

LISP HISTORY

Herbert Stoyan
University of Dresden
Togliattistr. 40
806 Dresden
DDR

For the SIGPLAN conference on history of programming languages held in Los Angeles in this June, J. McCarthy had to write a paper about LISP-history (1). He was very able to do this because he has given a talk on LISP history in summer 1974 at M.I.T. (2) and has contributed since then a lot of remarks and comments to my work on compiling a complete history of our language. His paper corresponds to the state of our knowledge in May of this year (1978) before D. Park found the original LISP 1 manual (3).

Using this material I can now give a better description of the old matters. But there are open questions too - the informed reader is kindly asked to help in answering them.

a) PREHISTORY

J. McCarthy was born in 1927 and studied Mathematics. In 1948 he became BS at the CalTech and in 1951 he graduated with a PhD on differential equations from Princeton University. His interest in problems of A.I. came from an accidental presence at a symposium on cybernetics held at CalTech in 1949. Here he heard J. Von Neumann and other pioneers in this field. Since then he worked among other things in this field and wrote a paper on the inverting of Turing-machines to summarize some of his thought. The paper was published later 1956 in the collection "Automata Studies" (4).

In 1952 C. Shannon invited J. McCarthy and M. Minsky to work in the Bell Telephone Labs where they worked together during that summer. Topics of discussion were questions of cybernetics and automata theory. In the time until 1956 McCarthy, Shannon, Minsky and some other people came to form the opinion that A.I. could be reached by using digital computers rather than by cybernetic vehicles.

When McCarthy became Assistant Professor at Dartmouth College, he organized a summer school on cybernetics and artificial intelligence (5). He asked the Rockefeller Foundation for money, writing a proposal that would, as he found in 1977, make sense today, too.

A lot of researchers came to visit the conference, ten were more or less present the entire time. Very important was the discussion with A. Newell and H. Simon. Both had some experience with the use of

electronic computers at this time (at the JOHNNIAC at RAND corporation) and were developing their "logic theory machine" (6). For this work they proposed a language for formulating the logical means in their system - the "logical language" LL. Key idea of this language was the concept of lists.

IPL (information processing language), as they called the language later, was more an assembly language for artificial intelligence and J. McCarthy didn't like it very much. He hoped to have, in a language of this kind, algebraic expressions as he probably saw in FORTRAN, which was, 2 years after its announcement (7), running in the spring of 1956 (8).

b) EARLY HISTORY

But it is clear and well known which influence IPL had on the further development of AI and of programming languages (9, 10). Now it was clear for McCarthy: his language must be of FORTRAN style but doing list-processing as IPL.

He had time to develop his ideas in connection with the project of the geometry theorem prover, developed under the direction of H. Gelernter at IBM. Also participating were G. Gerbrich and J. Hanson.

Due to the proposal of IBM to build up the "New England Computation Center" for the universities and colleges in this field, McCarthy concentrated on the IBM-704. He developed in Dartmouth the basic functions for a list processing system embedded in FORTRAN.

It is very well known that the word structure of the IBM-704 (the same as of IBM-7090) gave the names for LISP's basic functions. 1957 there were 4 of them: CPR (prefix), CDR (decrement), CTR (tag) and CAR (address part). The function CWR augmented this set in giving the contents of a word. In the beginning, all functions had to be used in connection with CWR to give the same result as what we do today with CAR etc. Then an operator CONS of four arguments was invented which served to fill a word with the 4 parts.

The use of the functions came to show a better definition of the basic functions, which allowed CWR to be dropped, and also showed the practical uselessness of the prefix and tag parts (only 3-bit quantities). Gelernter and Gerberich made CONS into a function having the address of the filled word as a result.

A clear shadow of this can be studied in the JACM-paper on the geometry theorem prover (11). Later the team developed FLPL independently of J. McCarthy. The exact dates for this cooperation are not known. The results of the geometry prover project of IBM were published in (12, 13,

14), the first proof could be stated in spring of 1959. During 1958 McCarthy had a subsidy of the A. Sloan foundation and he spent the year at MIT Cambridge thinking about his language and about paper-programming a chess program in FORTRAN. In doing this he became aware that the means of formulating the expression of conditional actions were very poor. It is well known that FORTRAN has only the so called "arithmetical" IF, which allows a test against 0 and 3 jumps to different statement numbers and the "logical" IF-statement, allowing a test and a statement performed if the test is true. But FORTRAN has both forms of IF as statements. (Later the IF-THEN-ELSE came, but it is a statement too). In this chess program McCarthy found it very useful to have a conditional expression. He defined a function IF with 3 arguments to do the work, but it is clear that all arguments are computed in FORTRAN and the selection could be done only afterwards. Here was a restriction in FORTRAN and McCarthy thought about it. The result of this was a paper send to Communications of ACM and a lot of propaganda for conditional expressions to come into ALGOL. But the idea was too new and the Communications published the paper in the form of a letter to the editor (15) and the ALGOL committee did not use the proposal.

