A Non-Commutative Multiple-Valued Logic

R. J. Bignall

School of Applied Science
Monash University College Gippsland
Switchback Road, Churchill 3842
Australia

Abstract
A set of operations which can be used to design n-valued switching functions is given. These give rise to a class of algebras which are left handed skew lattices together with dual implication operation. Such algebras form a decidable discriminator variety, and hence possess a well behaved structure theory and satisfy many identities. Algorithms for the design and optimization of switching functions are outlined.

1 Introduction
Boolean skew lattices generalize Boolean lattices, that is, relatively complemented distributive lattices with a zero element. A Boolean skew lattice may be defined as a distributive skew lattice with zero, such that its maximal lattice homomorphic image is Boolean lattice. A class of such lattices was studied by W. H. Cornish in [4]. By adding a relative complementation operation he obtained a variety of algebras called Boolean Skew Algebras.

Algebras with implication operations, and dual implication operations, have been studied by a number of authors; for some examples of implicative algebras the reader is referred to chapter II of [14]. BCK-algebras are algebras with an operation which is an abstraction of set subtraction. Implicative BCK-algebras are a subclass of these, and they are known to form a variety. Thus an implicative BCK-algebra may be defined as an algebra \((A; /, 0)\) of type \((2, 0)\) satisfying the identities

\[
\begin{align*}
(i) & \quad x / z \approx 0 \\
(ii) & \quad x / (x / y) \approx y / (y / x) \\
(iii) & \quad (x / y) / z \approx (x / z) / y \\
(iv) & \quad x / (y / x) \approx z.
\end{align*}
\]

An implicative BCK-algebra is partially ordered by the relation \(\subseteq\) given by \(a \subseteq b\) if \(a/b = 0\). Under this partial order \((A, \subseteq)\) is a lower semilattice with smallest element \(0\), with the property that any principal order ideal is a Boolean lattice with a greatest element, and hence is order isomorphic to a Boolean algebra. For details see Yutani [19], Iseki and Tanaka [7], or Cornish [5], and the references cited therein.

In Section 4 of this paper a variety of algebras called Quasi-Boolean Skew Algebras is described. These algebras are left handed skew lattices with an implicative BCK operation. The class of all such algebras forms a discriminator variety, and thus they more closely resemble Boolean algebras than do Boolean skew algebras. There is, up to isomorphism, precisely one subdirectly irreducible quasi-Boolean skew algebra for each cardinal \(n \geq 2\). For finite \(n\) each such algebra gives rise to an associated n-valued logic.

The notation used in this paper follows that of [3], except when otherwise stated. The reader should refer to chapter IV of that book for details about discriminator varieties.

2 Sums and Products
The purpose of this section is to briefly summarise some definitions and results which are central to multiple-valued switching theory.

Let \(A\) be a set containing at least two elements. For ease of notation we assume that \([0, 1] \subseteq A\). When \(A\) has finite cardinality \(r\) we denote its elements by \(0, 1, \ldots, \; R\), where \(R = r - 1\). Following Muzio and Wesselkamper, [13, Chapter 3], let \(+, \cdot\) and \(J\) be binary operations on \(A\) which satisfy

\[
\begin{align*}
x + 0 &= x = z + x = x \\
x \cdot 0 &= 0 = z \cdot x = z \\
z \cdot 1 &= z
\end{align*}
\]

\(J(x, y) = \begin{cases} 1 & \text{if } x = y; \\ 0 & \text{otherwise.} \end{cases}\)

+ is called a sum-type operation and \(\cdot\) is called a product-type operation. \(J\) is called the two place characteristic function.

To avoid excessive use of parentheses, let \(\cdot\) have precedence over \(+\) in expressions which involve both operations. Where multiple sums or products are involved, the omission of parentheses implies that association is from the left.

The "sums of products" theorem for finite \(A\) — see [13, page 42] gives an algorithm for writing any function...
In this section we give a set of binary operations which are functionally complete, and which may offer some significant advantages when applied to the problem of designing switching functions. These operations appear to have straightforward implementations in at least one multiple-valued switching technology (namely peristaltic CCD logic), and their algebraic properties make them quite suitable from the point of view of designing and optimizing r-valued switching functions.

