A Final Solution to the Dangling else of ALGOL 60 and Related Languages

PAUL W. ABRAHAMS
Information International, Inc., New York, N. Y.

Abstract

The dangling **else** problem consists of a class of potential ambiguities in ALGOL-like conditional statements whose basic form is "**if** $B1$ **then if** $B2$ **then** $S1$ **else** $S2$" where $B1$ and $B2$ are Boolean expressions and $S1$ and $S2$ are basic statements. The difficulty lies in whether to attach the **else** to the first **if** or to the second one. Existing solutions to the problem are either ambiguous or unnecessarily restrictive. Let S and $S1$ be statements. We define S to be *closed* if "S **else** $S1$" is not a statement, and to be *open* if "S **else** $S1$" is a statement. Thus an unconditional statement is a closed statement. Open and closed conditional statements are defined by syntax equations in such a way as to preserve openness and closure. In each case, an **else** must always be preceded by a closed statement. It is shown that the syntax equations are unambiguous, and that any change in the statement types required within the syntax equations would lead to either ambiguity or unnecessary restriction.

1. Introduction

The problem of the dangling **else** arises in any programming language that uses ALGOL-like conditional statements. The problem may be illustrated as follows: Let $B1$ and $B2$ be Boolean expressions and let $S1$ and $S2$ be statements that are sufficiently elementary so that their internal structure is not involved in the problem. Then the sequence

if $B1$ **then if** $B2$ **then** $S1$ **else** $S2$ (1)

can be interpreted in two ways, viz.,

if $B1$ **then begin if** $B2$ **then** $S1$ **else** $S2$ **end** (2)

and

if $B1$ **then begin if** $B2$ **then** $S1$ **end else** $S2$ (3)

In other words, there is an ambiguity as to whether the **else** belongs with the first **if** or with the second one. This report is intended to lay the dangling **else** to rest once and for all by presenting a necessary and sufficient resolution of the **else** ambiguities, and explaining the concepts involved in sufficient detail so that the reader understands not only what the resolution is but also why it works, and why other solutions do not work. This resolution is both simpler and less restrictive than the ones presently used; removing the unnecessary restrictions simplifies matters both for the programmer and for the compiler writer.

The dangling **else** problem first came to prominence in the computing community when ambiguities were noticed in the conditional statement described in the original ALGOL 60 report [8]. These ambiguities were both syntactic and semantic. The principle involved is illustrated in (1) although this particular construction was excluded and the only cases in which ambiguity arose were more complicated. Revised ALGOL 60 [7] excluded the ambiguous cases, but threw out certain unambiguous cases also. The paper by Kaupe [6] should be consulted for illustrations of the problem.

In existing programming languages, several approaches have been taken to resolve the ambiguity:

1. Construct a complicated set of syntax equations that excludes (1). This approach is taken in Revised ALGOL 60 [7].
2. Require that every **if** be accompanied by an **else.** This approach is taken in EULER [9]. A modification to ALGOL 60 in a similar spirit was suggested by Burkhardt [2], who proposed that unpaired **if**'s be replaced by the distinctive symbol **test.**
3. State verbally that each **else** is to be paired with the innermost unpaired **if,** but leave the ambiguity in the syntax equations. This approach is taken in PL/I [5]. The same effect is achieved in COBOL [3] by first requiring all **if**'s to be paired with **else**'s, and then permitting vacuous **else**'s (i.e., those followed by "NEXT SENTENCE") to be dropped when they appear at the end of a sentence.

The approach taken here will be to construct a set of syntax equations that provides an unambiguous analysis of every well-formed conditional expression. The ideas here are based on the article by Kaupe [6]. Unfortunately the solution presented there is incorrect, as will be shown later; furthermore, it does not clarify the concepts underlying the problem. The solution given here, though independently discovered, is the same as one mentioned by Floyd [4, p. 332]. However, Floyd's solution is not generally known, and his article does not provide any commentary on the matter.

2. Proposed Solution

The solution here is based on the concept of a closed statement vs. an open statement. Let S be a statement. Then S is *closed* if "S **else** $S1$" is not a statement for any statement $S1$, and S is *open* if "S **else** $S1$" is a statement for some $S1$. Thus, for example,

if $X<Y$ **then go to** A

is an open statement and

if $X<Y$ **then go to** A **else go to** B

is a closed statement.

