J. Mende  
Research Directions in Information Systems  
1

R. Dempster  
The SECD Machine: an Introduction  
5

J.M. Bishop  
Ways of Assessing Programming Skills  
11

N.C. Phillips  
An Alternative Development of Vienna Data Structures  
21

Book Reviews  
20

An official publication of the Computer Society of South Africa and of the South African Institute of Computer Scientists

'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Afrika en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes
QUAESTIONES INFORMATICAE

An official publication of the Computer Society of South Africa and of the South African Institute of Computer Scientists

'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Afrika en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

Editor

Professor G. Wiechers
INFOPLAN
Private Bag 3002
Monument Park 0105

Editorial Advisory Board

Professor D.W. Barron
Department of Mathematics
The University
Southampton S09 5NH, England

Professor J.M. Bishop
Department of Computer Science
University of the Witwatersrand
1 Jan Smuts Avenue
Johannesburg 2001

Professor K. MacGregor
Department of Computer Science
University of Cape Town
Private Bag
Rondebosch 7700

Dr H. Messerschmidt
IBM South Africa
P. O. Box 1419
Johannesburg 2000

Subscriptions

Annual subscriptions are as follows:

<table>
<thead>
<tr>
<th></th>
<th>SA</th>
<th>US</th>
<th>UK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individuals</td>
<td>R10</td>
<td>$7</td>
<td>£5</td>
</tr>
<tr>
<td>Institutions</td>
<td>R15</td>
<td>$14</td>
<td>£10</td>
</tr>
</tbody>
</table>

Quaestiones Informaticae is prepared by the Computer Science Department of the University of the Witwatersrand and printed by Printed Matter, for the Computer Society of South Africa and the South African Institute of Computer Scientists.
An Alternative Development of the Vienna Data Structures

N.C. PHILLIPS
University of Natal (Pietermaritzburg)
Department of Computer Science, Pietermaritzburg
3200

SYNOPSIS

This paper presents an axiomatic specification of a data type which is shown to be equivalent to the Vienna data structures. The advantage of this alternative approach is its obvious simplicity.

1. INTRODUCTION

The Vienna data structures, or Vienna objects, have been axiomized in [1] and elsewhere. In [1] the axioms are shown to be consistent by constructing a model for them, and the objects are then represented by rooted, edge-labelled trees with named terminal nodes. We can think of this work as an axiomatic specification of a data type which we shall call V-tree-µ. V-tree-µ consists of the Vienna objects with operators the selection functions and a path-constructing operator called µ in [1].

The intuition that the objects of this data type are determined by their paths may have suggested this specification in terms of the operator µ. The intuition that they are determined by objects "one level down from the root" and by pointers to these sub-objects leads to the alternative approach taken here. We shall give an axiomatic specification of a data type which we shall call V-tree-λ. The principle difference between V-tree-λ and V-tree-µ is that V-tree-λ has a sub-object-constructing operator λ instead of µ. We claim the following advantages for the axiomisation of V-tree-λ.

(1) It is simple.
(2) It is easy to see how to construct a model for it.
(3) It is easily shown that the axioms are categorical in that given the set of atoms, they determine the data type up to isomorphism.

Of course, V-tree-µ and V-tree-λ are different as data types since they have different operators, but with a little effort we shall show that they are equivalent in the sense that λ is definable in V-tree-µ, that µ is definable in V-tree-λ and, from a given set of atoms, V-tree-µ and V-tree-λ define the same set of objects.

2. SPECIFICATION OF V-TREE-µ AND V-TREE-λ

Each data type is specified in terms of two non-empty sets - a set S whose elements are called selectors and a set O whose elements are called objects. Furthermore, for each s ∈ S there is a corresponding function ś from O to O called a selection function. If A ∈ O we shall write sA for ś(A).

For V-tree-µ only, we construct (S*, I), the free monoid generated by S with identity element I and with the monoid operation denoted by concatenation. Elements of S* are called composite selectors. In addition, for each κ ∈ S* there is a corresponding selection function ̄κ defined on O. If A ∈ O we shall write κA for ̄κ(A). (In [1] no distinction is made between κ and ̄κr.)