Let $A$ be as in the previous section. Define the operations $V, \land, /$ and $\backslash$ on $A$ by

$$a \lor b = \begin{cases} b & \text{if } b \neq 0; \\ a & \text{otherwise}. \end{cases}$$

$$a \land b = \begin{cases} a & \text{if } a \neq 0; \\ 0 & \text{otherwise}. \end{cases}$$

$$a/b = \begin{cases} a & \text{if } a \neq b; \\ 0 & \text{otherwise}. \end{cases}$$

One easily checks that $V$ is a sum-type operation and $\land$ is a product-type operation with respect to which any non zero element is a right identity. We call these operations join and inf, since they give the maximum and minimum with respect to a quasiorder on $A$, where

$$x \leq y \iff x \lor y = x \quad \text{and} \quad y \leq x \iff y \lor x = y$$

A quasiorder is a relation which is reflexive and transitive. A quasiorder which is also anti-symmetric is a partial order. The equivalence relation given by

$$x \equiv y \iff x \leq y \quad \text{and} \quad y \leq x$$

has two equivalence classes, $[0]$ and $[x]$, where $x$ is any value $\neq 0$. The algebra $(A, V, \land, 0)$ is a skew lattice with zero, that is, a non-commutative lattice with 0 being a unique minimal element. For information about skew lattices the reader is referred to [11]. See also [9], [6], [16], [17], [1] and [2]. The $V$ and $\land$ operations satisfy a number of identities in addition to the skew lattice identities given in [11]. In particular, the inf operation distributes over joins.

One easily checks that the algebra $(A, /, 0)$ of type $(2, 0)$ satisfies the implicatve BCK identities given in the section 1. The operation $\backslash$ satisfies

$$a \backslash b = \begin{cases} a & \text{if } b = 0; \\ 0 & \text{if } b \neq 0. \end{cases}$$

Recall that the (left handed) discriminator on a set $A$ is the function $d : A^3 \to A$ given by

$$d(a, b, c) = \begin{cases} c & \text{if } b \neq c; \\ a & \text{otherwise}. \end{cases}$$
Many authors prefer to use the right handed discriminator:
\[ t(a, b, c) = \begin{cases} a, & \text{if } a \neq b; \\ c, & \text{otherwise}. \end{cases} \]

Clearly this is a matter of taste, since \( d(a, b, c) = t(c, b, a) \). In this paper “discriminator” will mean the left handed discriminator.

An algebra \( A \) is called a discriminator algebra if there is a term \( d(x, y, z) \) whose canonical interpretation on \( A \) is the discriminator function. A discriminator algebra is necessarily simple, and a finite discriminator algebra is functionally complete, because of a result of H. Werner in [18].

Let \( A_0 \) denote the algebra \((A; V, \wedge, /, 0)\), where the operations on \( A \) are as defined above.

Lemma 3.1. \( A_0 \) is a discriminator algebra.

Proof. One easily checks that the discriminator is given by
\[ d(a, b, c) = (a/b) \vee (a \wedge c) \vee (c/b). \]

It follows that the operations \( V, \wedge \) and \( / \) form a functionally complete set on \( A \). An alternative way of seeing this is to observe that the two place characteristic function on \( A \) is given by
\[ J(a, b) = \left((1 \wedge a) \wedge (1 \wedge b)\right) \vee \left((1 \wedge (b/(b/a)))\right) \]

As the above observations suggest, the operations defined in this section satisfy many identities. Some of these identities are given in the next section. A more detailed investigation of these operations and their applications in algebra appears in [2].

We conclude this section with a brief summary of some facts about discriminator varieties which will be needed later. For further details see Chapter IV of Burris and Sankappanavar [3].

If \( K \) is a class of algebras of the same type with a common discriminator term then the variety \( V = V(K) \) generated by \( K \) is called a discriminator variety. Equivalently, a variety \( V \) is a discriminator variety if there is a term \( d(x, y, z) \) which realizes the discriminator on every subdirectly irreducible member of \( V \). Thus if \( d_1(x, y, z) \) and \( d_2(x, y, z) \) are two discriminator terms for \( V \) then \( V \models d_1(x, y, z) \approx d_2(x, y, z) \). The following useful fact is easily proved. It is a consequence, for example, of the characterization of discriminator varieties given by R. McKenzie in [12, Theorem 1.3].

Lemma 3.2. If \( A \) is a member of a discriminator variety with a discriminator term \( d(x, y, z) \), then \( \text{Con } A = \text{Con } (A; d_A) \).

