$P(a) \text{ cand } P(a+1) \text{ cand} \ldots \text{cand } P(b).$$

and interpret

$$\exists k_{a<k\leq b}\, P(k) \quad (a, b \in \mathbb{Z})$$

as

$$P(a) \text{ cor } P(a+1) \text{ cor...cor } P(b).$$

Definition 3.3.8. For $M \subseteq \boldsymbol{Var}(\Sigma)$,

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

iff for all x $\in M$, $\sigma(\text{x}) \sqsubseteq \sigma'(\text{x})$

Proposition 3.3.9. *$\sigma \approx \sigma'(rel\ M) \Longleftrightarrow \sigma \underset{\sim}{\sqsubset} (rel\ M)$ and $\sigma' \underset{\sim}{\sqsubset} \sigma(rel\ M)$*

Proof. Clear from Definitions 3.3.1 and 3.3.8. □

Remarks 3.3.10.

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

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

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

Theorem 2 (Monotonicity for $\boldsymbol{Tm}(\Sigma)$). *Suppose $\underline{A}^\epsilon$ is monotonic over $\underline{A}$. Then for all $t \in \boldsymbol{Tm}(\Sigma)$, and $\sigma, \sigma' \in \boldsymbol{State}(\underline{A}^\epsilon)$:*

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

Proof. By structural induction on t. $\square$

Remark 3.3.11. As a Corollary we get: if $\underline{A}^\epsilon$ is monotonic over $\underline{A}$ then

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

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

Corollary 3.3.12. If A^ϵ is monotonic over A, then for all $t \in \boldsymbol{Tm}(\Sigma)$ and $\sigma \in \boldsymbol{State}(\underline{A}^\epsilon)$:

$$[\![t]\!]\sigma \neq \epsilon \implies \forall \sigma' \underset{\sim}{\sqsupset} \sigma\ (rel\ \boldsymbol{Var}(t)),\ [\![t]\!]\sigma' = [\![t]\!]\sigma$$

Remark 3.3.13. This again has the idea that an *error* in an input may be hidden, provided it is "irrelevant".

3.4 Error-consistency

Definition 3.4.1.

(1) Let $\underline{A}$ be Σ-algebra, $\underline{A}^\epsilon$ an *error* algebra over $\underline{A}$ and

$$F : s_1 \times \cdots \times s_m \to s \text{ in } \boldsymbol{Func}(\Sigma)$$

Then $F^{A,\epsilon}$ is *error-consistent* over $\underline{A}$ iff for all $a_1 \in A_{s_1}, \ldots, a_m \in A_{s_m}$:

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

(2) $\underline{A}^\epsilon$ is *error-consistent* over $\underline{A}$ if and only if for all $F \in Func(\Sigma)$, $F^{A,\epsilon}$ is *error-consistent* over $\underline{A}$.

An equivalent characterization of *error-consistency* is:

Lemma 3.4.2. *$F^{A,\epsilon}$ is error-consistent over $\underline{A}$ if and only if for all $a_1 \in A_{s_1}, \ldots, a_m \in A_{s_m}$:*

$$F^{A,\epsilon}(a_1, \ldots, a_m) = F^A(a_1, \ldots, a_m) \textit{ or } \mathbb{e}.$$

Notes.

(1) In the case where $\mathbb{e}$ is the result, think of this as giving an *error message* instead of a default value.

(2) Every $F^{A,\epsilon}$ considered so far is *error-consistent.*

Lemma 3.4.3. *consistency* $\Longrightarrow$ *error-consistency*

Proof. From definitions of *consistency* (Definition 3.1.1) and *error-consistency* (Definition 3.4.1). □

Theorem 3 (Error-consistency for $\boldsymbol{Tm}(\Sigma)$). *Let $\underline{A}^\epsilon$ be error-consistent and monotonic over $\underline{A}$, and let $t \in \boldsymbol{Tm}(\Sigma)$. Then:*

(1) *If $\sigma \in \boldsymbol{State}(\underline{A}^\epsilon)$ and for all $\mathrm{x} \in \boldsymbol{Var}(t)$, $\sigma(\mathrm{x}) \neq \mathbb{e}$ then*

$$[\![t]\!]^{A^\epsilon}\sigma \sqsubseteq [\![t]\!]^A\sigma^A;$$

(2) *If* $\sigma \in$ ***State***$(\underline{A})$ *then*

$$[\![t]\!]^{A^\epsilon}\sigma \sqsubseteq [\![t]\!]^{A}\sigma;$$

(3) *If t is closed then*

$$[\![t]\!]^{A^\epsilon} \sqsubseteq [\![t]\!]^{A}.$$

Proof.

