Semantics of quasi-boolean expressions

Introduction

In deriving programs, it is often useful to consider certain expressions as well-defined although they contain subexpressions to which no value can reasonably be attributed. For instance, one wishes to consider the expression

$$0 \leq i < n \text{ and } a[i] = 0$$

as false when $i = n$, without worrying whether or not n is in the subscript range of array a . Expressions like these are called quasi-boolean in [2]. Traditionally, they are treated by introducing the conditional connectives cand and cor [1, chapter 4] [3, § 4.1].

However, cand and cor do not satisfy pleasant algebraic laws: they are not commutative and do not in general distribute over each other. This makes calculations involving these connectives exceedingly burdensome; for this reason their introduction was termed "a strategic mistake" in [2]. It is the purpose of this note to suggest a possible alternative.

Let it be clear from the outset that the semantics of quasi-boolean expressions defined below is not intended for implementation in a programming language. Its aim is merely to facilitate the calculational derivation of programs. That these derivations are most easily expressed in a richer language than the programs themselves, one that need not be implementable, is a well-known observation.

On the other hand, in order to be suited for its purpose, any proposed semantics of quasi-boolean expressions should satisfy the following criteria:

- (i) if all atomic subexpressions of a certain expression are well-defined, the value attached to that expression should be the one that would follow from ordinary logic;
- (ii) no expression should be left undefined if it can be given a meaning by the introduction of conditional connectives;
- (iii) the relation "having the same value" between expressions should satisfy simple calculational laws.

The option of considering an expression undefined when any of its subexpressions is, satisfies (i) and (iii) but not (ii). The use of conditional connectives satisfies (i) and (ii) but not (iii). It is possible to do a bit better by considering two-sided conditional connectives [4] [0, § A0.3.2]. This restores commutativity, but, as we shall see in the next section, there remain important laws that fail to hold. The semantics defined in this note has the property that all laws of propositional calculus carry through.

Conventions and notations

We suppose a set At of logical atoms and a valuation $av \in At \rightarrow \{F, U, T\}$ given: this is meant to model the fact that formulae without logical connectives can be false, undefined or true respectively. For convenience's sake it is assumed that F and T do not themselves belong to At . The problem dealt with here is how to extend the given valuation av if logical connectives are introduced. Let Ex be the set of all finite expressions that can be formed with the elements of $At \cup \{F, T\}$, the unary operator non and the binary operators and and or. As yet, no meaning is attached to the elements of Ex ; in fact, discovering how to define such a meaning in a sensible way is precisely our purpose. (The formal operators non, and, or must not be confused with those of the ordinary logic in which our proofs will be expressed. These will be denoted by the symbols $\neg$, $\wedge$, $\vee$.)

We introduce a linear ordering $\sqsubseteq$ on the set $\{F, U, T\}$ such that $F \sqsubseteq U \sqsubseteq T$. For p and q in $\{F, U, T\}$, denote by $p \sqcap q$ the minimum and by $p \sqcup q$ the maximum of p and q with respect to the ordering $\sqsubseteq$. Furthermore, the prefix operator $\sim$ is defined by $\sim F = T$, $\sim U = U$, $\sim T = F$.

For $v \in At \rightarrow \{F, U, T\}$, we define $c.v \in Ex \rightarrow \{F, U, T\}$ by

- (0) $E \in \{F, T\} \Rightarrow c.v.E = E$,
- (1) $E \in At \Rightarrow c.v.E = v.E$,
- (2) $c.v.(\text{non } E) = \sim(c.v.E)$,
- (3) $c.v.(E0 \text{ and } E1) = c.v.E0 \sqcap c.v.E1$,
- (4) $c.v.(E0 \text{ or } E1) = c.v.E0 \sqcup c.v.E1$

for all $E, E0, E1 \in Ex$.

What we are looking for is a mapping $ev \in Ex \rightarrow \{F, U, T\}$ with $ev|_{At} = av$ that satisfies criteria (i)-(iii). An obvious candidate for ev would be $c.av$; in fact, this is precisely what is meant in the introduction by "two-sided conditional connectives". However, as

$$\begin{aligned} U \sqcap (\sim U \sqcup F) &= U, \\ U \sqcap F &= F, \end{aligned}$$

the complement rule

$$ev.(E0 \text{ and } (\text{non } E0 \text{ or } E1)) = ev.(E0 \text{ and } E1)$$

would fail to hold with ev chosen as $c.av$. Therefore this choice is not considered satisfactory.