In what follows: s, s₁, s₂, ... denote selectors, κ, κ₁, κ₂, ... denote composite selectors and A, B, C, ... denote objects.

Axioms, lemmas and definitions prefixed by µ are for the data type V-tree-µ and those prefixed by λ are for the data type V-tree-λ. Apart from minor rewriting we follow [1] closely
for V-tree-µ. Theorems and lemmas about V-tree-λ which are proved in [1] will not be proved again here.

µ Axiom 1. \( sA \in \Omega \)

µ Axiom 2. \((\forall s)(\kappa s)A = \kappa(sA)\)

µ Axiom 3. \( I\Lambda = \Lambda \)

µ Axiom 4. \((\exists A)(\forall s)[sA = A] \)

µ Axiom 5. \((\forall s)(sA = A) \Rightarrow (\forall B)(\exists \kappa)(\kappa B = A) \)

µ Lemma 1. \((\exists 1\Lambda)(\forall s)(sA = A) \)

λ Axiom 1. \((\exists 1\Lambda)(\forall s)(sA = A) \)

The object satisfying µ Lemma 1 and λ Axiom 1 is called the empty object and is denoted by Ω.

For both V-tree-µ and V-tree-λ we define the set of atoms, \( A \), to be \( \{ A : (\forall s)(sA = \Omega) \} \) and the set of elementary objects, \( \xi \), to be \( A \setminus \{ \Omega \} \). We assume that \( \xi \) is not empty and that no ordered pair is an element of an atom. In what follows, \( e, e_1, e_2, ... \) will denote elementary objects.

µ Axiom 6. \((\forall \kappa)(\forall e)(\kappa A = e \Leftrightarrow \kappa B = e) \Rightarrow A = B \)

λ Axiom 2. If \( A \not\in A \) then \((\forall s)(sA = sB) \Rightarrow A = B \)

For V-tree-µ we define: \( \kappa_1, \kappa_2 \) are dependent iff there is a \( \tau \) in \( S^* \) such that \( \kappa_1 = \tau \kappa_2 \) or \( \kappa_2 = \tau \kappa_1 \).

If \( \kappa_1, \kappa_2 \) are dependent we write \( \text{dep}(\kappa_1, \kappa_2) \).

µ Axiom 7. \((\exists B)(\kappa B = e \land (\forall \tau)(\text{dep}(\kappa, \tau) \Rightarrow \tau A) )\)

λ Axiom 3. If \( sA = \Omega \) and \( C \neq \Omega \) then \((\forall B)(\forall s_1)(s_1 B = s_1) \text{ if } s_1 = s \text{ then } C \text{ else } s_1 A \)

µ Lemma 2. The B satisfying µ axiom 7 is unique.

λ Lemma 3. The B satisfying λ axiom 3 is unique.

Proof.
Suppose that \( sA = \Omega \) and \( C \neq \Omega \) and that \( B_1 \) and \( B_2 \) satisfy λ axiom 3. Since \( sB_1 = C \) and \( C \neq \Omega \), \( B_1 \not\in A \), so \( B_1 = B_2 \) by λ axiom 2.

The B satisfying µ axiom 7 is denoted \( \mu(A, \kappa, e) \).

The B satisfying λ axiom 3 is denoted \( \lambda(A, s, C) \).

µ Axiom 8. \( O \) is the closure of \( A \) under µ.

λ Axiom 4. \( O \) is the closure of \( A \) under λ.

For ease of reference later, we now list the axioms for each data type.

V-tree-µ

µ1. \( sA \in O \)

µ2. \((\forall s)(\kappa s)A = \kappa(sA)\)

µ3. \( I\Lambda = \Lambda \)

µ4. \((\exists A)(\forall s)(sA = A) \)

- 22 -
\(\mu_5. \quad (\forall s)(sA = A) \Rightarrow (\forall B)(\exists k)(kB = A)\)

\(\mu_6. \quad (\forall k)(\forall e)[kA = e \Leftrightarrow kB = e] \Rightarrow A = B\)

\(\mu_7. \quad (\exists B)[kB = e \cap (\forall \tau)(\neg \text{dep}(k, \tau) \Rightarrow \tau B = \tau A)]\)

\(\mu_8. \quad O\) is the closure of \(A\) under \(\mu\).