(1) By structural induction on t:

(a) $t \equiv \mathbf{x}$:

Using the given fact that $\sigma(\mathbf{x}) \neq \epsilon$:

$$[\![\mathbf{x}]\!]^{A^\epsilon}\sigma = \sigma(\mathbf{x}) = [\![\mathbf{x}]\!]^{A}\sigma^{A}$$

and so $[\![t]\!]^{A^\epsilon}\sigma \sqsubseteq [\![t]\!]^{A}\sigma^{A}$

(b) $t \equiv F(t_1, \ldots, t_n)$:

By definition

$$[\![F(t_1, \ldots, t_n)]\!]^{A^\epsilon}\sigma = F^{A,\epsilon}([\![t_1]\!]^{A^\epsilon}\sigma, \ldots, [\![t_n]\!]_{A^\epsilon}\sigma)$$

where by the induction hypothesis:

$$\forall i\ [\![t_i]\!]^{A^\epsilon}\sigma \sqsubseteq [\![t_i]\!]^{A}\sigma^{A} \qquad 0 \leq i \leq n \tag{$*$}$$

continuing on we have:

$$\begin{aligned} F^{A,\epsilon}([\![t_1]\!]^{A^\epsilon}\sigma,\ldots,[\![t_n]\!]^{A^\epsilon}\sigma) &\sqsubseteq F^{A,\epsilon}([\![t_1]\!]^{A}\sigma^A,\ldots,[\![t_n]\!]^{A}\sigma^A) && \text{(monotonicity, } (*)\text{)} \\ &\sqsubseteq F^{A}([\![t_1]\!]^{A}\sigma^A,\ldots,[\![t_n]\!]^{A}\sigma^A) && \text{(error-consistency)} \end{aligned}$$

(2) Immediate, since $\sigma^A = \sigma$ if $\sigma \in \boldsymbol{State}(\underline{A})$.

(3) Directly from (1) and Corollary 2.5.10.

□

Chapter 4

Semantic of Improper Tables using Error Algebras

In this chapter we present a semantics for *function tables*, using error 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.

The first application of this technique was 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 [Par94, 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 tabular notation 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 [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.

In this chapter we extend the semantic theory of [Zuc96] by defining a uniform semantics for proper and improper tables, using error algebras. The actual error algebras are not specified precisely, except for the assumption that they are standard, monotonic and error-consistent.

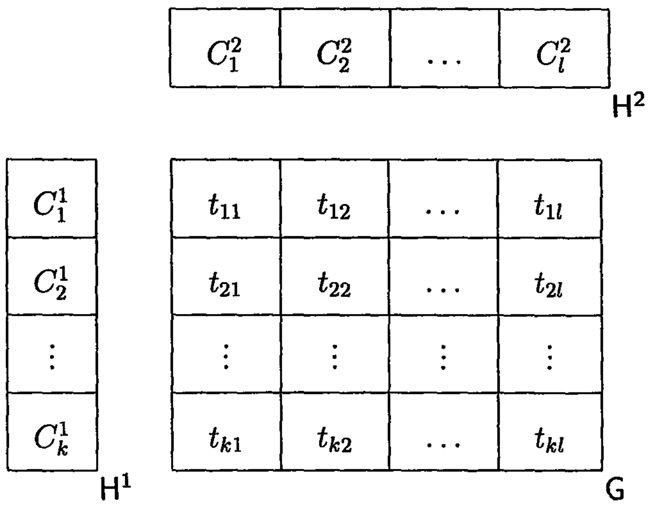
The approach take here is not to divide tables into proper and improper subclasses (as in [Zuc96]) but to consider, for any table T at any particular state σ, whether T is proper or improper at σ. (The answer will vary, in general, with σ). It is also found necessary to broaden the concept of "properness"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

For convenience, we restrict our attention to 1- and 2-dimensional tables. The theory presented here can be easily generalised to the case of n-dimensional tables, as in [Zuc96].

4.1 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 4.1.1 (A two dimensional normal table).



H²

C_1^2	C_2^2	$\dots$	C_l^2

H¹

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

G

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

Table 8

In Table 8, the headers H^1 and H^2 contain *conditions* C_i^1 $(1 \leq i \leq h)$ and C_j^2 $(1 \leq j \leq l)$ respectively. These are boolean-valued expressions over Σ, extended e.g. by bounded quantifiers (Remarks 3.3.7). The cells (i, j) of the grid G of T contain terms $t_{i,j}$, all of the same Σ-sort.