During the summer of 1958 McCarthy spent some time at IBM (it is not clear in which connection to the FPLP-group) and worked to write programs for symbolic differentiation. Doing this he was forced to write recursive programs and this concept was added to the conditional expressions. Another concept which was used in the (paper-) differentiation program was the use of functional arguments. They were very useful for differentiating sums with 3 and more summands. For writing functional argument McCarthy adopted the way of the Lambda-Calculus (16). (He wrote me that he didn't read the Church paper completely and didn't understand it fully but I guess this was more out of humility).

In this way a lot of basic concept for a language beyond FORTRAN arose and it was very nice for McCarthy to find a medium to discuss and formulate it's final version: in the autumn of 1958 the MIT AI-project was founded.

McCarthy became Assistant Professor of Communication Science (in the Electrical Engineering Department) and Minsky became Assistant Professor (in the Mathematics Department). "The project was supported by the MIT Research Laboratory of Electronics (RLE) ... No written proposal was ever made. When J. Wiesner (the director of RLE) asked Minsky and me what we needed for the project, we asked for a room, two programmers, a secretary and a keypunch; he also asked us to undertake the supervision of some of six mathematics graduate students that RLE had undertaken to support ... " (1)

N. Rochester from IBM was at this time visiting at MIT and participated in the work, and C.E. Shannon was strongly connected with the group. The students were P.W. Abrahams, D.G. Bobrow, K.R. Brayton, L. Hodes, L. Kleinrock, D.C. Luckham and K. Maling; the programmers were S.R. Russell and D.J. Edwards (the latest is mentioned only after May 1959).

During September and October McCarthy wrote some memos about the new language. At the same time he worked on the advice taker proposal (17). Then he started to write a paper concerning the formalism of conditional expressions and its impact for recursive function theory. Doing this he developed the universal function Apply. At the same time the students and programmers had to hand-translate some of the older paper-programs of J. McCarthy and some new ones (for reading and printing symbolic

expressions). Doing this the conventions for subroutine entry and exit, for work with the stack and for storage management, were developed. We must remember that none of the members of the AI group had a deeper knowledge of compilers or language implementation!

By writing the read and print programs the syntax of S-expressions was fixed. The available IBM026 key punch has only 47 characters and because of this way the syntax was rather poor. For organizing the work with recursive functions, at the beginning only the IPL way of doing this was known. McCarthy proposed to use, instead of the pushdown lists (locally to each function), linear stacks. This was abandoned soon in favor of a public stack. The work of subroutines was organized in such a way that, at entry time, they saved their local registers in the stack to do the work, restoring them at the end. It is clear that this work was completely new at this time (remember that Dijkstra's paper (18) on implementation of recursive ALGOL using a stack was written in 1960!). More complicated was the problem of managing the list storage. The IPL solution (let the programmer do the work) seemed not to be the goal and the counter-method developed at the same time by Collins (19) was not very well applicable. (The free places are only 6 bits and these are not directly linked together). The storage was large and the problems small and so they shifted the storage manipulation to later times. In transferring the paper programs to assembly programs as parts of the growing program system Steve Russell became soon very experienced. One nice day he saw the APPLY-EVAL-functions on McCarthy's desk and he asked if he should transfer them too. But McCarthy said that this is only theory and not practice. Russel didn't bother about it, took the functions and added them to the system.

This important event most likely happened in November of 1958, then D. Park, who joined the group at this time, claims he would come if these functions were running.

By April 1959 the status was as following (20):

(a)The source language has been developed and is described in several memos from the AI group.

(b)20 useful subroutines have been programmed in LISP, hand-translated into SAP and checked out on the IBM 704. These include routines for reading and printing list structures.

(c)A routine for differentiating elementary functions has been written. A simple version has been checked out etc...

(d)A universal function APPLY has been written in LISP, hand-translated, and checked out. Given a symbolic expression for a LISP function and a list of arguments, APPLY computes the result of applying the functions to the arguments. It can serve as an interpreter for the system and is being used to check out programs in the LISP language before translating them to machine language.

(e)Work on a compiler has been started. A draft version has been written in LISP and is being discussed before it is translated into machine language or checked out with APPLY...

LISP was used at this time for calculations of properties of linear passive networks (Rochester, Goldberg, Rubenstein, Edwards, Markstein). It may be that some of the work for this was written in FLPL. Further use was for chess (McCarthy, Shannon, Kleinrock and Abrahams). "Other projects include the Advice Taker, visual pattern recognition and an

artificial hand. Work has been started by Bobrow, Maling and Park on a proof checker for predicate calculus and by Slagle on a program for computing indefinite integrals".

The same source (20) contains the first version of the well known Communications of ACM paper (21).

c) TOWARDS LISP 1

As we can see, in April of 1959 the author himself didn't understand the interpreter as the main instrument for making the language run. The S-language was only used for data.

But it is clear: the possibility for fast and direct access to the machine had its consequences. So over the time all the students and the programmers used more and more the S-language directly for programming. The M-language became more a means for communicating about LISP.

"The unexpected appearance of an interpreter tended to freeze the form of the language, and some of the decisions made rather lightheartedly for the "Recursive functions..." paper later proved unfortunate. These included the COND notation for conditional expressions which leads to an unnecessary depth of parentheses, and the use of the number zero to denote the empty list NIL and the truth value false. Besides encouraging pornographic programming, giving a special interpretation to the address 0 has caused difficulties in all subsequent implementations.

