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 $\Sigma$: syntax and semantics . 16
3 Error Algebras

3.1 The error value $e$: algebras $A^e$ of signature $\Sigma^e$ .......................... 19

3.2 Semantics of term over $\Sigma^e$ ................................................................. 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) \textit{monotonicity}, which is a weaker condition than strictness, and

(2) \textit{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 boolean, has an error value $\varepsilon$ as well as $\mathsf{t}$ and $\mathsf{f}$, 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 $\Sigma$ and $\Sigma$-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
1. Introduction

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 $\Sigma$ and $\Sigma$-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 $\Sigma$). A many-sorted signature $\Sigma$ is a pair $\langle \text{Sort}(\Sigma), \text{Func}(\Sigma) \rangle$ where

1. $\text{Sort}(\Sigma)$ is a finite set of sorts;

2. $\text{Func}(\Sigma)$ is a finite set of primitive (or basic) function symbols

$$F : s_1 \times \cdots \times s_m \to s \quad (m \geq 0).$$
Each symbol $F$ has a type $s_1 \times \cdots \times s_m \to s$, where $s_1, \ldots, s_m \in \text{Sort}(\Sigma)$ are the domain sorts and $s \in \text{Sort}(\Sigma)$ 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 $\Sigma$).** A $\Sigma$-product type, or a product type over $\Sigma$, has the form $u = s_1 \times \cdots \times s_m$ ($m \geq 0$), where $s_1, \ldots, s_m \in \text{Sort}(\Sigma)$ are $\Sigma$-sorts. We write $u, v, w, \ldots$ for $\Sigma$-product types.

**Definition 2.1.3 ($\Sigma$-algebras).** A $\Sigma$-algebra $A$ has:

1. for each sort $s$ of $\Sigma$, a non-empty set $A_s$, called the carrier set of sort $s$;
2. for each $\Sigma$-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 $\Sigma$-product type $s_1 \times \cdots \times s_m$, and

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

The algebra $A$ is total if $F^A$ is total for each $\Sigma$-function symbol $F$. We write $\Sigma(A)$ for the signature of an algebra $A$ [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 $A$. 

Example 2.1.6. The algebra of *booleans* has signature

```
signature \( \Sigma(B) \)
sorts \( \text{bool} \)
functions \( \text{true, false : } \to \text{bool}, \)
\( \land, \lor : \text{bool}^2 \to \text{bool}, \)
\( \neg : \text{bool} \to \text{bool} \)
end
```

Then the algebra \( B \) has the carrier \( \mathbb{B} = \{\text{tt, ff}\} \) of sort \( \text{bool} \), and so

\[
B = (\mathbb{B}; \text{tt, ff, } \land, \lor, \neg)
\]

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

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

```
signature \( \Sigma(W) \)
sorts \( \text{letter, word} \)
functions \( \text{sing : letter } \to \text{word}, \)
\( \text{concat : word}^2 \to \text{word}, \)
\( ( ) : \to \text{word} \)
end
```
and so

\[ \mathcal{W}(A) = (A, A^*; \text{sing}^W, \text{concat}^W, \langle \rangle^W). \]

Example 2.1.8. Algebras of \textit{naturals}:

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

\[
\begin{array}{|c|}
\hline
\text{signature} & \Sigma(\mathcal{N}_0) \\
\text{sorts} & \text{nat} \\
\text{functions} & 0 : \rightarrow \text{nat}, \\
& \text{suc} : \text{nat} \rightarrow \text{nat} \\
\hline
\end{array}
\]

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

\[ \mathcal{N}_0 = (\mathbb{N}; 0^N, \text{suc}^N) \]

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

\[
\begin{array}{|c|}
\hline
\text{signature} & \Sigma(\mathcal{N}) \\
\text{import} & \mathcal{N}_0 \\
\text{functions} & +, \times : \text{nat}^2 \rightarrow \text{nat} \\
\hline
\end{array}
\]
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

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

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

$$\text{pred}^N(0) = 0$$

$$\text{div}^N(m, 0) = 0 \; \text{for all} \; m \in \mathbb{N}$$

$$\text{minus}^N(m, n) = 0 \; \text{for} \; m < n$$

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

<table>
<thead>
<tr>
<th>signature</th>
<th>$\Sigma(\mathcal{Z})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sorts</td>
<td>int</td>
</tr>
<tr>
<td>functions</td>
<td>$0, 1 : \to \text{int}$,</td>
</tr>
<tr>
<td></td>
<td>$+, \times : \text{int}^2 \to \text{int}$,</td>
</tr>
<tr>
<td></td>
<td>$\text{minus} : \text{int}^2 \to \text{int}$</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>end</td>
<td></td>
</tr>
</tbody>
</table>
2. Basic Concepts

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

\[
\mathcal{Z} = (\mathbb{Z}; 0, 1, +^\mathcal{Z}, \times^\mathcal{Z}, \text{minus}^\mathcal{Z}, \ldots).
\]

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

\[
\text{sqrt}^\mathcal{Z} : \text{int} \rightarrow \text{int}
\]

or \( \text{div}^\mathcal{Z} : \text{int}^2 \rightarrow \text{int} \) (integer division)

For now, we again use \textit{default values}:

\[
\text{sqrt}^\mathcal{Z}(m) = \begin{cases} 
0 & \text{for } m < 0 \\
 n & \text{where } n^2 \leq m < (n + 1)^2 \text{ for } m \geq 0
\end{cases}
\]

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

\textbf{Example 2.1.11.} The ring \( \mathcal{R}_0 \) of \textit{reals} has signature

\[
\text{signature } \Sigma(\mathcal{R}_0) = \begin{array}{c}
\text{sorts} \\
\text{real} \\\n\text{functions} \\
0, 1 : \rightarrow \text{real}, \\
+ , \times : \text{real}^2 \rightarrow \text{real}, \\
- : \text{real} \rightarrow \text{real}
\end{array}
\]

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

Definition 2.2.1. Let $\Sigma$ and $\Sigma'$ be signatures.