The syntax equations for statements are:

⟨unconditional statement⟩ ::= ⟨basic statement⟩|
⟨compound statement⟩ | ⟨block⟩ |
⟨for clause⟩⟨unconditional statement⟩ |
⟨label⟩:⟨unconditional statement⟩

⟨if clause⟩ ::= **if** ⟨Boolean expression⟩ **then**
⟨closed conditional statement⟩ ::=
⟨if clause⟩ ⟨closed statement⟩ **else** ⟨closed statement⟩ |
⟨for clause⟩ ⟨closed conditional statement⟩ |
⟨label⟩:⟨closed conditional statement⟩
⟨open conditional statement⟩ ::= ⟨if clause⟩ ⟨statement⟩ |
⟨if clause⟩ ⟨closed statement⟩ **else** ⟨open statement⟩ |
⟨for clause⟩ ⟨open conditional statement⟩ |
⟨label⟩:⟨open conditional statement⟩
⟨closed statement⟩ ::= ⟨unconditional statement⟩ |
⟨closed conditional statement⟩
⟨open statement⟩ ::= ⟨open conditional statement⟩
⟨conditional statement⟩ ::= ⟨open conditional statement⟩ |
⟨closed conditional statement⟩
⟨statement⟩ ::= ⟨unconditional statement⟩ |
⟨conditional statement⟩

The undefined terms in these syntax equations have the meanings assigned to them in ALGOL 60.

It is easily shown that closed and open statements as defined by this syntax have the properties of closure and openness as defined earlier. First, observe that a closed statement has exactly the same number of **if**'s and **else**'s (excluding any that are enclosed between brackets of one sort or another). This property can be verified by noting that it is true of unconditional statements and that the definition of a closed conditional statement preserves the property. Since a statement cannot have more **else**'s than **if**'s, it follows from the preceding observation that the introduction of an additional **else** at the end of a closed statement would produce a nonstatement. Hence closed statements have the closure property. In a similar way it can be shown that open statements have the openness property.

3. Unambiguousness of the Syntax

In order to show that the syntax equations are unambiguous, we must show, first of all, that for each equation

$$\alpha_0 ::= \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$$

the sets of strings described by α_1, α_2, $\cdots$, α_n are disjoint. We must also show that each alternative α_i of the equation

$$\alpha_i = \beta_1 \beta_2 \cdots \beta_k$$

(where each β_i is a basic symbol or a syntactic category) has a unique decomposition; i.e., if any string A is of the form α_i, then there is only one way of partitioning A into substrings a_1, a_2, $\cdots$, a_k such that each a_j is of the form β_j. These conditions are easily seen to be necessary and sufficient for unambiguity.

To show the disjointness of the alternatives in the different syntax equations, we note that two alternatives are disjoint if they begin with syntactic entities that have no initial characters in common (e.g., a for clause or a label). We also note that two alternatives are disjoint if one of them is open and the other is closed. These two observations are sufficient to handle all cases except the first two alternatives in the definition of an open conditional statement. These two alternatives can be separated out by noticing that the sequence

⟨closed statement⟩ **else** ⟨open statement⟩

is, by the definition of closure, not a statement.

The uniqueness of decomposition is also easily shown. First observe that an **if** clause is unambiguously delimited by **if** and **then,** since occurrences of **if** and **then** are always matched. Most of the remaining cases are straightforward. The sequences of the form

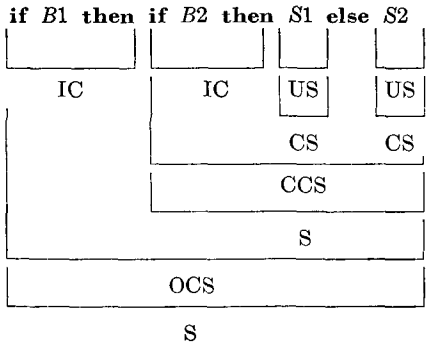
⟨closed statement⟩ **else** ⟨statement⟩

which appear in the definitions of closed conditional statements and open conditional statements, are handled by noting that the closed statement must have the same number of **if**'s and **else**'s, so that the **else** that separates the two statements in the decomposition can be found by counting from left to right until the number of **else**'s exceeds the number of **if**'s.