\textbf{V-tree-\(\lambda\)}

\(\lambda_1. \quad (\exists 1 A)(\forall s)[sA = A]\)

\(\lambda_2. \quad \text{If } A \notin A \text{ then } (\forall s)[sA = sB] \Rightarrow A = B\)

\(\lambda_3. \quad \text{If } sA = \Omega \text{ and } C \neq \Omega \text{ then } (\exists B)(\forall s_1)[s_1 B = \text{if } s_1 = s \text{ then } C \text{ else } s_1 A]\)

\(\lambda_4. \quad O\) is the closure of \(A\) under \(\lambda\).

3. PROPERTIES OF V-TREE-\(\lambda\)

\(\lambda \quad \text{Definition 1:}\)

\(A^* = \{\text{if } A \in A \text{ then } A \text{ else } \{(s,C) : sA = C \land C \neq \Omega\}\}\)

\(A^*\) will be called the character of \(A\).

\(\lambda \quad \text{Lemma 4.}\)

(i) \(A^* = B^* \Rightarrow A = B\).

(ii) \((sA)^* = \text{if } (\exists C)(s,C) \in A^* \text{ then } C \text{ else } \Omega\).

(iii) \(\text{If } sA = \Omega \land C \neq \Omega \text{ then } \lambda(A, s, C)^* = \text{if } A \in A \text{ then } \{(s,C)\} \text{ else } A^* \cup \{(s,C)\}\).

\textbf{Proof.} \text{Recall our assumption that no ordered pair is an element of an atom.}

(i) \text{We show first that if one of } A, B \text{ is an atom and the other is not then } A^* \neq B^*. \text{For, if } B \text{ is not an atom there is } s \text{ and } a \text{ such that } sB = C. \text{Then } (s,C) \in B^*. \text{But if } A \text{ is an atom then } A^* = A, \text{ so } A^* \neq B^* \text{ since } (s,C) \notin A. \text{Now suppose that } A^* = B^* \text{. If } A, B \text{ are atoms, } A = B \text{ follows by definition of character; if } A, B \text{ are not atoms, } A = B \text{ follows from } \lambda \text{ axiom 2.}

(ii) \text{This follows at once from the definition of character (consider two cases, } A \in A \text{ and } A \notin A).\)

(iii) \text{This follows easily from the definition of character and the fact that } s_1 \lambda(A, s, C) = \text{if } s_1 = s \text{ then } C \text{ else } s_1 A.

The above lemma suggests that for a model of V-tree-\(\lambda\) we construct all possible characters. \(\lambda\) Axioms 3 and 4 suggest that characters are atoms or are non-empty finite sets \(\{(s_1, C_1), \ldots, (s_n, C_n)\}\) where \(s_1, \ldots, s_n\) are distinct selectors and \(C_1, \ldots, C_n\) are non-empty objects. Accordingly we recursively define the set \(O\) of objects to be the smallest set which contains \(A\) and all finite sets described above. Next we need to define the selection functions and the function \(\lambda\) on \(O\). Again the above lemma suggests how this should be done.

We define: \(sA = \text{if } (\exists C)(s,C) \in A \text{ then } C \text{ else } \Omega, \text{ and if } sA = \Omega \text{ and } C \neq \Omega \text{ we define:}\)

\(\lambda(A, s, C) = \text{if } A \in A \text{ then } \{(s,C)\} \text{ else } A \cup \{(s,C)\}.\)

It is now easy to check that \(O\), with selection functions and the operator \(\lambda\) defined above satisfies \(\lambda\) axioms 1 - 4.
Theorem 1. Any two models of V-tree-\( \lambda \) with the same set of atoms are isomorphic.