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

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

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

(b) The $\Sigma'$-algebra $A'$ is a $\Sigma'$-expansion of $A$ if and only if $A$ is the $\Sigma$-reduct of $A'$.

Example 2.2.2.

$$N_0 = N|_{\Sigma(N_0)}$$

$$R_0 = R|_{\Sigma(R_0)}$$

2.3 Standard signatures and algebras

Definition 2.3.1 (Standard signatures). A signature $\Sigma$ is standard if $\Sigma(B) \subseteq \Sigma$, and the function symbols of $\Sigma$ include a conditional

$$\text{if}_s : \text{bool} \times s^2 \rightarrow s$$
for all sorts $s$ of $\Sigma$ other than $\text{bool}$.

For a standard signature $\Sigma$, a sort of $\Sigma$ is called an \textit{equality sort} if $\Sigma$ includes an \textit{equality operator}

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

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

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

and (iii) the operator $\text{eq}_s$ is interpreted as a \textit{identity} on each equality sort $s$.

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

\begin{verbatim}
signature $\Sigma(\mathcal{Z}^B)$
import $\mathcal{Z}$, $B$,
functions $\text{eq}_{\text{int}}$, $\text{less}_{\text{int}}$: $\text{int}^2 \to \text{bool}$,
          $\text{if}_{\text{int}}$: $\text{bool} \times \text{int}^2 \to \text{int}$
end
\end{verbatim}
Then

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

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

More generally: Given a signature $\Sigma$ and a $\Sigma$-algebra $A$, a boolean expansion of $\Sigma$ is a signature $\Sigma^B$ where

$$\text{Sort}(\Sigma^B) = \text{Sort}(\Sigma) \cup \{\text{bool}\}$$

$$\text{Func}(\Sigma^B) = \text{Func}(\Sigma) \cup \text{Func}(\Sigma(\mathcal{B})) \cup \{(\text{eq}_s : s^2 \rightarrow \text{bool})_{s \in S}, (\text{if}_s : \text{bool} \times s^2 \rightarrow s)_{s \in \text{Sort}(\Sigma)}\}$$

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

The boolean expansion of $A$ is the $\Sigma^B$-algebra

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

where

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

and $$\text{eq}_s^A : A^2_s \rightarrow \mathbb{B} \quad (s \in S)$$
2. Basic Concepts

Example 2.3.4. The standard algebra of reals $\mathcal{R}^B_0$ 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 $\Sigma$ can be standardized to a standard signature $\Sigma^B$ 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 $B$ and other boolean operators, e.g. the equality operation at the equality sorts of $\Sigma^B$.

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

Remark 2.3.7. The standard algebra $\mathcal{Z}^B$ (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 $\mathcal{D}$:

$$\mathcal{D}^B = (\mathcal{D}; B, F^D_1, \ldots, F^D_k, \text{if}_{\text{data}}^D, \text{eq}_{\text{data}}^D)$$
of signature $\Sigma$ where

$$
\begin{align*}
Sort(\Sigma) &= \{\text{data, bool}\} \\
Func(\Sigma) &= Func(\Sigma)(D) \cup Func(\Sigma(B)) \cup \\
&\quad \{eq_{\text{data}} : \text{data}^2 \to \text{bool}, \\
&\quad \quad \text{if}_{\text{data}} : \text{bool} \times \text{data}^2 \to \text{data}
\}
\end{align*}
$$

Then, $\Sigma_{\text{stk}}$ is the stack signature over $\Sigma$ where

$$
\begin{align*}
Sort(\Sigma_{\text{stk}}) &= \{\text{data, bool, stk}\} \\
Func(\Sigma_{\text{stk}}) &= Func(\Sigma) \cup \\
&\quad \{\text{empty} : \text{stk}, \\
&\quad \quad \text{push} : \text{data} \times \text{stk} \to \text{stk} \\
&\quad \quad *\text{pop} : \text{stk} \to \text{stk} \\
&\quad \quad *\text{top} : \text{stk} \to \text{data} \\
&\quad \quad \text{isempty} : \text{stk} \to \text{bool} \\
&\quad \quad \text{eq}_{\text{stk}} : \text{stk}^2 \to \text{bool} \\
&\quad \quad \text{if}_{\text{stk}} : \text{bool} \times \text{stk}^2 \to \text{stk}
\}
\end{align*}
$$

$D_{\text{stk}}$ is the $\Sigma_{\text{stk}}$ expansion of $D$ where the carrier of sort stk is

$$
S = D_{\text{stk}} = \text{set of all stacks of data}
$$

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

Remarks 2.4.1.

(1) $D_{\text{stk}}$ is a standard algebra, and $\Sigma_{\text{stk}}$ includes $eq_{\text{stk}}$, derived from $eq_{\text{data}}$.

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

**default values:**

\[
\begin{align*}
\text{pop}(&\text{empty}) = \text{empty} \\
\text{top}(&\text{empty}) = ? \\
\end{align*}
\]

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

For example

\[
\begin{align*}
\mathbb{B} & \text{ take } \text{tt} \text{ (or ff)} \\
\mathbb{N} & \text{ take } 0 \\
\mathbb{Z} & \text{ take } 0 \\
A^* & \text{ take } \langle \rangle \\
\mathbb{S} & \text{ take empty}
\end{align*}
\]

More generally we make the following assumption on \(\Sigma\):

**Assumption 2.4.2 (Instantiation).** For each sort \(s\) of \(\Sigma\), there is a *closed term* in \(\Sigma\). Using these *closed terms*, \(\delta_s\), as *default values* we can systematically extend all functions in \(\Sigma\) to be *total* on all \(\Sigma\)-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 $\Sigma$: syntax and semantics

Definition 2.5.1 (Variables).

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

(2) 
\[
\text{Var}(\Sigma) = \bigcup_{s \in \text{Sort}(\Sigma)} \text{Var}_s
\]

Definition 2.5.2 (Terms).