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 evaluate to $\mathbb{tt}$(at that state), assuming T is proper (see Remark 4.1.2 below). What if T is not proper? More generally, how may *errors* come in the output?

There are three ways in which the output of a normal table T can be $\mathbb{ee}$ for a given state σ:

(i) T is not proper at σ;

(ii) each header is *proper* (at σ) in the sense of having one condition that evaluates to $\mathbb{tt}$, but one of the non-true condition evaluates to $\mathbb{ee}$ instead of $\mathbb{ff}$;

(iii) each header is *proper* (at σ) with a true condition (say) $C_{i_0}^1$ and $C_{j_0}^2$, but $[\![t_{i_0 j_0}]\!]\sigma = \mathbb{ee}$.

Remarks 4.1.2. Above, by "proper" we mean:

(1) there is a *unique* i such that $[\![C_i^1]\!]\sigma = \text{tt}$, and for all $i' \neq i$, $[\![C_{i'}]\!] = \text{ff}$; and

(2) there is a *unique* j such that $[\![C_j^2]\!]\sigma = \text{tt}$, and for all $j' \neq j$, $[\![C_{j'}]\!] = \text{ff}$.

Later (Definition 4.1.7) we will give a different (more general, and more appropriate) definition of 'properness', and we will call the above "strict properness".

Remarks 4.1.3. 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 4.1.4. Here are two possible strategies for evaluating improper tables at a given state σ, assuming all headers have at least one condition which evaluates to true:

(1) Take the leftmost (or topmost) condition which evaluates to true (like the "case" statement in Pascal). But this is dangerous since 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) Give the output value as $\textit{\oe}$. So define the table function as:

$$f_T^A : A \to A \cup \{\textit{\oe}\}.$$

This is a better idea, but it is still not ideal, as we will see (Remark 4.1.19).

4.1.1 Properness

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

Definition 4.1.5. Let C be a condition. A tuple $(C_1, \ldots, C_n)$ of conditions is called

- *disjoint relative to* C at state σ over $\boldsymbol{Var}(C, C_1, \ldots, C_n) \iff$ if σ satisfies C, then σ satisfies *at most one of* $C_1, \ldots, C_n$.

- *universal relative to* C at state σ over $\boldsymbol{Var}(C, C_1, \ldots, C_n) \iff$ if σ satisfies C, then σ satisfies *at least one of* $C_1, \ldots, C_n$.

- *strictly proper relative to* C at state σ if it is both disjoint and universal relative to C at σ.

Equivalently, $(C_1, \ldots, C_n)$ is strictly proper relative to C at σ over $\boldsymbol{Var}(C, C_1, \ldots, C_n) \iff$ if σ satisfies C, then σ satisfies *exactly one of* $C_1, \ldots, C_n$.

An important special case of the above concepts is given by the following.

Definition 4.1.6. A tuple $(C_1, \ldots, C_n)$ of conditions is called (respectively) *disjoint*, *universal* or *strictly proper* at σ if it is (respectively) disjoint, universal or strictly proper relative to the condition true at σ.

Equivalently, $(C_1, \ldots, C_n)$ is *strictly proper* at σ over $\boldsymbol{Var}(C_1, \ldots, C_n)$ iff σ satisfies *exactly one of* $C_1, \ldots, C_n$.

Note that these concepts (disjointness, universality and strict properness) are all relative to the Σ-algebra A. For example, the tuple $(x < 0,\, x = 0,\, 0 < x)$ is not strictly proper in all algebras (of the appropriate signature), but only in those algebras in which the interpretation of '$<$' satisfies the trichotomy law. On the other hand, the tuple $(x < 0,\, x \not< 0)$ is strictly proper in all algebras (of the appropriate signature).

Let T be a normal table.

Definition 4.1.7 (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^1_i in header H^1 and C^2_j in header H^2 are true at σ.

Remarks 4.1.8.

(1) Condition (ii) says that the values agree on overlapping conditions given by non-disjoint headers.

(2) The important concept of a table T is not whether it is proper (at all states), but whether it is proper *at a particular state.* The reason for this is that it is not (in general) effectively decidable whether a table is proper at all states.

(3) Of course, some tables are proper at all states, for example:

Example 4.1.9 (A proper normal table).