Proof.

\( \lambda \) Lemma 4 and \( \lambda \) axiom 4 show that any model of the \( \lambda \) axioms will be isomorphic to the model of characters under the map which takes each object to its corresponding character.

In [1] some pains are taken to represent objects as trees. We can see no harm in directly defining a V.DL tree to be the character of an object. We give below some pictures of VDL trees.

<table>
<thead>
<tr>
<th>Object</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>e</td>
</tr>
<tr>
<td>A = ( \lambda(\Omega, s, e) )</td>
<td>s</td>
</tr>
<tr>
<td>B = ( \lambda(A, s_1, e_1) )</td>
<td>s_1</td>
</tr>
<tr>
<td>C = ( \lambda(A, s_2, A) )</td>
<td>s_2</td>
</tr>
</tbody>
</table>

\( \Omega \) is the empty tree, its tree is the empty picture.

4. THE EQUIVALENCE OF V-TREE-\( \mu \) AND V-TREE-\( \lambda \)

We need a few more facts about V-tree-\( \mu \) before we can proceed. In V-tree-\( \mu \) the character set, \( \overline{A} \), of \( A \) is defined to be \{\( (\kappa, e) : \kappa A = e \}\}.

\( \mu \) Lemma 5.

(i) \( \overline{A} = \overline{B} \Rightarrow A = B \).

(ii) if \( \kappa_1 A = e = \kappa_2 A \) and \( \kappa_1 \neq \kappa_2 \) then \( \sim \text{dep}(\kappa_1 \kappa_2) \)

(iii) \( \mu(A, \kappa, e) = \{ (\kappa, e) \} \cup \{ (\tau, e) : \tau(A) = e \cap \sim \text{dep}(\tau, \kappa) \} \).

(iv) \( s\overline{A} = \{ (\tau, e) : (\exists \kappa)[(\kappa, e) \in \overline{A} \cap \kappa = \tau s] \} \).

Proof.

(i)-(iii) are proved in [1]. For (iv) observe that \( (\tau, e) \in s\overline{A} \) iff \( \tau(s\overline{A}) = e \) iff \( \kappa A = e \) where \( \kappa = \tau s \).

Theorem 2. \( \lambda \) is definable in V-tree-\( \mu \).
Proof:
Let $sA = \Omega$ and $C \neq \Omega$. We need to show in $V$-tree-$\mu$ that there is a unique object which satisfies $\lambda$ axiom 3. By $\mu$ axiom 8, the characteristic set $C$ of $C$ is finite, and it is non-empty because $C \neq \Omega$.

Let $C = \{(\kappa_1, e_1), \ldots, (\kappa_n, e_n)\}$. Define: $B_0 = A$ and $B_m = \mu(B_{m-1}, \kappa_m, s, e_m)$ for $1 \leq m \leq n$.

Note that, since $sA = \Omega$, if $t(A) = e$ then $\neg \text{dep}(t, \kappa_m)$ for $1 \leq m \leq n$. In particular when $m = 1$ this fact and $\mu$ Lemma 5 (iii) show that $B_1 = \{(\kappa_1, e_1)\} \cup A$.

Assume that $1 \leq m \leq n$ and $B_m = \{(\kappa_1, e_1), \ldots, (\kappa_m, e_m)\} \cup A$.

Note that by $\mu$ Lemma 5 (ii), $\neg \text{dep}(\kappa_j)$ for $1 \leq j \leq m$.

By $\mu$ Lemma 5 (iii) again,

$$B_{m+1} = \{(\kappa_{m+1}, s, e_{m+1})\} \cup \{(\tau, e) : \tau(B_m) = e \neg \text{dep}(\tau, \kappa_m)\}$$

$$= \{(\kappa_{m+1}, s, e_{m+1})\} \cup B_m$$

by the induction hypothesis and the noted facts about dependence

$$= \{(\kappa_1, e_1), \ldots, (\kappa_m, e_m)\} \cup A$$

by the induction hypothesis again.