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

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

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

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

(2) 
\[
\text{Tm}(\Sigma) = \bigcup_{s \in \text{Sort}(\Sigma)} \text{Tm}_s(\Sigma)
\]

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

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

$$\sigma = (\sigma_s)_{s \in \text{Sort}(\Sigma)}$$
2. Basic Concepts

of functions

$$\sigma_s : \text{Var}_s \rightarrow A_s.$$  

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

$$[t]^A : \text{State}(A) \rightarrow A_s$$

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

(a) \([x_s]^A \sigma = \sigma_s(x^s)\).

(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. \( \text{Var}(t) \) is the set of variables occurring in \( t \).

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

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

$$\sigma \approx \sigma'(\text{rel } M) \iff \sigma \upharpoonright M = \sigma' \upharpoonright M$$

i.e. \( \sigma \) and \( \sigma' \) agree on \( M \).

Lemma 2.5.8 (Coincidence Lemma). For any \( \Sigma \)-term \( t \):

$$\sigma \approx \sigma'(\text{rel } \text{Var}(t)) \implies [t]^A \sigma = [t]^A \sigma'$$
Proof. By structural induction on t. \hfill \Box

Definition 2.5.9 (Closed terms over \( \Sigma \)).

(1) \( t \) is closed if \( \text{Var}(t) = \emptyset \)

(2) \( CT(\Sigma) \) is the set of all closed \( \Sigma \)-terms.

Corollary 2.5.10. If \( t \) is closed then \( [t]_\sigma \) is independent of \( \sigma \).

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 $\varepsilon$: algebras $A^\varepsilon$ of signature $\Sigma^\varepsilon$

Given a standard $\Sigma$-algebra

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

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

$$A^\varepsilon_s = A_s \cup \{\varepsilon\}$$

In particular, $B^\varepsilon = \{\texttt{t}, \texttt{f}, \varepsilon\}$, producing a three-valued logic.
For each $F : u \rightarrow s$ in $\text{Func}(\Sigma)$, if $u = s_1 \times \cdots \times s_m$, define

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

let

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

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

$$F^{A^{e}} = F^{A^{e}} : A^{u,e} \rightarrow A^{e}_{s}$$

be some extension of $F^{A} : A^{u} \rightarrow A_{s}$.

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

1. $F^{A^{e}}$ is strict over $A$ if $F^{A^{e}}(a_1, \ldots, \alpha, \ldots, a_m) = \alpha$
   
   (i.e. $F^{A^{e}}(a_1, \ldots, a_m)$ has value $\alpha$ if any argument is $\alpha$); and

2. $F^{A^{e}}$ is consistent over $A$ if it extends $F^{A}$, i.e. $F^{A^{e}} \upharpoonright A = F^{A}$.

**Definition 3.1.2 (Basic error signature and algebra).** Let $A^{e}$ be the algebra

$$(A^{e}_{s_1}, \ldots, A^{e}_{s_{k-1}}, B^{e}, F^{A^{e}}_{1}, \ldots, F^{A^{e}}_{n}, (\alpha_{s})_{s \in \text{Sort}(\Sigma)})$$

of signature $\Sigma^{e}$ where

$$\text{Sort}(\Sigma^{e}) = \text{Sort}(\Sigma)$$

$$\text{Func}(\Sigma^{e}) = \text{Func}(\Sigma) \cup \{(\text{errors})_{s \in \text{Sort}(\Sigma)}\}$$
and for all sorts $s : \text{error}_s^A = \varepsilon$

We call

(1) $\Sigma^\varepsilon$ the \textit{basic error signature} over $\Sigma$;

(2) $A^\varepsilon$ the \textit{basic error algebra} over $A$.

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

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

$$B^\varepsilon = (\mathbb{B}^\varepsilon; \text{tt, ff, } \varepsilon, \text{and}^\varepsilon, \text{or}^\varepsilon, \text{not}^\varepsilon)$$

The logical operators and$^\varepsilon$, or$^\varepsilon$ and not$^\varepsilon$ which extend and, or and not strictly and consistently, give rise to \textit{weak 3-valued logic} [Kle52, §64].

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

<table>
<thead>
<tr>
<th>and</th>
<th>tt</th>
<th>ff</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>tt</td>
<td>tt</td>
<td>ff</td>
<td>$\varepsilon$</td>
</tr>
<tr>
<td>ff</td>
<td>ff</td>
<td>ff</td>
<td>$\varepsilon$</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>$\varepsilon$</td>
<td>$\varepsilon$</td>
<td>$\varepsilon$</td>
</tr>
</tbody>
</table>

\textit{Table 1: Strict 'and'}
Table 2: Strict ‘or’

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

Table 3: Strong ‘and’

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:

<table>
<thead>
<tr>
<th>strong-imp</th>
<th>tt</th>
<th>ff</th>
<th>ε</th>
</tr>
</thead>
<tbody>
<tr>
<td>tt</td>
<td>tt</td>
<td>ff</td>
<td>ε</td>
</tr>
<tr>
<td>ff</td>
<td>tt</td>
<td>tt</td>
<td>tt</td>
</tr>
<tr>
<td>ε</td>
<td>tt</td>
<td>ε</td>
<td>ε</td>
</tr>
</tbody>
</table>

Table 5: Strong ‘imply’

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

(1) $x \neq 0$ and $(1 \div x) > 0$

(2) $x = 0$ or $(1 \div x) > 0$

Suppose $x = 0$ (*i.e.* evaluate at $\sigma$ with $\sigma(x) = 0$). 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 $\varepsilon$.