If in the definition of an open conditional statement we had permitted an open statement to precede **else** in the second alternative, then the first and second alternatives would no longer be disjoint, by the definition of openness. If in the definition of a closed conditional statement we had permitted an open statement to precede **else** in the first alternative, then closure would no longer be preserved and the disjointness property would again be lost. Intuitively, the difficulty arises because an open statement can under certain circumstances absorb an **else** that follows it. We therefore see that it is not only sufficient but also necessary that the statement preceding **else** be closed if ambiguity is to be avoided. It is precisely the failure to distinguish between closed and open statements that accounts for the ambiguities of the original ALGOL 60 conditional statement and for the unnecessary restrictions in later versions of the conditional statement.

4. Illustration and Commentary

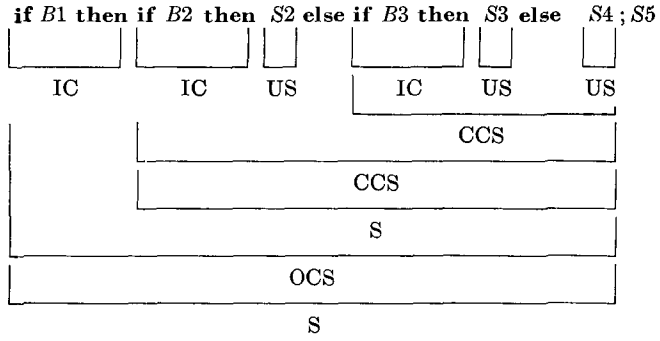
To illustrate these definitions, we first note that the statement (1) has the unambiguous parse



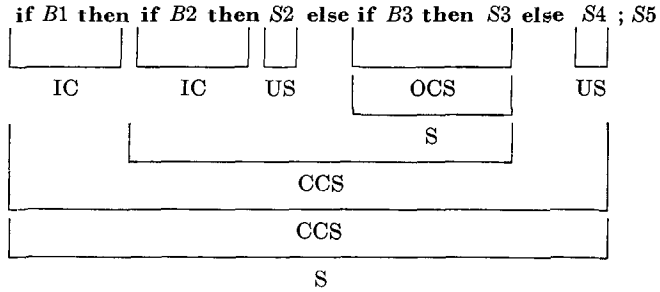
using obvious abbreviations, and this corresponds to the interpretation (2). For a more complicated example, we let $B1$, $B2$ and $B3$ be Boolean expressions and let $S1$, $S2$, $S3$, $S4$ and $S5$ be basic statements. Then the statement

if $B1$ **then if** $B2$ **then** $S2$ **else if** $B3$ **then** $S3$ **else** $S4$; $S5$

can be parsed as follows:



With the definitions given in [6], this statement has the additional parse:



A glance at the results when $B1$ is false shows that the ambiguity is semantic as well as syntactic. The trouble arises because the definition of a closed conditional statement in [6] permits an open conditional statement to follow an **else**. The result is that the definition of a closed conditional statement does not preserve closure.

In the syntax of conditional expressions, **if**'s and **else**'s play the role of left parentheses and right parentheses respectively. A glance at the syntax equations shows that each **else** is paired with the innermost possible **if** even in an open conditional statement. Furthermore, an open conditional statement can be converted to a closed conditional statement by adding one **else** with a dummy statement at the end of the open conditional statement for each unpaired **if.** This closed conditional statement is semantically equivalent to the open conditional statement from which it was obtained, as one can see by induction on the number of unpaired **if**'s. The inserted **else**'s are just like implicit right parentheses. The syntax equations imply a particular resolution of the ambiguities that would exist if we did not distinguish between open and closed statements; any resolution other than the standard one can be obtained through the inclusion of explicit **else**'s with dummy statements at positions other than the end of the entire statement.

A conditional statement is well-formed if and only if the number of **else**'s never exceeds the number of **if**'s as we scan from the left, provided the statement is not malformed because of extraneous considerations (e.g., a non-statement between **then** and **else**). Every well-formed conditional statement has an unambiguous syntactic analysis, and hence is semantically unambiguous. Furthermore, if we leave considerations of ambiguity aside, there is no obvious way to assign a meaning to statements that do not satisfy this well-formedness condition; thus the well-formedness condition is not overly restrictive.

5. Generalizations

Although the syntax of conditional expressions that we have given here is unambiguous, it is also slightly redundant. There are several possible variants which are not syntactically correct according to our equations, but which are quite convenient to use and can be defined unambiguously by syntax equations. For instance, the form

if $B1$ **then** $S1$ **if** $B2$ **then** $S2$

is permitted in JOVIAL [10], and is equivalent to:

if $B1$ **then** $S1$ **else if** $B2$ **then** $S2$ (4)

In other words, an **else** can be omitted when it immediately precedes an **if** (as is often the case). Another possible modification is to allow

if $B1$ **then** $S1$; $S2$ **else** $S3$

when $S1$ and $S2$ are closed, which is equivalent to

if $B1$ **then begin** $S1$; $S2$ **end else** $S3$.