Write $B$ for $B_n$. We have shown that $B = \{(\kappa_1, s, e_1), \ldots, (\kappa_n, s, e_n)\} \cup A$.

Hence, since $sA = \Omega$ and by $\mu$ Lemma 5 (iv), $s\overline{B} = \{(\kappa_1, s, e_1), \ldots, (\kappa_n, s, e_n)\} = \overline{C}$.

Thus $B$ satisfies $\lambda$ axiom 3.

Finally we must prove the uniqueness of $B$. Suppose that $D$ satisfies $\lambda$ axiom 3.

If $\kappa = t \tau s$ then $\kappa D = e$ iff $\kappa B = e$ because $sD = sB = \overline{C}$.

If $\kappa = t \tau s_1$ and $s_1 \neq s$ then $\kappa D = e$ iff $\kappa B = e$ because $s_1 D = s_1 B = \overline{A}$.

Therefore, by $\mu$ axiom 5 and since $D$ and $B$ are not atoms, $D = B$.

Theorem 3. $\mu$ is definable in $V$-tree-$\lambda$.

Proof. First we extend $V$-tree-$\lambda$ by defining a function $\overline{\kappa}$ for each composite selector $\kappa$ in accord with $\mu$ axioms 1-3. To do this we simply take $\mu$ axiom 3 as a definition and $\mu$ axiom 2 as a recursive definition of $\overline{\kappa}s$.

We now prove in $V$-tree-$\lambda$ that given $A, \kappa, e$ there is a unique $B$ which satisfies $\mu$ axiom 7.

If $\kappa = I$ then taking $e$ for $B$ satisfies $\mu$ axiom 7.

Suppose that $\kappa = s_1, \ldots, s_n$.

Define $B_0 = e$ and $B_m = \lambda(\Omega, s_m, B_{m-1})$ for $1 \leq m \leq n$.

If $A^* = A$ then $B = B_n$ satisfies $\mu$ axiom 7.

Suppose that $A^* = \{(s_1, C_1), \ldots, (s_m, C_m)\}$.

If $s_n \neq s_j$ for $1 \leq j \leq m$ then $B = \lambda(A, s_n, B_{n-1})$ satisfies $\mu$ axiom 7.

Suppose $s_n = s_1$ : we may assume without loss of generality that $j = 1$. 

- 25 -
If \( m = 1 \) then again \( B = B_n \) satisfies \( \mu \) axiom 7.

If \( m > 1 \) then define \( D_2 = \lambda(\Omega s^1 2, C_2) \) and \( D_{i+1} = \lambda(D_i s^1 i+1, C_{i+1}) \) for \( 1 < i < m \).

Now \( B = \lambda(D_m s_n B_{n-1}) \) satisfies \( \mu \) axiom 7.

It remains to prove uniqueness. Suppose \( D \) satisfies \( \mu \) axiom 7.

Consider any \( \tau \) such that \( \tau \neq \kappa \) and \( \tau D \in \xi \).

\( \tau \neq \tau_1 \kappa \) since otherwise \( \tau D = \tau_1 \kappa D = \tau_1 e = \Omega \)

\( k \neq \tau_1 \tau \) since otherwise \( k D = \tau_1 \tau D = \Omega \) since \( \tau D \in \xi \).

Thus \( \sim \text{dep}(\tau, \kappa) \), so \( \tau D = \tau A = \tau B \).

We have shown: \( (\forall \tau)(\forall e)[\tau D = e \iff \tau B = e] \Rightarrow A = B \).

Now \( D = B \) follows from \( \lambda \) Lemma 6 which we prove below.

**\( \lambda \) Lemma 6.** \( (\forall \tau)(\forall e)[\tau B = e \iff \tau B = e] \Rightarrow A = B \)

**Proof.** \( \lambda \) Axiom 4 justifies the following recursive definition of depth:

\[
\text{depth } A = \begin{cases} 
0 & \text{if } A \in A \\
1 + \max \{ \text{depth } sA : sA \neq \Omega \} & \text{else}
\end{cases}
\]

Suppose \( (\forall \tau)(\forall e)[\tau A = e \iff \tau B = e] \).