In fact, there is a simple way to ensure that ev attaches the same value to expressions that are equivalent in propositional calculus. Note that for $w \in At \rightarrow \{F, T\}$, rules (0)-(4) define $c.w \in Ex \rightarrow \{F, T\}$. Now the restrictions of $\sim, \sqcap, \sqcup$ to $\{F, T\}$ model ordinary logic. Hence, if $E0$ and $E1$ can be transformed into one another by the rules of propositional calculus, then for any $w \in At \rightarrow \{F, T\}$ we have $c.w.E0 = c.w.E1$. If for $ev \in Ex \rightarrow \{F, U, T\}$ the implication

$$(5) \quad (\underline{A} w: w \in At \rightarrow \{F, T\}: c.w.E0 = c.w.E1) \Rightarrow ev.E0 = ev.E1$$

holds, criterion (iii) is satisfied in the sense that all rules of propositional calculus apply. The rest of this note is concerned with the construction of such an ev .

Definition of ev

For $w \in At \rightarrow \{F, T\}$ we define

$$(6) \quad f.w \equiv (\underline{A} x: x \in At \wedge av.x \neq U: w.x = av.x).$$

(Actually, the $\wedge$ in (6) should strictly speaking be a conditional and, since $av.x$ is undefined when $x \in At$ is false. However, at the end of this note we shall have succeeded in showing that such scruples are unnecessary.)

For $E0, E1 \in Ex$ we define

$$(7) \quad E0 == E1 \equiv (\underline{A} w: w \in At \rightarrow \{F, T\} \wedge f.w: c.w.E0 = c.w.E1).$$

For later use we note that $==$ is an equivalence relation on Ex .

Now we are ready to define the mapping $ev \in Ex \rightarrow \{F, U, T\}$ we have been looking for. For any $E \in Ex$,

$$(8) \quad \begin{aligned} ev.E &= \underline{if} \ E==F \rightarrow F \\ &\quad \square \ E==T \rightarrow T \\ &\quad \square \ \neg(E==F) \wedge \neg(E==T) \rightarrow U \\ &\quad \underline{fi} \end{aligned}$$

The rest of this note is organised as follows. First we show that (8) is a proper definition in the sense that the guards are mutually exclusive (Theorem 0). Then we show that ev as defined by (8) indeed satisfies $ev|_{At} = av$ (Theorem 1). Finally, we deal with criteria (i)-(iii) (Theorems 2-4).

Theorems and proofs

THEOREM 0. For $E \in Ex$,

$$\neg(E==F) \vee \neg(E==T) \quad .$$

PROOF.

$$\begin{aligned} &\neg(E==F) \vee \neg(E==T) \\ \Leftarrow &\quad \{ == \text{ is an equivalence relation} \} \\ &\neg(F==T) \\ = &\quad \{(7)\} \\ &(\underline{E} \ w: w \in At \rightarrow \{F, T\} \wedge f.w: c.w.F \neq c.w.T) \\ = &\quad \{(0)\} \\ &(\underline{E} \ w: w \in At \rightarrow \{F, T\} \wedge f.w: F \neq T) \\ = &\quad \{\text{nonempty domain}\} \\ &F \neq T \quad . \end{aligned}$$

□

THEOREM 1. For $x \in At$,

$$ev.x = av.x \quad .$$

PROOF. For $q \in \{F, T\}$,

$$\begin{aligned} &ev.x = q \\ = &\quad \{(8)\} \\ &x==q \\ = &\quad \{(7)\} \\ &(\underline{A} \ w: w \in At \rightarrow \{F, T\} \wedge f.w: c.w.x = c.w.q) \\ = &\quad \{(1), (0)\} \\ &(\underline{A} \ w: w \in At \rightarrow \{F, T\} \wedge f.w: w.x = q) \\ = &\quad \{(6), \text{nonempty domain}\} \\ &av.x = q \quad . \end{aligned}$$

□

We have now reached the point where it has been established that (8) really does

define a mapping $ev \in Ex \rightarrow \{F, U, T\}$ with $ev|_{At} = av$. Next we turn to criterion (iii).