4 Quasi-Boolean Skew Algebras

Definition 4.1. A Quasi-Boolean Skew Algebra is an algebra \( A = (A; V, \wedge, /, 0) \) of type \((2, 2, 2, 0)\) such that the following identities hold:
\[
\begin{align*}
(4.1) & \quad x \wedge z \approx x \\
(4.2) & \quad x \wedge (y \wedge z) \approx (x \wedge y) \wedge z \\
(4.3) & \quad z \wedge (x \vee y) \approx (z \wedge x) \vee (z \wedge y) \\
(4.4) & \quad (x \vee y) \wedge z \approx (x \wedge z) \vee (y \wedge z) \\
(4.5) & \quad (y \wedge z) \vee z \approx z \\
(4.6) & \quad z/x \approx 0 \\
(4.7) & \quad (z/x) \wedge z \approx (z \wedge x)/(y \wedge z) \\
(4.8) & \quad z \vee (y/z) \approx z \vee y \\
(4.9) & \quad (z \vee y)/z \approx y/z \\
(4.10) & \quad z \approx (x \wedge y) \vee ((z \vee y)/y)
\end{align*}
\]

The variety of quasi-Boolean skew algebras will be denoted by \( \text{QBSA} \).

4.1 Examples

4.1.1 Commutative QBSA's

i. Let \( B = (B; V, \wedge', 0, 1) \) be a Boolean algebra. Put \( a/b = a \wedge b' \). Then \( B_0 = (B; V, \wedge', 0) \) is a QBSA.

ii. Let \( L = (L; V, \wedge, 0) \) be a relatively complemented distributive lattice with zero. Then \( L_0 = (L; V, \wedge, 0) \) is a QBSA when \( a/b \) is defined to be the relative complement of \( a \wedge b \) in the interval \([0, a] \).

Both of the above examples give commutative QBSAs. It is shown in [2] the lattice reduct of a commutative QBSA is always a Boolean lattice.

4.1.2 Discriminator Algebras

Proposition 4.3. \( A_0 \) is a quasi-Boolean skew algebra, and \( d \) is the left discriminator on \( A \). Hence \( A_0 \) is simple.

Proof. Using case splitting arguments, it is not difficult to check that \( A_0 \) satisfies the identities of Definition 4.1, and we have shown in the previous section that the \( d(a, b, c) \) is the left discriminator on \( A \).

We call \( A_0 \) the flat QBSA with base set \( A \).

4.1.3 Algebras in Discriminator Varieties

More general examples of QBSA's are given by the following proposition. The definitions of \( V, \wedge, \) and \( / \) given below were used by K. Keimel and H. Werner in [10].

Proposition 4.4. Let \( V \) be a discriminator variety, and let \( A \in V \). Let \( d(x, y, z) \) be the discriminator term for \( V \). Select any element \( 0 \in A \) and define the operations \( V, \wedge, / \) on \( A \) as follows
\[
\begin{align*}
av b & = d_A(a, 0, b) \\
\text{a} \wedge b & = d_A(a, d_A(a, 0, b), b)
\end{align*}
\]
Thus it is simple matter to check that in this case

In the general case, let $A$ be a QBSA, and

subdirect product of flat QBSA's, and

Proof. A.

The final assertion of the proposition now follows from

Ao/B

Ao,

Then each maximal

subdirect product in

QBSA. Let $t(x, y, z)$ be the

Con $A_0$.

By

part of the proof. Then

where

Con $A_0$

is the bijection

$V(x/y)$. Because of Proposition

4.6.

The following are equivalent

any

elements

$A$, $B$

of $A$.

Proposition 4.7. The following are equivalent for $A \in$

QBSA such that $|A| \geq 2$.

(i) $A$ is subdirectly irreducible.

(ii) $M(A)$ has two elements.

(iii) For any $x, y \in A$, $y \neq 0$ implies

$x \wedge y = x$ and $x \vee y = y$.

(iv) $A$ is a flat QBSA.

(v) $d(A, y, z)$ is the left discriminator on $A$.

(vi) $A$ is simple.

Proof. Further examples are given in [2], where proofs of the remaining results in this section also appear.