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

$$B = (\mathbb{B}; \top, \bot, \text{and}, \text{or}, \text{not})$$
which is then extended *consistently* but *not strictly* to:

\[ B^\varepsilon = (\mathbb{B}^\varepsilon; \text{tt}, \text{ff}, \text{and}^\varepsilon, \text{or}^\varepsilon, \text{not}^\varepsilon, \text{cand}, \text{cor}) \]

**Remarks 3.1.8.** Consider the data algebra

\[ A = D^B = (\mathbb{D}, \mathbb{B}; \ldots, \text{tt}, \text{ff}, \wedge, \vee, \neg, \text{eq}^D, \text{if}^D) \]

and the basic algebra over \( A \)

\[ A^\varepsilon = D^{B^\varepsilon} = (\mathbb{D}^\varepsilon, \mathbb{B}^\varepsilon; \ldots, \text{tt}, \text{ff}, \wedge^\varepsilon, \vee^\varepsilon, \neg^\varepsilon, \text{eq}^{D^\varepsilon}, \text{if}^{D^\varepsilon}, \text{ae}^D, \text{ae}^B) \]

Now, consider the interpretation of equality eq and the conditional if in \( A^\varepsilon \):

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

\[
\text{eq}^{A^\varepsilon}(x, \varepsilon) = \begin{cases} 
\varepsilon \text{ (not tt)} & \text{if } x = \varepsilon \\
\varepsilon \text{ (not ff)} & \text{otherwise}
\end{cases}
\]

On the other hand, the identity function ("strong equality") on \( A^\varepsilon \) has the form:

\[
\text{id}^{A^\varepsilon} : (\mathbb{D}^\varepsilon)^2 \rightarrow \mathbb{B}
\]

where

\[
\text{id}^{A^\varepsilon}(x, \varepsilon) = \begin{cases} 
\text{tt} & \text{if } x = \varepsilon \\
\text{ff} & \text{otherwise}
\end{cases}
\]

Note that \( \text{id}^{A^\varepsilon} \) is a *non-strict* extension of \( \text{eq}^A \).
(2) **Conditional:** Note that if$A^\epsilon$ extends if$A$ by strictness. But it is not a conditional operation on $D^\epsilon$:

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

This is a "weak conditional" on $A^\epsilon$.

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

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

where

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

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

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

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

(a) $A^\epsilon$ contains $B^\epsilon$ instead of $B$; and

(b) $A^\epsilon$ has if$A^\epsilon$ and 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 $A^\epsilon$. 
Example 3.1.9. Consider

\[ \mathcal{R} = (\mathbb{R}, \mathbb{Q}, \mathbb{Z}, \mathbb{B}; \ldots, \text{eq}_\mathbb{Q}, \text{eq}_\mathbb{Z}) \]

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:

\[ \mathcal{R}^e = (\mathbb{R}^e, \mathbb{Q}^e, \mathbb{Z}^e, \mathbb{B}^e; \alpha^R, \text{eq}_\mathbb{Q}, \text{eq}_\mathbb{Z}, \text{eq}_\mathbb{Q}^e, \text{eq}_\mathbb{Z}^e) \]

This does not have \(\text{eq}^e\) on \(\mathcal{R}^e\). However we assume we can still distinguish between "real" reals and \(\alpha^R\), so we add the predicate

\[ \text{is-error}^R : \mathbb{R}^e \rightarrow \mathbb{B}^e \]

where

\[ \text{is-error}^R(x) = \begin{cases} 
\text{tt} & \text{if } x = \alpha^R \\
\text{ff} & \text{otherwise} 
\end{cases} \]

(as well as \(\text{error}_s\) on the other sorts \(s\))

Remark 3.1.10 (Other operators in \(A^e\)). Over the basic error algebra \(A^e\) we can define operators such as

- \(\text{cand}\) or \(\text{cor}\) from \(\text{if}_{ns}\);
• id from eq, is-error and if_{ns};

• is-error from id.

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

Definition 3.1.12 (Strict and consistent error signature and algebra). Let $A$ be a $\Sigma$-algebra and $A^e$ an error algebra over $A$.

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

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

Remarks 3.1.13.

1. The **basic error algebra** over $A$ is consistent and strict over $A$.

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

3.2 Semantics of term over $\Sigma^e$

Note that $State(A) \subset State(A^e)$. 
Definition 3.2.1 (States over $A^e$ represented over $A$).

For $\sigma \in \text{State}(A^e)$, define $\sigma^A \in \text{State}(A)$ by

$$\sigma^A(x^e) = \begin{cases} 
\sigma(x^e) & \text{if } \sigma(x) \neq \varepsilon \\
\delta^A & \text{if } \sigma(x) = \varepsilon 
\end{cases}$$

where $\delta$ 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 $A^e$ be an error algebra over $A$, $t \in Tm(\Sigma)$ and $\sigma \in \text{State}(A^e)$. Then

1. If $A^e$ is strict over $A$, and there exists $x \in \text{Var}(t)$ such that $\sigma(x) = \varepsilon$, then

$$[t]^A^e \sigma = \varepsilon$$

2. If $A^e$ is consistent over $A$, and for all $x \in \text{Var}(t)$, $\sigma(x) \neq \varepsilon$, then

$$[t]^A^e \sigma = [t]^A \sigma^A \neq \varepsilon$$

Proof.

1. By structural induction on $t$.

   (a) $t \equiv x$:

   By assumption, $\sigma(x) = \varepsilon$,

   and so,

$$[x]^A^e \sigma = \sigma(x) = \varepsilon.$$
(b) $t \equiv F(t_1, \ldots, t_m)$:

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

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

$x \in \text{Var}(t_i)$ and $\sigma(x) = \varepsilon$

By induction hypothesis:

$\llbracket t \rrbracket^{A^e} \sigma = \varepsilon$

Hence,

$\llbracket F^{A^e}(t_1, \ldots, t_m) \rrbracket^{A^e} \sigma = F^{A^e}(\llbracket (t_1) \rrbracket^{A^e} \sigma, \ldots, \llbracket t_m \rrbracket^{A^e} \sigma) = \varepsilon$

since $F^{A^e}$ 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,

$\llbracket c \rrbracket^{A^e} \sigma = \varepsilon$.