As observed in a previous section, in order to show that ev attaches the same value to expressions that are equivalent in propositional calculus, it is sufficient to show that ev satisfies (5). This will be a corollary of the next theorem.

THEOREM 2. For $E0, E1 \in Ex$,

$$E0 == E1 \Rightarrow ev.E0 = ev.E1.$$

PROOF.

$$\begin{aligned} & E0 == E1 \\ \Rightarrow & \{ == \text{ is an equivalence relation} \} \\ & (E0 == F \equiv E1 == F) \wedge (E0 == T \equiv E1 == T) \\ \Rightarrow & \{(8)\} \\ & ev.E0 = ev.E1. \end{aligned}$$

□

REMARK. The implication in the theorem is not an equivalence. If x and y are distinct elements of At such that $av.x = av.y = U$, then $ev.x = ev.y$ but not $x == y$.

COROLLARY. ev satisfies (5).

PROOF.

$$\begin{aligned} & (5) \\ = & \{ \} \\ & (\underline{A} w: w \in At \rightarrow \{F, T\}: c.w.E0 = c.w.E1) \Rightarrow ev.E0 = ev.E1 \\ \Leftarrow & \{ \text{strengthening the domain} \} \\ & (\underline{A} w: w \in At \rightarrow \{F, T\} \wedge f.w: c.w.E0 = c.w.E1) \Rightarrow ev.E0 = ev.E1 \\ = & \{(7)\} \\ & E0 == E1 \Rightarrow ev.E0 = ev.E1 \\ = & \{ \text{Theorem 2} \} \\ & \text{true}. \end{aligned}$$

□

There is still more to be said about criterion (iii), since not all steps in a derivation consist in the application of a rule from propositional calculus. The remaining ones consist in replacement of a logical atom that has a defined value by that value. That expressions linked by such a step have the same value under ev is not trivial, since ev is not defined by induction on the structure of its argument. Nevertheless it is correct, as is shown by the following theorem.

THEOREM 3. For $E \in Ex$, $x \in At$ with $av.x \neq U$,

$$ev.E = ev.((x := av.x).E).$$

PROOF. It is clear from (0)-(4) that, for any $v \in At \rightarrow \{F, U, T\}$,

$$(9) \quad c.v.E = c.v.((x := v.x).E).$$

Now for $q \in \{F, T\}$,

$$\begin{aligned}
& \text{ev.E} = q \\
& = \{ (8) \} \\
& \text{E} == q \\
& = \{ (7) \} \\
& (\underline{A} \ w: w \in \text{At} \rightarrow \{F, T\} \wedge f.w: c.w.E = c.w.q) \\
& = \{ (9) \} \\
& (\underline{A} \ w: w \in \text{At} \rightarrow \{F, T\} \wedge f.w: c.w.((x := w.x).E) = c.w.q) \\
& = \{ (6), \text{av.x} \neq U \} \\
& (\underline{A} \ w: w \in \text{At} \rightarrow \{F, T\} \wedge f.w: c.w.((x := \text{av.x}).E) = c.w.q) \\
& = \{ (7), (8) \} \\
& \text{ev.}((x := \text{av.x}).E) = q \quad .
\end{aligned}$$

□

Finally, we deal with criteria (i) and (ii) simultaneously by showing that ev.E equals $c.\text{av.E}$ except perhaps when the latter value is U . This disposes of (i) since it follows from the definition of c that the exceptional case does not arise if all atoms x occurring in E satisfy $\text{av.x} \neq U$. It also disposes of (ii) since the definition of c shows that $c.\text{av.E} \neq U$ whenever E can be given a meaning by the introduction of conditional connectives.

THEOREM 4. For $E \in \text{Ex}$,

$$\text{ev.E} = c.\text{av.E} \vee c.\text{av.E} = U \quad .$$

PROOF. For $E \in \text{Ex}$ with $c.\text{av.E} \neq U$,

$$\begin{aligned}
& \text{ev.E} = c.\text{av.E} \\
& = \{ (8) \} \\
& \text{E} == c.\text{av.E} \\
& = \{ (7), (0) \} \\
& (\underline{A} \ w: w \in \text{At} \rightarrow \{F, T\} \wedge f.w: c.w.E = c.\text{av.E}) \quad .
\end{aligned}$$

Assume $w \in \text{At} \rightarrow \{F, T\}$ with $f.w$ fixed. We shall prove

$$(10) \quad c.w.E = c.\text{av.E} \vee c.\text{av.E} = U$$

by induction on the structure of E .