If \( A = \Omega \) then \( A = B \) since \( A = \Omega \iff (\forall \tau)[\tau A = \Omega] \).

If \( A \in \xi \) then \( A = B \) since \( (\forall \tau)[\tau A \neq \Omega \iff \tau = I] \).

Thus \( A = B \) if depth \( A = 0 \). Suppose depth \( A > 0 \).

For any \( s \), \( (\forall \tau)(\forall e)[\tau sA = e \iff \tau sB = e] \) so by our induction hypothesis, \( (\forall s)[sA = sB] \). So by \( \lambda \) axiom 2, \( A = B \).

**Theorem 4.** Given \( A \), V-tree-\( \mu \) and V-tree-\( \lambda \) determine the same set of objects (up to isomorphism).

**Proof.** In view of theorems 2 and 3 it remains to show that the closure of \( A \) under \( \mu \) is the closure of \( A \) under \( \lambda \). But theorem 2 implies that the closure of \( A \) under \( \lambda \) is contained in the closure of \( A \) under \( \mu \) and theorem 3 implies that the closure of \( A \) under \( \mu \) is contained in the closure of \( A \) under \( \lambda \).

5. CONCLUSION

There are close connections between the work presented here and other descriptions of the Vienna objects [2, 3]. The axioms for V-tree-\( \lambda \) are similar to the "ground axioms" in Standish [4]. Models of V-tree-\( \lambda \) are precisely the "constructive models" of [4]. Section 4 above gives the precise relationship between the Vienna objects and the "constructive models" of [4].

The VHL language QUADLISP [6] which is under development at the University of South Africa requires tree-like objects which are considerably more complicated than the VDL trees. The selector which pointed me to this work was invoked when S.W. Postma asked me for a specification of the QUADLISP objects.

REFERENCES

1. OLLONGREN, O. (1974). Definition of Programming Languages by Interpreting Automata,
2. LUCAS, P., LAUER, P. and STIGLEITNER, H. (1968). Method and Notation for the formal Definition of Programming Languages, TR 25.087, IBM Research Laboratory, Vienna.
The purpose of this journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review articles, exploratory articles of general interest to readers of the Journal. The preferred languages of the Journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be submitted in triplicate to:
Prof. G. Wiechers at:
INFOPLAN
Private Bag 3002
Monument Park 0105

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. The original ribbon copy of the typed manuscript should be submitted. Authors should write concisely.

The first page should include the article title (which should be brief), the author's name, and the affiliation and address. Each paper must be accompanied by an abstract less than 200 words which will be printed immediately below the title at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review categories.

Tables and figures

Illustrations and tables should not be included in the text, although the author should indicate the desired location of each in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Illustrations should also be supplied on separate sheets, and each should be clearly identified on the back in pencil with the Author's name and figure number. Original line drawings (not photoprints) should be submitted and should include all relevant details. Drawings etc., should be submitted and should include all relevant details. Drawings etc., should be about twice the final size required and lettering must be clear and "open" and sufficiently large to permit the necessary reduction of size in block-making.

Where photographs are submitted, glossy bromide prints are required. If words or numbers are to appear on a photograph, two prints should be sent, the lettering being clearly indicated on one print only. Computer programs or output should be given on clear original printouts and preferably not on lined paper so that they can be reproduced photographically.

Figure legends should be typed on a separate sheet and placed at the end of the manuscript.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters; between the letter G and zero; between the letter I, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetical order of author's name, and cited in the text by number in square brackets. Journal references should be arranged as thus:


Proofs and reprints

Galley proofs will be sent to the author to ensure that the papers have been correctly set up in type and not for the addition of new material or amendment of texts. Excessive alterations may have to be disallowed or the cost charged to the author. Corrected galley proofs, together with the original typescript, must be returned to the editor within three days to minimize the risk of the author's contribution having to be held over to a later issue.

Only original papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.

Hierdie notas is ook in Afrikaans verkrygbaar.