$y \leq 10$	$y = 10$	$y > 10$

H^2

$x \geq 0$
$x < 0$

H^1

$x + y$	$10 + x$	$-y^2$
$x - y$	$x - 10$	$x - y$

G

Table 9

Table 9 is an example of a proper normal table (at all states). Suppose $\sigma(y) = 10$. Then conditions $(y \leq 10)$ and $(y = 10)$ in header H^2 are satisfied, and the value agrees on overlapping conditions. Note that this table, and most of the following examples of tables, are based on the signature $\Sigma(\mathcal{Z}^B)$ of the standardised algebra of integer (or some expansion of it).

Example 4.1.10 (A strictly proper normal table).

$y = 10$	$y > 10$	$y < 10$

H^2

$x \geq 0$
$x < 0$

H^1

0	y^2	$-y^2$
x	$x + y$	$x - y$

G

Table 10

Note that each header is strictly proper (at all states).

4.1.2 Semantics of normal tables

Definition 4.1.11. 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^1_i and C^2_j hold at σ. Then the *meaning of T relative to σ* is

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

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

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

Definition 4.1.12. 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 4.1.13. 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 $\mathtt{x}$, T *names* or *defines* a *table function symbol*

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

with interpretation on A

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

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

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

Discussion 4.1.14 (Semantics of improper tables). We turn to the semantics of tables that are improper at certain states. The semantics in [Zuc96] only works when T is proper at a given state σ. Thus, we must use another method. The two strategies mentioned in Remarks 4.1.4. are not satisfactory (see Remark 4.1.19). A better idea is to define an *improper table function* using error algebras which are *monotonic* and *error-consistent.*

We must first modify the definition (4.1.5) of universality of condition tuples evaluated over error algebras.

Definition 4.1.15 (Universality for headers over error algebras). A tuple of conditions $(C_1, \ldots, C_n)$ is said to be *universal* at $\sigma \in \textbf{\textit{State}}(A^\epsilon)$ if:

(1) for some i, $[\![C_i]\!]^{A^\epsilon}\sigma = \mathtt{tt}$;

(2) for all j, $[\![C_j]\!]^{A^\epsilon}\sigma \neq \epsilon$.

Remarks 4.1.16.

(1) the definition (4.1.7) of properness of a normal table T at a state σ over A^ϵ now presupposes that none of the header conditions evaluates to ϵ.

(2) However a term t_{ij} in the grid of T may very well evaluate to ϵ at σ without causing T to be improper. This is analogous to the (non-strict) semantics for the conditional

$$\mathsf{if}_{\mathsf{ns}}(t^{\mathsf{bool}}, t_1^s, t_2^s)$$

We now extend the definition (4.1.11) of $[\![T]\!]^A\sigma$ in the case that T is improper at σ:

Definition 4.1.17 (Semantics of table over Σ^ϵ). Let T be a normal table over Σ^ϵ, and σ a state over T in A^ϵ. We define $[\![T]\!]^{A^\epsilon}\sigma$ as follows:

Case 1: T is proper at σ. Then $[\![T]\!]^{A^\epsilon}\sigma$ is as in Definition 4.1.11.

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

Definition 4.1.18 (Table function). Let A^ϵ be an error algebra over A. Let T be a table, $\bar{\mathrm{x}} \equiv (\mathrm{x}_1, \ldots, \mathrm{x}_m)$ be any list of variables which covers T with $\mathrm{x}_i : s_i$ for $i = 1, \ldots, m$. For $a_1 \in A_{s_1}, \ldots, a_m \in A_{s_m}$ let σ be a state over T satisfying $\sigma(\mathrm{x}_i) = a_i$ for $i = 1, \ldots, m$. Then:

(1) if T is proper at σ (according to Definition 4.1.7 applied to A^ϵ), define $f^{A,\epsilon}_{T,\mathrm{x}}(a_1, \ldots, a_m)$ as above (using A^ϵ instead of A).

(2) if T is not proper at σ, then

$$f^{A,\epsilon}_{T,\mathrm{x}}(a_1, \ldots, a_m) = \text{\ae}.$$

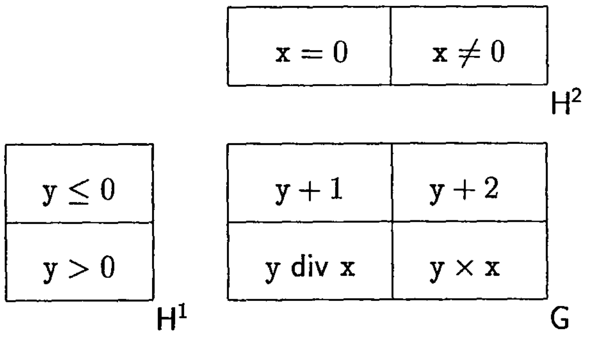
Remarks 4.1.19 (Semantics with error algebras).

