to work out system needs and standards. As far as codes were concerned there was need for some rationalization; that was difficult but there were now hopeful signs of some successful thinking and collaboration in this field.

Mr. Stringer concluded by saying that many of the things he had said he had said with his tongue in his cheek, and it had had the desired effect of encouraging discussion. This he found most gratifying, since he considered that the discussion was more important than what he had said in his paper.

The Chairman said he was sure it would be agreed that Mr. Stringer's provocative address was the note on which to close the conference. He would like to say "Thank you" for coming along in such numbers and making the Conference such a success.

The Conference then terminated.

Correspondence

To the Editor,
The Computer Journal.

Sir,

I was very interested in the recent articles of Brooker and Morris (1960, 1961) on an "Assembly Program for a Phase Structure Language." Their scheme is very general, but it suffers from the disadvantage that it is not possible to use statements as parameters in statement definitions—only one level and not a hierarchy of statements is allowed. This restriction could be serious if their program were used to translate a language such as ALGOL (Woodger, 1960).

In their articles, Brooker and Morris distinguish carefully between *phrases* and *statements*. The format or *syntax* of each is defined, but the meaning or *semantics* of a phrase is defined only when this occurs as part of a statement.

If the distinction between phrases and statements is removed (the term *class* being used to include both concepts), a simpler and more powerful scheme can be developed. We construct a dictionary of class identifiers giving the syntactic definition of the associated classes. Some of these classes will also be defined semantically and then the dictionary will contain a reference to this definition also. This semantic definition corresponds to the statement definition of Brooker and Morris. It will be obeyed interpretively by the translation routine to produce the compiled program. One can now easily allow any class to appear as parameter in the syntactic and semantic definitions. The definition of general types of statement (such as the ALGOL conditional statement) is now much more convenient.

The analysis record constructed by the expression recognition routine will now include, at the end of the record of each subclass that is defined semantically, a reference to this semantic definition. This will be used later by the translation routine.

As an example I give the definitions of [GE] which would result in instructions being compiled, which when obeyed would set the accumulator A equal to the current value of [GE].

Syntactic definition : [GE] = [$\pm$?]T, [GE] $\pm$ T
Semantic definition : [GE]
$\rightarrow$ 1 if [GE] $\equiv$ [GE/1] $\pm$ T
Let [GE] $\equiv$ [$\pm$?]T
A = [$\pm$?]T
END
1] [GE/1]
A = A $\pm$ T
END

As syntactic definition of a conditional statement we might take:

Syntactic definition : [Unconditional statement] = [jump N], [Y = [GE]]
Syntactic definition : [Conditional statement] = if [GE/1] ϕ [GE/2] then [unconditional statement]

which includes both the forms:

If $y > 0$ then jump N

and

If $a^2 + b^2 > a/b$ then $x = a^2 + b^2$.

The semantic definition is easy to construct once elementary conditional jumps have been defined. It would take the form

semantic definition : [conditional statement]
Let [conditional statement] $\equiv$ if [GE/1] ϕ [GE/2] then [unconditional statement]
ts1 = [GE/1]
ts2 = [GE/2]
jump α_3 unless ts1 ϕ ts2
[unconditional statement]
$(\alpha_1 + \alpha_3) = \alpha_2$
END.
(here α_3 contains the number of an unused label).

The scheme suggested here seems to be a little simpler logically, as well as being more general, than that of Brooker and Morris, but no account has been taken of the efficiency of translation. On a particular machine it might be found that one method can be programmed much more efficiently than the other, and that method would then be chosen.

J. M. Watt.

Computer Laboratory,
The University,
Liverpool 3.
3 March 1961

(*See* p. 176 *for authors' reply.*)

References

BROOKER, R. A., and MORRIS, D. (1960). "An Assembly Program for a Phrase Structure Language," *The Computer Journal*, Vol. 3, p. 168.

BROOKER, R. A., and MORRIS, D. (1961). "Some Proposals for the Realization of a Certain Assembly Program," *The Computer Journal*, Vol. 3, p. 220.

WOODGER, M. (1960). "An Introduction to ALGOL 60," *The Computer Journal*, Vol. 3, p. 67.