REFERENCES

1. Ritchie, R.W. Classes of predictably computable functions, Trans. AMS, 106, 1965, 1027-1044.

2. Grzegorczyk, A. Some classes of recursive functions, Rozprawy Matematyczne, 4, 1953, 1-45.

3. Jones, N. Preliminary report: reducibility among combinatorial problems in logn space, 7th Princeton Conf. on Information Sciences and Systems, 1973.

4. Stockmeyer, L. and A.R. Meyer. Word problems requiring exponential time, 5th ACM Symp. on Theory of Computing, 1973, 1-9.

5. Cook, S.A. The complexity of theorem-proving procedures, 3rd ACM Symp. on Theory of Computing, 1971, 151-158.

6. Karp, R.M. Reducibility among combinatorial problems, in Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, eds., Plenum Press, N.Y. 1972, 85-104.

7. Lind, J. Computing in logarithmic space, Bachelor's Thesis, Department of Electrical Engineering, M.I.T., 1973, 72 pp. (to appear as a Project MAC Technical Report, Summer, 1973).

****

SOME FURTHER COMMENTS ABOUT NOTHING

G. Ricci*

1. Foreword. In Formal Language Theory it is necessary to distinguish the empty word from the empty set of words, namely $\lambda \neq \emptyset$. It is suggested in [3] that this necessity could arise only from notational oversimplification, e.g. from the confusion between the concatenation of words and the concatenation of sets of words. So if we avoid such confusions, we do not need any distinction between λ and $\emptyset$.

This point was treated in [3] by a simple definition of the notion of word. A word was considered as a function in the set-theoretical sense. This implies $\lambda = \emptyset$, i.e. a blasphemy against the well respected principle $\lambda \neq \emptyset$. Thus, in order to avoid this conflict several methods were introduced, but some of them were not simple.

In [1] there is a different approach. A word is considered as a triple (A,f,B) where f is a function, A is its domain and B is its codomain. According to this approach, an empty word λ becomes a triple $(\emptyset,\emptyset,B)$, that being clearly different from $\emptyset$.

* Dept. of A.A. & C.S., University of Waterloo (Ontario) and Instituto di Matematica, Università di parma (Italy)

D. B. Benson introduced this approach by recalling the algebraic practice of specifying the codomains in order to distinguish the onto functions from the into functions. I accept this solution (as well as the other several possible solutions), but I think that some points deserve further discussion.

2. Remarks and questions. (a) Benson's solution makes $\lambda \neq \emptyset$, but it avoids the problem of deciding whether this distinction is logically necessary for the developments of formal language theory.

(b) The first of the solutions suggested in [3] n. 5 was clearly complicated. Yet this was a solution only for the (additional) problem of paying some respect to the convention $\lambda \neq \emptyset$. On the contrary, the starting approach of n. 3 was much simpler. It defined a word simply as a function f, which is simpler than a triple (A,f,B).

(c) The second solution suggested in [3] n. 5 redefined λ as the set containing the empty word $\lambda = \{\emptyset\}$. It is also quite simple.

(d) A triple (A,f,B) differs from a triple (A,f,B') whenever $B' \neq B$. Hence any word defined in Benson's manner is different if considered in different alphabets. This implies an infinite proliferation of empty words, reminding of the proliferations of empty sets in some set theoretic treatments [2].

(e) It is known that we pass from words to labeled trees by making minor changes on the definition of the domain of our functions [4]. Therefore a triple $(\emptyset,\emptyset,B)$ is always the same thing whether considered as an empty word or as an empty tree. I would like the identification of the two, but how can we compare it with the proliferation mentioned in (d)?

(f) I agree with Benson that, in some fields of algebra, it might be convenient to consider triples rather than functions. But does this convenience exist in our case? If it does, it should be for stronger reasons than for making $\lambda \neq \emptyset$.

3. Conclusion. In my opinion the choice between $\lambda = \emptyset$ and $\lambda \neq \emptyset$ is a matter of personal preference. I prefer the heretical solution, $\lambda = \emptyset$, which I think is more economical. However I thank all the people who disagree explicitely with me. Indeed this strange paradox could be useful. I suspect that at the present stage of computer science there are a lot of such things to discuss.

REFERENCES

1. D.B.Benson, Another Comment on the Null Word Definition Problem, SIGACT News, 21, January 1973

2. W.S.Hatcher, Foundations of Mathematics, W.B.Saunders Company, Philadelphia, 1968.

3. G. Ricci, An observation on a Formal Language notation, SIGACT News, 17, October 1972.

4. J.W. Thatcher, Characterizing Derivation Trees of Context-Free Grammars through a Generalization of Finite Automata Theory, J.C.S.S. 1 (1967).