(1) If T is improper at σ, then

$$f^{A,\epsilon}_{T,\mathrm{x}}(a_1, \ldots, a_m) = [\![T]\!]^{A^\epsilon}\sigma = \text{\ae}.$$

(2) But even if T is proper at σ, this semantics may give an output of ϵ, because A^ϵ is *error-consistent* rather than *consistent.* This method is better since it shows error values which would be hidden by default values. To illustrate this:

Example 4.1.20. Let T be the table



	$x = 0$	$x \neq 0$
$y \leq 0$	$y + 1$	$y + 2$
$y > 0$	y div x	$y \times x$

Table 11

At the state $(x = 0, y = 1)$, the value of T is :

(i) 0 according to the "default semantics" of Example 2.1.10;

(ii) ϵ according to Definition 4.1.18(2).

The latter output is more desirable, since it does not hide error by default values.

The general situation here is shown by the following theorem.

Theorem 4. *Let A^ϵ be an error algebra which is monotonic and error-consistent. Let T be a normal table, with* $\textbf{\textit{Var}}(T) \subseteq \mathrm{x}$. *Then $f^{A,\epsilon}_{T,\mathrm{x}}$ is*

(1) *monotonic, and*

(2) *error-consistent.*

Proof.

(1) *Monotonicity*: It is sufficient to show:

$$\sigma_1 \underset{\sim}{\sqsubseteq} \sigma_2 \Longrightarrow [\![T]\!]\sigma_1 \sqsubseteq [\![T]\!]\sigma_2 \qquad (*)$$

If $[\![T]\!] = \mathbb{e}$, this is trivial. So we may assume: $[\![T]\!] \neq \mathbb{e}$.

Hence T is proper at σ, i.e.,

(a) the headers of T are both universal at σ_1,

(b) for all i, j such that:

$$[\![C_j^2]\!]\sigma_1 = \mathbf{tt}, \quad [\![t_{ij}]\!]\sigma_1 = a \ (say) \ \neq \mathbb{e},$$

and so,

$$[\![T]\!]\sigma_1 = a.$$

By *monotonicity* of A^ϵ, for all i, j:

$$[\![C_i^1]\!]\sigma_2 = [\![C_i^1]\!]\sigma_1 \quad \text{and} \quad [\![C_j^2]\!]\sigma_2 = [\![C_j^2]\!]\sigma_1.$$

Hence the headers of T are also universal at σ_2 and for all i, j such that:

$$[\![C_i^1]\!]\sigma_2 = [\![C_j^2]\!]\sigma_2 = \mathbf{tt},$$

$$[\![t_{i,j}]\!] = a,$$

and so

$$[\![T]\!]\sigma_2 = a = [\![T]\!]\sigma_1.$$

proving (*).

(2) *Error-consistency*: Let σ be a state over A. It is sufficient to show:

$$[\![T]\!]^{A,\epsilon}\sigma \sqsubseteq [\![T]\!]^{A}\sigma \tag{**}$$

There are two cases.

(a) T is proper at σ.

Then the header H^1 and H^2 are universal, w.r.t. σ. By Theorem 3 for $i = 1, \ldots, k$ and $j = 1, \ldots, l$:

$$[\![C_i^1]\!]^{A,\epsilon}\sigma \sqsubseteq [\![C_i^1]\!]^{A}\sigma \quad \text{and} \quad [\![C_j^2]\!]^{A,\epsilon}\sigma \sqsubseteq [\![C_j^2]\!]^{A}\sigma. \tag{* * *}$$

Then by (***), for all i, j

$$[\![C_i^1]\!]^{A,\epsilon}\sigma = [\![C_i^1]\!]^{A}\sigma \quad \text{and} \quad [\![C_j^2]\!]^{A,\epsilon}\sigma = [\![C_j^2]\!]^{A}\sigma. \tag{* * **}$$

Since T is *proper* at σ, there exists i, j:

$$[\![C_i^1]\!]^{A}\sigma = \mathtt{tt} \quad \text{and} \quad [\![C_j^2]\!]^{A}\sigma = \mathtt{tt}.$$

Then for this i, j

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

By (****) for the same i, j,

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

By Theorem 3:

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

Hence,

$$[\![T]\!]^{A,\epsilon}\sigma = [\![T]\!]^{A}\sigma.$$

(b) T is improper at σ. Then

$$[\![T]\!]^{A,\epsilon}\sigma = \epsilon\!\epsilon.$$

Combining (a) and (b), we conclude (**)

$\square$

4.2 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).

