Does LISP differ from ALGOL essentially ?

Friedemann Simon

Institut für Informatik
Universität Kiel
Olshausenstr. 4o - 6o
D-23 Kiel

A fare spread opinion is that LISP and ALGOL belong to different "families" of programming languages. In our current activities concerning LISP, we are trying to characterise pure LISP as an ALGOL-like programming language in the sense of ALGOL 6o resp. ALGOL 68 /1/, /2/. LISP is considered as a sublanguage of ALGOL 6o, where the datatype "s-expression" with its 5 standard functions is introduced and where procedure identifiers are allowed as values of function-procedures (ALGOL 68) in order to have upward FUNARGs /3/. In contrast to the operational semantic definitions via interpreters, this approach gives a precise, mathematical definition of the LISP-semantic. Within this framework we are able to prove properties of LISP-programs much easier than using inductive proofs based on an interpreter. Our method for modelling variable bindings follows the well known ALGOL 6o definitions, which are very close to the FUNARG-feature of LISP 1.5, while other authors prefer the "shallow access binding" method; e. g. M.J. Gordon gives a formal definition of pure LISP by algebraic methods /4/.

LISP- and ALGOL 6o implementations have an important difference concerning their run-time storage management. ALGOL 6o only needs a stack (deletion strategie), while LISP requires a heap (retention strategie) in order to keep variable bindings and data i. e. s-expressions. Starting from a proof scetch given by M.J. Fischer /5/ it has been shown that every LISP-program π can be transformed in a functional equivalent program π', which is deletion-tolerant, and where the operator cons is only used to construct a non-atomic final result of π' from its atoms /1/.

A simple consequence of the proof is that cons-free LISP with FUNARGs (c. f. LISP 1.5) is universal. If we restrict ourselves to cons-free LISP without FUNARGs, it can be shown using the theory of stack automata that this sublanguage is not universal /6/. The universality of LISP is given either by cons (neccessary to implement an interpreter) or in cons-free LISP by downward FUNARGs (procedure calls as in ALGOL 6o); upwards FUNARGs are irrelevant for the universality.

/1/ Simon,F.:Zur Charakterisierung von LISP als ALGOL-ähnliche Programmiersprache mit einem strikt nach dem Kellerprinzip arbeitenden Laufzeitsystem. Bericht 2/78, Institut für Informatik und praktische Mathematik der Universität Kiel, 1978

/2/ Lippe,W.M.,Simon,F.:A mathematical semantic definition for LISP. to appear

/3/ Sandewall,E.:A Proposed Solution to the FUNARG Problem. Uppsala Univ.-Dept. of Comp. Sc., Rep. 29, 197o

/4/ Gordon,M.J.C.:Models of Pure LISP (a worked example in semantics). Dept. of Machine Intelligence, Univ. of Edinburgh, Rep.31, 1973

/5/ Fischer,M.J.:Lambda Calculus Schemata. SIGPLAN Notices 7(1),1972

/6/ Simon,F.,Trademann,P.:Eine Beziehung zwischen cons-freiem LISP und Stackautomaten.Elektronische Informationsverarbeitung und Kybernetik EIK 14, Nr.12, 1978