THE LOGIC OF THE TERNARY SENTENTIAL CONNECTIVE "IF-THEN-ELSE"

William J. Rapaport

Department of Computer Science and Center for Cognitive Science
State University of New York at Buffalo, Buffalo, NY 14260

`rapaport@cs.buffalo.edu`
`http://www.cs.buffalo.edu/pub/WWW/faculty/rapaport/`

July 2, 1997

Abstract

This document was originally intended to be a section of Schagrin, Morton L.; Rapaport, William J.; & Dipert, Randall R. (1985), *Logic: A Computer Approach* (New York: McGraw-Hill).

This document discusses the ternary (i.e., 3-place) if-then-else *sentential connective*, which is based on the if-then-else *instruction* of computer programming languages.

Before we begin, however, we should look closely at an important difference between the use of if-then (and of if-then-else) in programming languages and its use in sentential (i.e., propositional) logic. In most logic texts (e.g., Shagrin et al. 1985) that are about the logic of declarative sentences—i.e., sentences, expressing propositions, that can be either true or false—the antecedent and consequent of a conditional sentence are themselves declarative sentences. But in an "imperative" programming language such as Basic or Pascal, an if-then statement is really a conditional instruction or command whose antecedent is a delcarative sentence but whose consequent is an instruction or command, i.e., an *imperative* sentence. Thus, whereas in sentential logic we are interested in conditional *sentences* such as:

> If Ann goes to the party, then Bob will go to the party

in many programming languages, we would be interested in conditional *instructions* such as:

```
IF a < b THEN print "yes"
```

Instructions, or commands, are neither true nor false. (It is possible to devise analogies to truth and falsity for commands, but we won't do that here. If you are interested, see Rescher 1966, Castañeda 1975.)

The if-then-else command is similar to the if-then command. In general, it has the form:

```
IF P, THEN X ELSE Y
```

where the antecedent, `P`, is some sentence that *is* either true or false (in the exclusive sense of "or"!) and where the "then-consequent" (`X`) and the "else-consequent" (`Y`) are instructions. The meaning of the if-then-else command is this:

If `P` is true, then do `X`, but if `P` is false, then do `Y`.

What we shall do here is study the logic of the if-then-else conditional *sentence*, where both consequents (as well as the antecedent) are sentences that are either true or false, e.g.:

If Ann goes to the party, then Bob will, else Cal will.

(Actually, such a sentence would more likely be expressed in ordinary English using the word 'otherwise' rather than 'else'.)

How would we decide whether such a sentence is true? We can compute its truth values by realizing that

if P then Q else R

is really a conjunction of two conditionals:

$(P \rightarrow Q) \wedge (\neg P \rightarrow R)$

So we can construct the following truth table:[1]

V(P)	V(Q)	V(R)	V($\neg P$)	V($P \rightarrow Q$)	V($\neg P \rightarrow R$)	V(if P then Q else R)
0	0	0	1	1	0	0
0	0	1	1	1	1	1
0	1	0	1	1	0	0
0	1	1	1	1	1	1
1	0	0	0	0	1	0
1	0	1	0	0	1	0
1	1	0	0	1	1	1
1	1	1	0	1	1	1

To see if this is a reasonable interpretation, we can try to find a sentence that expresses the result of performing a conditional command. The command:

```
IF P THEN X
```

seems to correspond to the sentence

if P, then `X` will be done.

[1]The notation 'V(P)' means: "the truth value of P"; '0' represents the truth value *false*; '1' represents the truth value *true*.

Similarly, the command:

```
IF P THEN X ELSE Y
```

seems to correspond to the sentence:

if P, then `X` will be done, else `Y` will be done.

The action is indicated by '`X`', but '`X` will be done' is a sentence. So we let:

Q = "`X` will be done"
R = "`Y` will be done"

where 'Q' and 'R' are sentences. Now, if you study our truth table carefully, you will see that it can be understood as saying the following:

if P is true, then at least Q (i.e., Q and possibly R)
else at least R (i.e., R and possibly Q)

You may or may not find this a plausible interpretation of the corresponding command:

```
IF P THEN X ELSE Y
```

which suggests to some people that if P is true, then `Y` will definitely not be done (as a result of this command), and, further, if P is false, then `X` will not be done (as a result of this command).