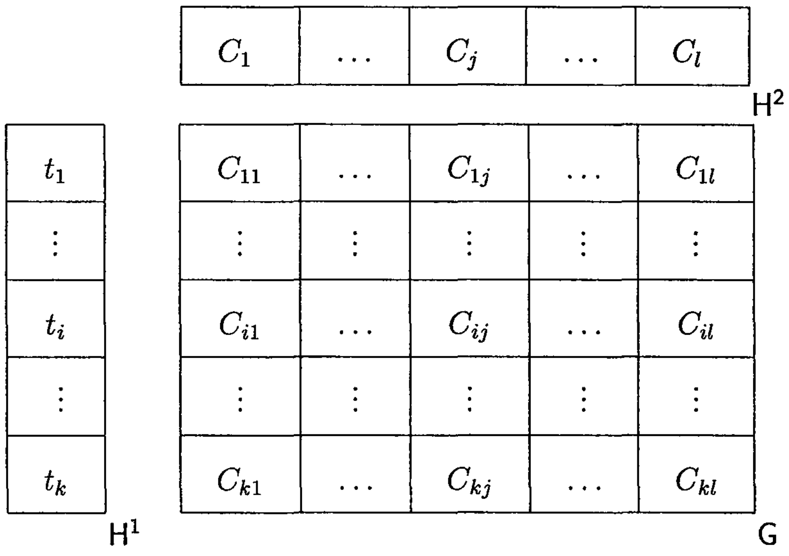
(1) One of its headers (H^1), is the *value header.* It contains *terms,* all of the same sort, instead of conditions. The other header (H^2) the *condition header* , contains conditions as before.

(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 the cell (i, j) whose entry C_{ij} has the value 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 definitions. Let T be an inverted table as follows:

Example 4.2.1 (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}

Table 12

Definition 4.2.2 (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. $\sigma \models C_j$, and all i s.t. $\sigma \models C_{ij}$, the value of $[\![t_{ij}]\!]\sigma$ is the same.

Example 4.2.3 (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 13

Table 13 is an example of a proper inverted table. Notice that the columns are *not* proper (for some values of y), but *are* proper (in fact, strictly proper) relative to the corresponding conditions in the column header H^2.

Definition 4.2.4 (Semantics of inverted tables). The semantic function $[\![T]\!]^A\sigma$ and the table function $f_{T,\mathbf{x}}^{A,\epsilon}$ are defined analogously to Definitions 4.1.11 and 4.1.18, according to the above informal operational semantics.

4.3 Transformations of tables

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

Definition 4.3.1 (Semantic equivalence of tables over A^ϵ). Let T_1 and T_2 be two tables. T_1 and T_2 are *semantically equivalent on* A^ϵ (written $T_1 \approx_{A^\epsilon} T_2$) iff for all states σ over $\boldsymbol{Var}(T_1, T_2)$ in A^ϵ, $[\![T_1]\!]^{A^\epsilon}\sigma = [\![T_2]\!]^{A^\epsilon}\sigma$.

Remark 4.3.2. 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

$$\varphi : \tau \to \tau'$$

of tables from one class τ to another class τ'. These transformations must satisfy the following three properties:

(1) φ is *semantics preserving*, in the sense that, if $T \in \tau$ is improper, then so is $\varphi(T)$, and $\varphi(T) \approx T$.

(2) φ is *effective* or *computable.*

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

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

4.4 Inverting a normal table

We have two methods (or algorithms) [Zuc96] for transforming a normal table to a semantically equivalent inverted one.

We illustrate the first inversion method with a simple example. Consider the case of a 2-dimensional 3×3 normal table T, given in Table 14.

Example 4.4.1 (A normal table).

H^2

	C^2_1	C^2_2	C^2_3
C^1_1	t_{11}	t_{12}	t_{13}
C^1_2	t_{21}	t_{22}	t_{23}
C^1_3	t_{31}	t_{32}	t_{33}

H^1 G

Table 14

T is "inverted along dimension 1" to produce an inverted table (Table 15) 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 in its grid is *small.* Suppose, e.g., the grid in Table 14 contains only 2 terms, 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 4.4.2 (Inversion of Table 14: Method 1).