(2) By structural induction on $t$:

(a) $t \equiv x$:

Using the given fact that $\sigma(x) \neq \varepsilon$:

$\llbracket x \rrbracket^{A^e} \sigma = \sigma(x) = [x]^{A} \sigma^{A}$
By definition

\[ [F(t_1, \ldots, t_n)]^{A^e} \sigma = F^{A^e}(\llbracket t_1 \rrbracket^{A^e} \sigma, \ldots, \llbracket t_n \rrbracket^{A^e} \sigma) \]

where by the induction hypothesis:

\[ \forall i \llbracket t_i \rrbracket^{A^e} \sigma = \llbracket t_i \rrbracket^{A^e} \sigma^A \quad 0 \leq i \leq n \quad (\ast) \]

continuing on we have:

\[
F^{A^e}(\llbracket t_1 \rrbracket^{A^e} \sigma, \ldots, \llbracket t_n \rrbracket^{A^e} \sigma) = F^{A^e}(\llbracket t_1 \rrbracket^{A^e} \sigma^A, \ldots, \llbracket t_n \rrbracket^{A^e} \sigma^A) \quad (by \ (\ast))
\]

\[
= F^A(\llbracket t_1 \rrbracket^{A^e} \sigma^A, \ldots, \llbracket t_n \rrbracket^{A^e} \sigma^A) \quad (A^e \ is \ consistent)
\]

(c) \( t \equiv c: \)

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

\[ \square \]

Remarks 3.2.2.

(1) Theorem 1(1) says that the semantics of \( A^e \) extends the semantics of \( A \) strictly.

i.e. errors propagate or persist: once \( \varepsilon \) 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 \( A^e \) extends the semantics of \( A \) consistently.
Discussion 3.2.3. So there are two possible reasons why $[t]^{A^e} \sigma = \varepsilon$:

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

2. From Theorem 1(2): $A^e$ is not consistent over $A$, so some function at some argument returns $\varepsilon$ instead of a default value.

Corollary 3.2.4. If $A^e$ is consistent over $A$, $t \in Tm(\Sigma)$ and $\sigma \in \text{State}(A)$ then

$$[t]^{A^e} \sigma = [t]^A \sigma$$

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

Corollary 3.2.5. If $A^e$ is consistent over $A$, and $t \in CT(\Sigma)$ then

$$[t]^{A^e} = [t]^A.$$

Proof. Immediate from Theorem 1(2). □

Discussion 3.2.6. Theorem 1 works only assuming $A^e$ 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 ifns as (non-strict) extensions of and, or and ifA. We will return to this.

- **Consistency**: Consider algebras containing functions with default values, for example:
3. Error Algebras

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

Now define:

(2) \( N^\varepsilon = (\mathbb{N}^\varepsilon, \mathbb{B}^\varepsilon; 0, \text{suc}^\varepsilon, \text{pred}^\varepsilon, \ldots) \) but with \( \text{pred}^\varepsilon(0) = \varepsilon \).

So now \( \text{pred}^\varepsilon \) is not a consistent extension of \( \text{pred} \).

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

\[
\text{div in } N \text{ or } \mathbb{Z}; \text{ or }
\]

\[
\text{pop and top in } D_{\text{stk}}.
\]

So consistency is not always desirable.

Remarks 3.2.7.

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

(2) We can think of \( \sigma(x) = \varepsilon \) as "\( \sigma \) is unspecified at \( x \)" or "\( x \) is not yet initialized in state \( \sigma \)".

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

(1) strictness; and
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 $A^c_s$ of $A^c$.

**Definition 3.3.1.** For all $x, y \in A^c_s$

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

**Definition 3.3.2.**

(1) Let $F : s_1 \times \cdots \times s_m \rightarrow s$ be in $\textbf{Func}(\Sigma)$. Then $F^{A^c}$ is monotonic over $A$ iff

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

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

(2) $A^c$ is monotonic over $A$ iff for all $F \in \textbf{Func}(\Sigma)$, $F^{A^c}$ is monotonic over $A$.

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

**Proof.** From definitions of strictness (Definition 3.1.1) and monotonicity (Definition 3.3.2). \qed
An equivalent characterization of monotonicity is:

**Lemma 3.3.4.** If $F : u \rightarrow s$ is in $Func(\Sigma)$, then $F^{A,\epsilon}$ is monotonic over $A$ 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 $\epsilon$),

if $F^{A,\epsilon}(\bar{a}) = y \neq \epsilon$, then replacing $\epsilon$ 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). \qed

**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

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

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 \( M \subseteq \text{Var}(\Sigma) \),

\[ \sigma \sqsubseteq \sigma'(\text{rel } M) \quad ("\sigma \text{ is extended by } \sigma' \text{ relative to } M") \]

iff for all \( x \in M \), \( \sigma(x) \sqsubseteq \sigma'(x) \)

**Proposition 3.3.9.** \( \sigma \approx \sigma'(\text{rel } M) \iff \sigma \sqsubseteq \text{rel } M \) and \( \sigma' \sqsubseteq \sigma(\text{rel } M) \)

**Proof.** Clear from Definitions 3.3.1 and 3.3.8. \( \square \)

**Remarks 3.3.10.**

(1) The relation \( \sqsubseteq \text{rel } M \) (for fixed \( M \)) is a pre-partial order on \( \text{State}(A^e) \), i.e. it is transitive and reflexive (but not anti-symmetric).

(2) The relation \( \approx \text{rel } M \) is the corresponding equivalence relation at \( \text{State}(A^e) \).

(3) The \( \sqsubseteq \text{rel } M \) -minimal states (rel \( M \)) are those which are totally unspecified on \( M \).

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

\[ \sigma \sqsubseteq \sigma'(\text{rel Var}(t)) \implies [t]\sigma \sqsubseteq [t]\sigma' \]
3. Error Algebras

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

Remark 3.3.11. As a Corollary we get: if $A^e$ is monotonic over $A$ then

$$\sigma \approx \sigma'(rel\ Var) \implies [t]\sigma = [t]\sigma'$$

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

Corollary 3.3.12. If $A^e$ is monotonic over $A$, then for all $t \in Tm(\Sigma)$ and $\sigma \in State(A^e)$:

$$[t]\sigma \neq \varepsilon \implies \forall \sigma' \approx \sigma (rel\ Var(t)), [t]\sigma' = [t]\sigma$$

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 $A$ be $\Sigma$-algebra, $A^e$ an error algebra over $A$ and

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

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

$$F^{A,e}(a_1, \ldots, a_m) \subseteq F^A(a_1, \ldots, a_m)$$
(2) $\mathcal{A}^\varepsilon$ is error-consistent over $\mathcal{A}$ if and only if for all $F \in \text{Func}(\Sigma)$, $F^{\mathcal{A}^\varepsilon}$ is error-consistent over $\mathcal{A}$.