In other words, **begin** and **end** can be omitted when surrounded by **then** and **else** provided that they enclose a list of closed statements. The COBOL treatment of conditional expressions is along these lines. This idea would avoid the common error of putting a semicolon before the delimiter **else** in ALGOL programs.

The MAD language [1] has an interesting treatment of this question, which essentially allows the omission of **begin** and **end.** The MAD statement

WHENEVER $B1$; $S1$; $S2$; OR WHENEVER $B2$; $S3$; $S4$; OTHERWISE; $S5$; $S6$; END OF CONDITIONAL;

is equivalent to the ALGOL statement

if $B1$ **then begin** $S1$; $S2$ **end else if** $B2$ **then begin** $S3$; $S4$ **end else begin** $S5$; $S6$ **end**

Here we have used the semicolon in place of a card boundary. END OF CONDITIONAL matches the implicit **begin** created by OTHERWISE. The restricted form

WHENEVER $B1$, $S1$;

requires that $S1$ be an unconditional statement. This restriction could be avoided through the approach that we have described here.

Generally speaking, these modifications trade gains in ease of programming for losses in language complexity. For instance, the form (4) may have arbitrarily many statements preceding the **else.** To tell whether control should or should not pass to $S2$ in the case where $B1$ is false, the translator must scan through the statements

following $S1$ until either the end of the block or an unmatched **else** is found. In more deeply nested conditionals, the task of the translator is correspondingly more difficult.

6. Conclusion

The equations presented here embody a straightforward conception of the notion of a conditional statement. They can easily be applied to conditional expressions, as is done (correctly) in [6]. They permit unpaired **if**'s without ambiguity, and do not rely on informal remarks in order to avoid ambiguity. For that reason they can be used as input to a syntax-directed compiler. If the equations are to be used in ALGOL 60 then they can be inserted as they stand; else they can be adapted to the language at hand.

Acknowledgment. I wish to thank the editor of this *Communications* department for his excellent suggestions regarding the revisions of this paper, particularly in connection with the treatment of conditional statements in COBOL and with the formalization of the necessary and sufficient conditions for unambiguity in a context-free language.

RECEIVED APRIL, 1966; REVISED MAY, 1966

REFERENCES

1. ARDEN, B., GALLER, B., AND GRAHAM, R. *The MAD Manual.* U. of Michigan Press, Ann Arbor, Mich., 1965.
2. BURKHARDT, W. Syntax and generalization of ALGOL 60. (Letter to the editor) *Comm. ACM 8* (May 1965), 261.
3. COBOL—1961: revised specifications for a common business oriented language. Publ. 0-598941, U.S. Government Printing Office, Washington, D.C., 1961.
4. FLOYD, R. Syntactic analysis and operator precedence. *J ACM 10*, 3 (July 1963), 316–333.
5. PL/I: Language specifications. Form C26-6571-2, IBM Programming Systems Publications, Poughkeepsie, N.Y., Jan. 1966.
6. KAUPE, A. A note on the dangling **else** in ALGOL 60. *Comm. ACM 6* (Aug. 1963), 460–462.
7. NAUR, P., AND WOODGER, M. (Eds.) Revised report on the algorithmic language ALGOL 60. *Comm. ACM 6* (Jan. 1963), 1–17.
8. NAUR, P. (Ed.) Report on the algorithmic language ALGOL 60. *Comm. ACM 3* (May 1960), 299–314.
9. WIRTH, N., AND WEBER, H. EULER: A generalization of ALGOL, and its formal definition: Pt. II. *Comm. ACM 9* (Feb. 1966), 89–99.
10. SHAW, C. A specification of JOVIAL. *Comm. ACM 6*, 12 (Dec. 1963), 721–736.

Addendum To

"A Nonrecursive Method of Syntax Specification" by John W. Carr III and Jerome Weiland, *Comm. ACM 9*, 4 (April, 1966), 267–269:

The research behind this paper was made possible by the joint support of the National Science Foundation (NSF-GP-1476) and the Army Research Office (Durham) (DA-31-124-ARO(D)-98); and by the Atlantic Refining Company.