Proposition 4.5. Let $A$ denote an arbitrary QBSA. Let $a/b = (a \vee b)/b$. Then $A$ satisfies the identities

\begin{align*}
(4.11) & \quad x \vee z \approx x \\
(4.12) & \quad x \vee (y \vee z) \approx (x \vee y) \vee z \\
(4.13) & \quad x \wedge (y \vee z) \approx (x \wedge y) \vee z \\
(4.14) & \quad x \wedge (y \vee z) \approx (x \wedge y) \wedge z \\
(4.15) & \quad (y \wedge z) \approx z \wedge x \\
(4.16) & \quad (x \wedge y) \vee z \approx (x \vee z) \wedge (y \vee z) \\
(4.17) & \quad x \wedge 0 \approx 0 \wedge 0 \\
(4.18) & \quad x \vee 0 \approx 0 \vee x \\
(4.19) & \quad x/0 \approx x \\
(4.20) & \quad 0/x \approx 0 \\
(4.21) & \quad x/0 \approx x \\
(4.22) & \quad 0/x \approx 0
\end{align*}

Proposition 4.6. The following are equivalent for any elements $a, b \in A$, where $A \in QBSA$.

\begin{align*}
(a) & \quad a/b = a/b \\
(b) & \quad a \wedge b = b \wedge a = a \\
(c) & \quad a \vee b = a \wedge b = b \\
(d) & \quad a/b = b/a = 0 \\
(e) & \quad a/b \vee b/a = 0
\end{align*}

Proof. Further examples are given in [2], where proofs of the remaining results in this section also appear.

Proposition 4.7. The following are equivalent for $A \in$

QBSA such that $|A| \geq 2$.

(i) $A$ is subdirectly irreducible.

(ii) $M(A)$ has two elements.

(iii) For any $x, y \in A$, $y \neq 0$ implies

$x \wedge y = x$ and $x \vee y = y$.

(iv) $A$ is a flat QBSA.

(v) $d(A, y, z)$ is the left discriminator on $A$.

(vi) $A$ is simple.

Proof. Further examples are given in [2], where proofs of the remaining results in this section also appear.

Proposition 4.5. Let $A$ denote an arbitrary QBSA. Let $a/b = (a \vee b)/b$. Then $A$ satisfies the identities

\begin{align*}
(4.11) & \quad x \vee z \approx x \\
(4.12) & \quad x \vee (y \vee z) \approx (x \vee y) \vee z \\
(4.13) & \quad x \wedge (y \vee z) \approx (x \wedge y) \vee z \\
(4.14) & \quad x \wedge (y \vee z) \approx (x \wedge y) \wedge z \\
(4.15) & \quad (y \wedge z) \approx z \wedge x \\
(4.16) & \quad (x \wedge y) \vee z \approx (x \vee z) \wedge (y \vee z) \\
(4.17) & \quad x \wedge 0 \approx 0 \wedge 0 \\
(4.18) & \quad x \vee 0 \approx 0 \vee x \\
(4.19) & \quad x/0 \approx x \\
(4.20) & \quad 0/x \approx 0 \\
(4.21) & \quad x/0 \approx x \\
(4.22) & \quad 0/x \approx 0
\end{align*}

Proposition 4.6. The following are equivalent for any elements $a, b \in A$, where $A \in QBSA$.

\begin{align*}
(a) & \quad a/b = a/b \\
(b) & \quad a \wedge b = b \wedge a = a \\
(c) & \quad a \vee b = a \wedge b = b \\
(d) & \quad a/b = b/a = 0 \\
(e) & \quad a/b \vee b/a = 0
\end{align*}

Proof. Further examples are given in [2], where proofs of the remaining results in this section also appear.

Proposition 4.7. The following are equivalent for $A \in$

QBSA such that $|A| \geq 2$.
an equational base for $V_n$ is given by the identities of Definition 4.1, together with the identity $\epsilon_n$. □

The case $n = 1$ is of interest, since the $\vee$ and $\wedge$ operations on $A_1$ are commutative. There are many different identities which are equivalent to $\epsilon_1$ modulo the QBSA identities. In fact, any member of $V_1$ is of the form $B_0$ for some Boolean algebra $B$, where $B_0$ is constructed as in example 4.1.1. The next result follows immediately from Theorem 4.12, and Theorem 4.4 of [4]

Theorem 4.13. The following QBSA identities are all equivalent. They hold on $A \in QBSA$ if and only if $A \in V_1$.