Of course, there is nothing wrong with the following program:

```
IF P THEN BEGIN X;Y END
     ELSE BEGIN Y;X END
```

But this appears to be a redundant set of instructions: If both `X` and `Y` are going to be done anyway, why bother testing if P is true? Well, one reason is that the programming-language "control structure" known as *sequencing* is not commutative: The sequence `X;Y` does not necessarily compute the same function as the sequence `Y;X`, since the first instruction might change the environment in such a way that the second instruction would not be using the same input as it would in the other case.

Thus, in both cases, it seems more reasonable to use the if-then-else *command* when you want to perform `X` or `Y` *but not both* (i.e., in the exclusive sense of "or").

This suggests the following, "stronger" interpretation of the sentential *connective* if-then-else:

if P then Q else R (in the "strong" sense)

means:

if P, then Q but not R, and if not P, then R but not Q.

NOTE.

After this document was written, I learned of the following reference that deals with the logic of if-then-else: Manna & Waldinger 1985: 12–13.

EXERCISES.

1. Symbolize "if P then Q else R" in the strong sense using $\rightarrow$, $\neg$, and $\wedge$.

2. Construct a truth table (using 0 and 1) for the strong sense of if-then-else. (If you do this correctly, you will see that V(if P then Q else R) $= 1$ in only two situations: when either $\mathrm{V}(P) = \mathrm{V}(Q) = 0$ and $\mathrm{V}(R) = 1$, or when $\mathrm{V}(P) = \mathrm{V}(Q) = 1$ and $\mathrm{V}(R) = 0$.)

3. (a) Construct an arithmetical truth-function for the *weak* sense of if-then-else (i.e., a function that arithmetically computes the truth value of an if-then-else sentence; see Schagrin et al. 1985 for more discussion of arithmetical truth functions).[2]
   (b) Construct an arithmetical truth-function for the *strong* sense of if-then-else.

4. (a) Construct a flowchart for computing the truth value of if-then-else in the *weak* sense.
   (b) Construct a flowchart for computing the truth value of if-then-else in the *strong* sense.

5. In programming languages, the sequence of instructions:

   ```
   IF P THEN X;
   Y
   ```

   means:

   if P is true, then do `X` and then do `Y`, else do `Y` but not `X`.

   What would a "weak" interpretation of 'if-then' be? What would a "strong" interpretation of 'if-then' be?

6. (a) Using the weak sense of if-then-else, determine the truth values of the following sentences with the indicated atomic truth-values.
   (b) Using the strong sense of if-then-else, determine the truth values of the following sentences with the indicated atomic truth-values.

V(A) = TRUE	V(C) = TRUE
V(B) = FALSE	V(D) = FALSE

   i. (if (A $\rightarrow$ D) then (A $\wedge$ $\neg$D) else D)
   ii. (if (A $\rightarrow$ D) then (A $\vee$ $\neg$D) else D)
   iii. (if (A $\rightarrow$ D) then A else D)
   iv. (if (A $\wedge$ D) then D else $\neg$A)
   v. (if (A $\vee$ D) then $\neg$A else D)
   vi. (if A then A else $\neg$A)
   vii. ((if A then B else B) $\leftrightarrow$ B)
   viii. ((if B then A else B) $\leftrightarrow$ A)
   ix. ((if $\neg$A then B else C) $\leftrightarrow$ (if A then C else B))

[2] E.g., the arithmetical truth function $\mathrm{FNEG}(\mathrm{V}(P)) = 1 - \mathrm{V}(P)$.

REFERENCES

1. Castañeda, Hector-Neri (1975), *Thinking and Doing: The Philosophical Foundations of Institutions* (Dordrecht: D. Reidel).

2. Manna, Zohar, & Waldinger, Richard (1985), *The Logical Basis for Computer Programming*, Vol. I: Deductive Reasoning (Reading, MA: Addison-Wesley).

3. Rescher, Nicholas (1966), *The Logic of Commands* (London: Routledge & Kegan Paul; New York: Dover).

4. Schagrin, Morton L.; Rapaport, William J.; & Dipert, Randall R. (1985), *Logic: A Computer Approach* (New York: McGraw-Hill).