Case 0: $E \in \{F, T\}$. Then

$$\begin{aligned}
& c.w.E = c.\text{av.E} \vee c.\text{av.E} = U \\
& = \{ (0) \} \\
& E = E \vee E = U \\
& = \{ \} \\
& \text{true} \quad .
\end{aligned}$$

Case 1: $E \in \text{At}$. Then

$$\begin{aligned}
& c.w.E = c.\text{av.E} \vee c.\text{av.E} = U \\
& = \{ (1) \} \\
& w.E = \text{av.E} \vee \text{av.E} = U
\end{aligned}$$

$$= \{ f.w, (6) \}$$

$$\text{true}.$$

Case 2: $E = \text{non } E0$, with $(E:=E0).(10)$. Then

$$c.w.E = c.av.E \vee c.av.E = U$$

$$= \{ (2) \}$$

$$\sim c.w.E0 = \sim c.av.E0 \vee \sim c.av.E0 = U$$

$$= \{ \text{definition of } \sim \}$$

$$\sim c.w.E0 = \sim c.av.E0 \vee c.av.E0 = U$$

$$= \{ (E:=E0).(10) \}$$

$$\text{true}.$$

Case 3: $E = E0$ and $E1$, with $(E:=E0).(10)$ and $(E:=E1).(10)$. Then

$$c.w.E = c.av.E \vee c.av.E = U$$

$$= \{ (3) \}$$

$$c.w.E0 \sqcap c.w.E1 = c.av.E0 \sqcap c.av.E1 \vee c.av.E0 \sqcap c.av.E1 = U$$

$$= \{ \text{definition of } \sqcap, \text{ distribution} \}$$

$$(c.w.E0 \sqcap c.w.E1 = c.av.E0 \sqcap c.av.E1 \vee c.av.E0 = U \vee c.av.E1 = U)$$

$$\wedge (c.w.E0 \sqcap c.w.E1 = c.av.E0 \sqcap c.av.E1 \vee U \sqsubseteq c.av.E0)$$

$$\wedge (c.w.E0 \sqcap c.w.E1 = c.av.E0 \sqcap c.av.E1 \vee U \sqsubseteq c.av.E1)$$

$$= \{ (E:=E0).(10), (E:=E1).(10), \text{ complement rule} \}$$

$$(F \sqcap c.w.E1 = F \sqcap c.av.E1 \vee U \sqsubseteq c.av.E0)$$

$$\wedge (c.w.E0 \sqcap F = c.av.E0 \sqcap F \vee U \sqsubseteq c.av.E1)$$

$$= \{ F \sqcap q = F \text{ for all } q \}$$

$$\text{true}.$$

Case 4: $E = E0$ or $E1$, with $(E:=E0).(10)$ and $(E:=E1).(10)$. Consider $E' = \text{non}(\text{non } E0 \text{ and } \text{non } E1)$. As, for any p, q in $\{F, U, T\}$,

$$p \sqcup q = \sim(\sim p \sqcap \sim q) ,$$

it follows from the definition of c that for any $v \in At \rightarrow \{F, U, T\}$ we have $c.v.E = c.v.E'$, in particular for $v = w$ and for $v = av$. Hence this case reduces to a combination of cases 2 and 3.

□

Acknowledgement

The introduction of the ordering $\sqsubseteq$ was suggested to me by Jaap van der Woude.

References

- [0] R.L. Baber, The spine of software. John Wiley & Sons, Chichester, 1987.
- [1] E.W. Dijkstra, A discipline of programming. Prentice-Hall, Englewood Cliffs, 1976.
- [2] E.W. Dijkstra & W.H.J. Feijen, The linear search revisited. Struct. Prog. 10 (1989), 5-9.
- [3] D. Gries, The science of programming. Springer-Verlag, New York, 1981.
- [4] E.C.R. Hehner, L.E. Gupta, & A.J. Malton, Predicative methodology. Acta Inf. 23 (1986), 487-505.

September, 1989

A. Bijlsma
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513, 5600 MB Eindhoven, The Netherlands