C_1^2	C_2^2	C_3^2

H^2

t_{11}
t_{21}
t_{31}
t_{11}
t_{21}
t_{31}
t_{11}
t_{21}
t_{31}

H^1

C_1^1	false	false
C_2^1	false	false
C_3^1	false	false
false	C_1^1	false
false	C_2^1	false
false	C_3^1	false
false	false	C_1^1
false	false	C_2^1
false	false	C_3^1

G

Table 15

Example 4.4.3 (A special case of Table 14).

	C^2_1	C^2_2	C^2_3
C^1_1	t_1	t_1	t_2
C^1_2	t_2	t_1	t_2
C^1_3	t_1	t_2	t_2

H^2

H^1 G

Table 16

Example 4.4.4 (Inversion of Table 16: Method 2).

	C^2_1	C^2_2	C^2_3
t_1	$C^1_1 \vee C^1_3$	$C^1_1 \vee C^1_2$	false
t_2	C^1_2	C^1_3	$C^1_1 \vee C^1_2 \vee C^1_3$

H^2

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

Table 17

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

Lemma 4.4.5. *Let T be an normal table, and $\tilde{T}$ the inverted table obtained from T by Method 1 or 2. Then*

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

Proof. We show

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

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

(1) T is proper at σ.

The proof is similar as for Theorem 2 (1) in [Zuc96].

(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 $\tilde{T}$. If H^1 is not universal in T, then all the columns in the grid of $\tilde{T}$ 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 $\tilde{T}$.

□

Remarks 4.4.6. Suppose the normal table T (Table 15) 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 17) is *strictly proper.* Hence Lemma 4.4.5 does not hold with "properness" replaced by "strict properness". This explains our new, more liberal, definition of properness.

Theorem 5. *Suppose T is a normal table, and $\tilde{T}$ is the inverted table obtained from T by Method 1 or Method 2. Then*

$$\tilde{T} \approx_{A^\epsilon} T.$$

Proof. There are two cases.

(1) T is a proper normal table. Similar to Theorem 2 in section 8 of [Zuc96].

(2) T is an improper table.

For all $a_1 \in A_{s_1}, ..., a_m \in A_{s_m}$, $\sigma(x_i) = a_i$ for $i = 1, ..., m$, we have

$$f^{A,\epsilon}_{T,\mathbf{x}}(a_1, ..., a_m) = [\![T]\!]^{A^\epsilon}\sigma = \epsilon$$

by definition of the *improper table function* (Definition 4.1.18(2)).

From Lemma 4.4.5 the inverted table $\tilde{T}$ is also improper. Then we have

$$f^{A,\epsilon}_{\tilde{T},\mathbf{x}}(a_1, ..., a_m) = [\![\tilde{T}]\!]^{A^\epsilon}\sigma = \epsilon.$$

Thus,

$$[\![\tilde{T}]\!]^{A^\epsilon}\sigma = [\![T]\!]^{A^\epsilon}\sigma,$$

and so

$$\tilde{T} \approx_{A^\epsilon} T.$$

□

4.5 Normalising an inverted Table

We now consider the transformation of an inverted table to a normal one. The situation is less satisfactory since the normal table produced here is one-dimensional.

We consider the 2-dimensional 3 × 2 inverted table shown as Table 18, with value header H^1.

Example 4.5.1 (Two-dimensional table).

	C^2_1	C^2_2
t_1	C_{11}	C_{12}
t_2	C_{21}	C_{22}
t_3	C_{31}	C_{32}

H^2 H^1 G

Table 18

This can be normalised to a 1-dimensional table, shown as Table 19.

Example 4.5.2 (Normalisation of Table 18).

$\hat{H}^1$	$\hat{G}$
$(C^2_1 \wedge C_{11}) \vee (C^2_2 \wedge C_{12})$	t_1
$(C^2_1 \wedge C_{21}) \vee (C^2_2 \wedge C_{22})$	t_2
$(C^2_1 \wedge C_{31}) \vee (C^2_2 \wedge C_{32})$	t_3

Table 19

Table 18 can also be normalized to Table 20, by "splitting disjunctions" in the conditions.

Example 4.5.3 (Another normalisation of Table 18).

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

Table 20

Lemma 4.5.4. *Let $\hat{T}$ be the normal table obtained from T by the method of either Table 19 or Table 20. Then*

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

Proof.

By extending the method of Theorem 3(1) in [Zuc96] for proper tables, as in Lemma 4.4.5(2).