(i) $\epsilon_1$
(ii) $z \vee y = y \vee z$
(iii) $z \wedge y = y \wedge z$
(iv) $z \vee (y \wedge z) = x$
(v) $(x \vee y) \wedge z = x$
(vi) $z \vee (x \wedge y) = (z \vee x) \wedge (z \vee y)$ □

5 Switching Functions

In this section we discuss how the operations given in section 2 can be used to design switching functions. Let $r > 1$ and let $A$ be the set $\{0, 1, \ldots, R\}$, where $R = r - 1$. The problem is to express an arbitrary function $f : A^r \rightarrow A$ as a polynomial function of the algebra $(A; \vee, \wedge, 0, 1, \ldots, R)$, or equivalently, as an algebraic function of the flat QBSA with base set $A$.

If we use the sums of products method, then it is clear that any such function $f$ can be written in disjunctive form. By this is meant a join of product expressions; that is, expressions of the form $\bigvee_{i \in I} t_i$, where $t_i$ is a polynomial in $x_1, \ldots, x_n$, in which the only operations used are $\vee$, $\wedge$ and the constants.

Note that $\vee$ and $\wedge$ are associative operations. Also, the inf operation satisfies the identity $x \wedge y \wedge z \approx x \wedge z \wedge y$, so that in a product expression all terms after the first one commute.

In view of the discussion in section 2, and the fact that any non-zero element of $A$ is a right identity for the inf operation, we have

$$f(x_1, \ldots, x_n) = \bigvee_{i=1}^R i \wedge f_i(x_1, \ldots, x_n)$$

where $f_i(x_1, \ldots, x_n)$ takes non-zero values when $f(x_1, \ldots, x_n) = i$, and is zero otherwise.

As is usual in a sums of products algorithm, the number of terms in the expression for $f$ can be reduced by searching the truth table of this function for prime implicants. In our case, prime implicants are given by product terms in which each $t_i$ has one of the forms $j, x_1 x_2, \ldots, j \vee x_1$, for $j \in \{1, \ldots, R\}$.

Further optimization is possible, however, because of the following decomposition rule, which is an immediate consequence of the definition of the sup operation. We write $\bar{x}$ for $x_1, \ldots, x_n$.

Lemma 5.1. If $f$ and $g$ are any two $n$-ary functions such that $g(\bar{x})$ is zero whenever $f(\bar{x})$ is, then $f(\bar{x})$ can be written as

$$f(\bar{x}) = g(\bar{x}) \vee h(\bar{x})$$

where

$$h(\bar{x}) = \begin{cases} 0 & \text{if } g(\bar{x}) = f(\bar{x}); \\ f(\bar{x}) & \text{otherwise.} \end{cases}$$

In view of this, it is not necessary for earlier products to be disjoint from later ones in a sum of product terms. This allows the use of "don't care" entries in the truth tables for the above functions $f_i$, which in turn, reduces the complexity of its prime implicants. The following example for a function of two variables, $z, y$ on four logic values illustrates how such decompositions may be employed.

Let $f(z, y)$ be given by the following table.

<table>
<thead>
<tr>
<th>$z$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>$f(z, y)$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Then $f(z, y) = f_1(z, y) \vee f_2 \vee f_3(z, y)$, where $f_1, f_2$ and $f_3$ are given as follows.

<table>
<thead>
<tr>
<th>$z$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>$f_1(z, y)$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$z$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>$f_2(z, y)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$z$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>$f_3(z, y)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

We have

$$f_1(z, y) = (1 \wedge 2/y \wedge 3/y) \vee (1/x)$$
$$f_2(z, y) = 2 \wedge 1/z \wedge 2$$
$$f_3(z, y) = (3 \wedge 2/2z/3 \wedge y/2 \wedge y/3) \vee (3 \wedge z/1/y/1/y/2)$$

In the case of $f_3(z, y)$ a further reduction can be achieved by making use of the fact that the inf operation distributes over sums from both the left and the right. Thus we can write

$$f_3(z, y) = 3 \wedge ((z/2 \wedge z/3 \wedge y/3) \vee (z/1 \wedge y/1)) \wedge y/2$$

53
Acknowledgement

Some of the research for this paper was undertaken while the author was a visiting Fellow in the Department of Computer Science at the University of New South Wales. The facilities and assistance provided by the University and the Department are gratefully acknowledged.

References