An equivalent characterization of error-consistency is:

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

$$F^{\mathcal{A}^\varepsilon}(a_1, \ldots, a_m) = F^\mathcal{A}(a_1, \ldots, a_m) \text{ or } \varepsilon.$$

**Notes.**

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

(2) Every $F^{\mathcal{A}^\varepsilon}$ considered so far is error-consistent.

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

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

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

(1) If $\sigma \in \text{State}(\mathcal{A}^\varepsilon)$ and for all $x \in \text{Var}(t)$, $\sigma(x) \neq \varepsilon$ then

$$[t]^{\mathcal{A}^\varepsilon} \sigma \subseteq [t]^{\mathcal{A}} \sigma^\mathcal{A},$$
3. Error Algebras

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

\[
[t]^A \sigma \subseteq [t]^A \sigma;
\]

(3) If $t$ is closed then

\[
[t]^A \subseteq [t]^A.
\]

Proof.

(1) By structural induction on $t$:

(a) $t \equiv x$:

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

\[
[x]^A \sigma = \sigma(x) = [x]^A \sigma^A
\]

and so $[t]^A \sigma \subseteq [t]^A \sigma^A$

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

By definition

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

where by the induction hypothesis:

\[
\forall i \ [t_i]^A \sigma \subseteq [t_i]^A \sigma^A \quad 0 \leq i \leq n \quad (\star)
\]

continuing on we have:
\[
F^{A^e}(\llbracket t_1^{A^e} \sigma, \ldots, t_n^{A^e} \sigma \rrbracket) \subseteq F^{A^e}(\llbracket t_1^{A^e} \sigma^A, \ldots, t_n^{A^e} \sigma^A \rrbracket) \quad \text{(monotonicity, \((*)\))}
\]

\[
\subseteq F^A(\llbracket t_1^{A^e} \sigma^A, \ldots, t_n^{A^e} \sigma^A \rrbracket) \quad \text{(error-consistency)}
\]

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

(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 $\sigma$, whether $T$ is proper or improper at $\sigma$. (The answer will vary, in general, with $\sigma$). 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 $\mathcal{Tab}_N(\Sigma)$ of normal (function) tables over $\Sigma$. Consider (for convenience) a 2-dimensional normal table [Par92, Zuc96].

**Example 4.1.1 (A two dimensional normal table).**

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 $\Sigma$, 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 $\Sigma$-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 $\mathsf{tt}$ (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 $\varepsilon$ for a given state $\sigma$:

(i) $T$ is not proper at $\sigma$;

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

(iii) each header is proper (at $\sigma$) with a true condition (say) $C_i^1$ and $C_j^2$, but

$[t_{i_0j_0}]\sigma = \varepsilon$.

Remarks 4.1.2. Above, by “proper” we mean:

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

(2) there is a unique $j$ such that $[C_j^2]\sigma = \mathsf{tt}$, and for all $j' \neq j$, $[C_{j'}] = \mathsf{ff}$.
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 \( \sigma \), 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 \( \varepsilon \). So define the table function as:

\[
f^A_T : A \rightarrow A \cup \{\varepsilon\}.
\]

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 $\sigma$ over $\text{Var}(C,C_1,\ldots,C_n)$ if $\sigma$ satisfies $C$, then $\sigma$ satisfies at most one of $C_1,\ldots,C_n$.

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

- strictly proper relative to $C$ at state $\sigma$ if it is both disjoint and universal relative to $C$ at $\sigma$.

Equivalently, $(C_1,\ldots,C_n)$ is strictly proper relative to $C$ at $\sigma$ over $\text{Var}(C,C_1,\ldots,C_n)$ if $\sigma$ satisfies $C$, then $\sigma$ 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 $\sigma$ if it is (respectively) disjoint, universal or strictly proper relative to the condition true at $\sigma$.

Equivalently, $(C_1,\ldots,C_n)$ is strictly proper at $\sigma$ over $\text{Var}(C_1,\ldots,C_n)$ iff $\sigma$ satisfies exactly one of $C_1,\ldots,C_n$. 
4. Semantic of Improper Tables using Error Algebras

Note that these concepts (disjointness, universality and strict properness) are all relative to the $\Sigma$-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 $\sigma$ if

(i) all its headers are universal at $\sigma$, and

(ii) the value of a term $t_{ij}$ at $\sigma$ is the same for all $(i, j)$ for which conditions $C_i^1$ in header $H_1^1$ and $C_j^2$ in header $H_2^2$ are true at $\sigma$.

**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).

\[ \begin{array}{c|c|c}
\text{y} & 10 & y > 10 \\
\hline
x \geq 0 & x + y & 10 + x \\
\hline
x < 0 & x - y & x - 10 \end{array} \]

\( H^2 \)

\[ \begin{array}{c|c|c|c}
\text{H}^1 & \text{G} & \\
\hline
x \geq 0 & 0 & y^2 & -y^2 \\
\hline
x < 0 & x & x + y & x - y \end{array} \]

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 \( \Sigma(Z^R) \) of the standardised algebra of integer (or some expansion of it).

Example 4.1.10 (A strictly proper normal table).

\[ \begin{array}{c|c|c}
y = 10 & y > 10 & y < 10 \\
\hline
\text{H}^2 & \\
\hline
\end{array} \]

\[ \begin{array}{c|c|c|c}
x \geq 0 & 0 & y^2 & -y^2 \\
\hline
x < 0 & x & x + y & x - y \end{array} \]

\( \text{H}^1 \)

\( \text{G} \)

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 $\Sigma$, and $\sigma$ a state over $T$ in $A$. Suppose $T$ is proper at $\sigma$. Choose indices $i, j$ for which the entries $C^1_i$ and $C^2_j$ hold at $\sigma$. Then the meaning of $T$ relative to $\sigma$ 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 $[C_i]\sigma = [C_j]\sigma = t$.

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

Definition 4.1.12. A list $\bar{x}$ of variables is said to cover $T$ if it includes all of $\text{Var}(T)$, i.e., if $\text{Var}(T) \subseteq \bar{x}$.