□

Theorem 6. *Suppose T is an inverted table, and $\hat{T}$ is the normal table obtained from T as above. Then:*

$$\hat{T} \approx_{A^\epsilon} T.$$

Proof. Similar to Theorem 5. □

Remark 4.5.5. Here also, we see that properness and improperness are both preserved with our new definition of properness (see Remark 4.4.6).

4.6 Comparison with the logic of Parnas

In [Par93] there are two types of expressions:

(1) *Terms*, such as the expressions in the grid of a normal table.

(2) *Predicate expressions*, such as the boolean-valued *conditions* of the table headers.

The *semantics of terms* (including boolean-valued terms) is *3-valued*, essentially like ours. But the semantics of conditions is *2-valued.*

An *atomic condition* $C \equiv f(t_1, \ldots, t_n)$, where $f \in \boldsymbol{Func}(\Sigma)$ of type $s_1 \times \cdots \times s_n \to$ bool is evaluated as:

$$[\![C]\!]\sigma = \begin{cases} \text{tt} & \text{if } [\![t_1]\!]\sigma \neq \epsilon, \ldots, [\![t_n]\!]\sigma \neq \epsilon \text{ and } C_A([\![t_1]\!]\sigma, \ldots, [\![t_n]\!]\sigma) = \text{tt} \\ \text{ff} & \text{if } [\![t_1]\!]\sigma \neq \epsilon, \ldots, [\![t_n]\!]\sigma \neq \epsilon \text{ and } C_A([\![t_1]\!]\sigma, \ldots, [\![t_n]\!]\sigma) = \text{ff} \\ \text{ff} & \text{if } [\![t_1]\!]\sigma = \epsilon \text{ or} \ldots \text{or } [\![t_n]\!]\sigma = \epsilon \end{cases}$$

This gives a *2-valued* semantics for boolean conditions, which is *non-monotonic.* Note that the *equality* predicate is then also *non-monotonic.*

Example 4.6.1. Say $C \equiv (\mathrm{x} = 0)$, then for

$$\sigma_1(\mathrm{x}) = \text{\ae} \quad \text{and} \quad \sigma_2(\mathrm{x}) = 0,$$

we get

$$[\![C]\!]\sigma_1 = \mathsf{ff} \quad \text{and} \quad [\![C]\!]\sigma_2 = \mathsf{tt},$$

and so

$$\sigma_1 \sqsubseteq \sigma_2 \quad \text{but} \quad [\![C]\!]\sigma_1 \not\sqsubseteq [\![C]\!]\sigma_2.$$

Note that with our semantics $[\![C]\!]\sigma_1 = \text{\ae} \sqsubseteq \mathsf{tt} = [\![C]\!]\sigma_2$.

Example 4.6.2. Compare following two tables.

$t < 0$	$\neg(t < 0)$

H^2

$\mathrm{y} < 10$
$\mathrm{y} \geq 10$

H^1

$\mathrm{x} + \mathrm{y}$	$\mathrm{x} - \mathrm{y}$
x^2	y^2

G

Table 21

$t \geq 0$	$\neg(t > 0)$

H^2

$y < 10$
$y \geq 10$

H^1

$x - y$	$x + y$
y^2	x^2

G

Table 22

Suppose $t \equiv (1 \text{ div } x)$ in the state $(x = 0)$. These conditions are equivalent, but the semantics in [Par93] gives different outputs for the two tables, since $(t \not< 0)$ in Table 21, being a complex expression, is evaluated to $\mathbb{tt}$ while $(t \geq 0)$ in Table 22 is evaluated to $\mathbb{ff}$. Our semantics gives $\mathbb{e}$ as the output for both tables.

We should however, point out that in the above example the table header H^2 used by Parnas would most likely be of the form

$t < 0$	$t \geq 0$

which would then yield (in this case) the same semantics as ours.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis we have developed a systematic method for handling error cases in computation over many-sorted algebras, with the use of *error algebras.* The desirable properties of these algebras, in computing with error cases, are:

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

(2) *error-consistency*, which is a weaker condition than consistency.

We have applied this theory to the semantics of (not necessarily proper) function tables.

5.2 Future work

Some possible applications or extensions of our work are the following.

(1) To generalise the theory in Chapter 4 to n-dimensional tables would be routine. More interesting, perhaps, would be generalising this theory to the other types of tables considered in [Par92].

(2) It would be interesting to find applications of error algebras, with our emphasis on monotonicity and error-consistency, in other areas of software analysis and verification, such as Hoare logic [TZ88, Zhu03], equational specifiability [Luo03] and program development [Jon06].

Bibliography

[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. Tuker 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.

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

[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.

3754 64