Definition 4.1.13. Let $\bar{x} \equiv (x_1, \ldots, x_m)$ be any list of variables which covers $T$, with $x_i : s_i$ for $i = 1, \ldots, m$. Then relative to $\bar{x}$, $T$ names or defines a table function symbol

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

with interpretation on $A$

$$f_{T,\bar{x}}^A : s_1 \times \cdots \times s_m \to A_s$$

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

$$f_{T,\bar{x}}^A(a_1, \ldots, a_m) = [T]^A\sigma.$$
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 $\sigma$. 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 $\sigma \in State(A^e)$ if:

1. for some $i$, $[C_i]^{A^e} \sigma = \texttt{tt}$;
2. for all $j$, $[C_j]^{A^e} \sigma \neq \texttt{ff}$.

**Remarks 4.1.16.**

1. the definition (4.1.7) of properness of a normal table $T$ at a state $\sigma$ over $A^e$ now presupposes that none of the header conditions evaluates to $\texttt{ff}$.

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

$$\text{if}_{ns}(t^{\text{bool}}, t^a_1, t^a_2)$$
We now extend the definition (4.1.11) of $[T]^A\sigma$ in the case that $T$ is improper at $\sigma$:

**Definition 4.1.17 (Semantics of table over $\Sigma^e$).** Let $T$ be a normal table over $\Sigma^e$, and $\sigma$ a state over $T$ in $A^e$. We define $[T]^A\sigma$ as follows:

*Case 1:* $T$ is proper at $\sigma$. Then $[T]^A\sigma$ is as in Definition 4.1.11.

*Case 2:* $T$ is improper at $\sigma$. Then $[T]^A\sigma = \varepsilon$.

**Definition 4.1.18 (Table function).** Let $A^e$ be an error algebra over $A$. Let $T$ be a table, $\vec{x} = (x_1, \ldots, x_m)$ be any list of variables which covers $T$ with $x_i : s_i$ for $i = 1, \ldots, m$. For $a_1 \in A_{s_1}, \ldots, a_m \in A_{s_m}$ let $\sigma$ be a state over $T$ satisfying $\sigma(x_i) = a_i$ for $i = 1, \ldots, m$. Then:

1. if $T$ is proper at $\sigma$ (according to Definition 4.1.7 applied to $A^e$), define $f_{T,\vec{x}}^{A,e}(a_1, \ldots, a_m)$ as above (using $A^e$ instead of $A$).

2. if $T$ is not proper at $\sigma$, then

$$f_{T,\vec{x}}^{A,e}(a_1, \ldots, a_m) = \varepsilon.$$

**Remarks 4.1.19 (Semantics with error algebras).**

1. If $T$ is improper at $\sigma$, then

$$f_{T,\vec{x}}^{A,e}(a_1, \ldots, a_m) = [T]^A\sigma = \varepsilon.$$
(2) But even if \( T \) is proper at \( \sigma \), this semantics may give an output of \( \varepsilon \), because \( A^\varepsilon \)

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

\[
\begin{array}{|c|c|}
\hline
y \leq 0 & y + 1 \\
\hline
y > 0 & y + 2 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
x = 0 & x \neq 0 \\
\hline
\end{array}
\]

\( x \neq 0 \)

\( y \div x \)

\( y \times x \)

\( y + 2 \)

\( G \)

\( H^1 \)

\( H^2 \)

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) \( \varepsilon \) 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 \( A^\varepsilon \) be an error algebra which is monotonic and error-consistent. Let \( T \) be a normal table, with \( \operatorname{Var}(T) \subseteq x \). Then \( f_{T,x}^{A^\varepsilon} \) is

(1) monotonic, and
(2) *error-consistent*.

Proof.

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

\[
\sigma_1 \subseteq \sigma_2 \implies [T]_{\sigma_1} \subseteq [T]_{\sigma_2} \tag{*}
\]

If \([T] = \emptyset\), this is trivial. So we may assume: \([T] \neq \emptyset\).

Hence \(T\) is proper at \(\sigma\), i.e.,

(a) the headers of \(T\) are both universal at \(\sigma_1\),

(b) for all \(i, j\) such that:

\[
[C_j^2]_{\sigma_1} = \texttt{tt}, \quad [t_{ij}]_{\sigma_1} = a \text{ (say)} \neq \emptyset,
\]

and so,

\[
[T]_{\sigma_1} = a.
\]

By *monotonicity* of \(A^e\), 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 \(\sigma_2\) and for all \(i, j\) such that:

\[
[C_i^1]_{\sigma_2} = [C_j^2]_{\sigma_2} = \texttt{tt},
\]

\[
[t_{i,j}] = a,
\]
and so

\[ [T]_{\sigma_2} a = [T]_{\sigma_1}. \]

proving (*).

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

\[ [T]^{A,\epsilon}\sigma \subseteq [T]^{A}\sigma \]  

\((**)\)

There are two cases.

(a) \( T \) is proper at \( \sigma \).

Then the header \( H^1 \) and \( H^2 \) are universal, w.r.t. \( \sigma \). By Theorem 3

for \( i = 1, \ldots, k \) and \( j = 1, \ldots, l \):

\[ [C_i^1]^{A,\epsilon}\sigma \subseteq [C_i^1]^{A}\sigma \quad \text{and} \quad [C_j^2]^{A,\epsilon}\sigma \subseteq [C_j^2]^{A}\sigma. \]  

\((***)\)

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

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

\((****)\)

Since \( T \) is proper at \( \sigma \), there exists \( i, j \):

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

Then for this \( i, j \)

\[ [T]^{A}\sigma = [t_{ij}]^{A}\sigma. \]
4. Semantic of Improper Tables using Error Algebras

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

$$[T]^{A_{i,j}} = [t_{ij}]^{A_{i,j}}.$$  

By Theorem 3:

$$[t_{ij}]^{A_{i,j}} = [t_{ij}]^{A}.$$  

Hence,

$$[T]^{A_{i,j}} = [T]^{A}.$$  

(b) $T$ is improper at $\sigma$. Then

$$[T]^{A_{i,j}} = \varepsilon.$$  

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

\[\Box\]

4.2 Inverted tables

In this section we consider the class $Tab_f(\Sigma)$ of inverted (function) tables over $\Sigma$.

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 $\sigma$ over $T$, search the condition header $H^2$ until you find a condition $C_j$ which holds at $\sigma$. The index $j$ determines a column. Search along this column for the cell $(i, j)$ whose entry $C_{ij}$ has the value $\mathtt{t}$. 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).**

![Inverted Table Example]

**Table 12**

**Definition 4.2.2 (Proper inverted tables).** $T$ is proper at $\sigma$ iff

1. For some $j$, $\sigma \vdash C_j$. 

(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).

![Table 13]

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 \( f_{T,x}^{A,\varepsilon} \) 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 $A^e$).** Let $T_1$ and $T_2$ be two tables. $T_1$ and $T_2$ are **semantically equivalent on $A^e$** (written $T_1 \approx_{A^e} T_2$) iff for all states $\sigma$ over $\text{Var}(T_1, T_2)$ in $A^e$, $[T_1]^{A^e} \sigma = [T_2]^{A^e} \sigma$.

**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 $\tau$ to another class $\tau'$. These transformations must satisfy the following three properties:

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

2. $\varphi$ is **effective or computable**.

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

If $\varphi(T) = T'$, then $T'$ is called the **transform** of $T$ under $\varphi$. 
4. Semantic of Improper Tables using Error Algebras

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).

$T$ 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).

Table 15
4. Semantic of Improper Tables using Error Algebras

Example 4.4.3 (A special case of Table 14).

Table 16

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

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 $\bar{T}$ the inverted table obtained from $T$ by Method 1 or 2. Then

$\bar{T}$ is proper at $\sigma$ $\iff$ $T$ is proper at $\sigma$. 
Proof. We show

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

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

(1) $T$ is proper at $\sigma$.

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

(2) $T$ is improper at $\sigma$.

If $H^2$ is not universal in $T$ (at some state $\sigma$), 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 $\sigma$) but lead to different values on the overlap, then these different values will also manifest themselves in the value header of $\tilde{T}$.

\[\square\]

Remarks 4.4.6. Suppose the normal table $T$ (Table 15) is proper but not strictly proper, e.g. if $\sigma \models C^2_1$ 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

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

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

$$f_{T,x}^{A_\epsilon}(a_1, \ldots, a_m) = [T]^{A^*} \sigma = \epsilon$$

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

$$f_{\tilde{T},x}^{A_\epsilon}(a_1, \ldots, a_m) = [\tilde{T}]^{A^*} \sigma = \epsilon.$$

Thus,

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

and so

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

\qed
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 $3 \times 2$ inverted table shown as Table 18, with value header $H^1$.

Example 4.5.1 (Two-dimensional table).

![Table 18]

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

Example 4.5.2 (Normalisation of Table 18).

![Table 19]
4. Semantic of Improper Tables using Error Algebras

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

Example 4.5.3 (Another normalisation of Table 18).

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
$C_1^2 \& C_{11}$ & $t_1$ \\
\hline
$C_2^2 \& C_{12}$ & $t_1$ \\
\hline
$C_1^2 \& C_{21}$ & $t_2$ \\
\hline
$C_2^2 \& C_{22}$ & $t_2$ \\
\hline
$C_1^2 \& C_{31}$ & $t_3$ \\
\hline
$C_2^2 \& C_{32}$ & $t_3$ \\
\hline
\end{tabular}
\caption{Table 20}
\end{table}

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

$\hat{T}$ is proper at $\sigma$ $\iff$ $T$ 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:

$$\hat{T} \approx_{A'} T.$$ 

**Proof.** Similar to Theorem 5. \qed

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

2. (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 $f \in \textbf{Func}(\Sigma)$ of type $s_1 \times \cdots \times s_n \to \text{bool}$ is evaluated as:

$$[C]_{\sigma} = \begin{cases} 
\text{tt} & \text{if } [t_1]_{\sigma} \neq \varepsilon, \ldots, [t_n]_{\sigma} \neq \varepsilon \text{ and } C_A([t_1]_{\sigma}, \ldots, [t_n]_{\sigma}) = \text{tt} \\
\text{ff} & \text{if } [t_1]_{\sigma} \neq \varepsilon, \ldots, [t_n]_{\sigma} \neq \varepsilon \text{ and } C_A([t_1]_{\sigma}, \ldots, [t_n]_{\sigma}) = \text{ff} \\
\text{ff} & \text{if } [t_1]_{\sigma} = \varepsilon \text{ or } \ldots \text{ or } [t_n]_{\sigma} = \varepsilon 
\end{cases}$$
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 $C \equiv (x = 0)$, then for

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

we get

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

and so

$$
\sigma_1 \subseteq \sigma_2 \quad \text{but} \quad [C]_\sigma_1 \not\subseteq [C]_\sigma_2.
$$

Note that with our semantics $[C]_\sigma_1 = \varepsilon \subseteq \mathsf{tt} = [C]_\sigma_2$.

**Example 4.6.2.** Compare following two tables.

<table>
<thead>
<tr>
<th></th>
<th>$t &lt; 0$</th>
<th>$\neg(t &lt; 0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$H^2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$y &lt; 10$</th>
<th>$x + y$</th>
<th>$x - y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y \geq 10$</td>
<td>$x^2$</td>
<td>$y^2$</td>
</tr>
</tbody>
</table>

$H^1$ $G$

Table 21
Suppose \( t \equiv (1 \div x) \) in the state \( (x = 0) \). These conditions are equivalent, but the semantics in [Par93] gives different outputs for the two tables, since \( (t \neq 0) \) in Table 21, being a complex expression, is evaluated to \( \text{tt} \) while \( (t \geq 0) \) in Table 22 is evaluated to \( \text{ff} \). Our semantics gives \( \epsilon \) 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

\[
\begin{array}{|c|c|} 
\hline
\text{t < 0} & \text{t \geq 0} \\
\hline
\end{array}
\]

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