Another reason for the initial acceptance of awkwardnesses in the initial form of LISP is that we still expected to switch to writing programs as M-expressions. The project of defining M-expressions precisely and compiling them or at least translating them into S-expressions was neither finalized nor explicitly abandoned. It just receded into the indefinite future, and a new generation of programmers appeared who preferred internal notation to any FORTRAN-like or ALGOL-like notation that could be devised."(1)

It is not very clear if the M-language at the beginning (t.m. 1958) has included the PROG feature. It is very probably that the former programs were all more or less FORTRAN, so sequential parts were usual. The capacity to translate into S-expressions should be made the introduction of a special notation in the M-language (But this is a hypothesis of mine and can be proven only by the old memos. At present nobody, including McCarthy can find them).

The growing system, in any case the work of Goldberg in formula manipulation (symbolic matrices!) was one nice day full. So there was a need for a solution. The idea was to reclaim all used data and form a list of free registers. McCarthy developed the classical algorithm that uses a stack for saving unmarked list parts. Very probably this was not recursive in spite of the external notation and discussion. It is very curious to see, besides its stack usage, the nearly optimal performance of this version of GC-algorithm. It was only Toshiaki Kurokawa who found a faster solution (22). The work of Goldberg came to an end in the summer of 1959 (23). J. Moses claims his

simplificator program as the first ever written and he says it should be written in assembly language (24). The next problem solved in 1959 is the problem of free variables. All LISP-people know it as the FUNARG-problem.

J. Slagle, writing his program for integration, tried to write a function which could apply a test-predicate to all elements of an S-expression. It is very near to the function TESTR as described by R.A. Saunders (25).--See for a complete version in (1).

The function didn't run correctly and nobody had any idea why. The programmers and Slagle went to McCarthy..."..naturally he complained. And I never understood the complaint very well, namely, I said: 'oh, there must be a bug somewhere, fix it!' And Steve Russell and Dan Edwards said, there is a fundamental difficulty and I said, there can't be, fix it, and eventually they fixed it and there was something they called the funarg device. He tried to explain it to me but I was distracted or something and I didn't pay much attention so I didn't really learn what the funarg thing did until really quite a few years later. But then it turned out that these same kinds of difficulties appeared in ALGOL and at the time, the LISP solution to them, the one that Steve Russell and Dan Edwards had simply cooked up in order to fix this bug, was a more comprehensive solution to the problem than any which was at that time in the ALGOL compiler."(2)

D. Park claims that Patrick Fischer should have a part in this solution (1).

The next steps towards the first complete LISP-systems are :

- introduction of property lists (and the usage of properties EXPR etc.)
- introduction of pseudofunctions RPLACA and RPLACD
- introduction of floating point numbers to make arithmetic possible without using lists. A very impressive picture of how the early LISP did work with numbers can be studied using the paper of Jenkins and Woodward (26). The floating point numbers in LISP 1 were to be quoted if they came into the interpreter. Three functions (SUM, PRDCT and EXPT) could be used for arithmetical work.

The state of the LISP-language and system in september of 1959 could be studied if we had the paper (27). In November, 59, McCarthy started to write the first manual. But he didn't finish it and the LISP 1 manual was written by Dr. Phyllis Fox, who had joined the AI group in September, 1959.

In January 1960 the compiler, written by R. Brayton under assistance of D. Park was running. It is not very clear why McCarthy remembers only a failing compiler project in connection with LISP 1. Maybe he was waiting for an M-language compiler and they constructed an S-language compiler. The compiler was, if we follow the reports (28,29) indeed running.

This is also the claim of D. Park. The report (28) gives speed-up figures of 30 minutes in interpretative mode to 0.6 minutes compiled. The result was to some degree very similar to our later compilers, and to another degree very different. First, the compiler produced a LAP-code list. Then it was added as pseudo-function to the system. Many parts of it may be the model of the later compilers (30). The code

itself is very different in its kind to the later delivered code. For local variables it generated GENSyms which were appended to the end of the code. Recursive functions had to shift their local variables in the stack before they started. Labels were generated as instructions having the first place used (all instructions in LAP format had a first symbol field). CONS, NULL, PROG, RETURN, GO, CAR, CDR and possibly ATOM were translated open.

The well known paper (21) tells not very much about the state in March 1960. Only error diagnostics and tracing facilities are mentioned.

The manual (29) shows an interesting system. We found ca. 90 functions some of which were very singular (like the function CP1 in LISP 1.5), others stem from Goldberg's work (SIMPLIFY, DIFF, MATRIXMULTIPLY, REDUCE and REDUCETONXN for simplification, differentiation, matrix multiplication, matrix reduction resp.) or from former assembled programs.

An important (for the time!) part was the "Flexowriter system" which was constructed by Luckham and Edwards. Using a pre time-sharing possibility "time stealing", the user could communicate directly with the LISP system (cf. the story told by McCarthy in (1)).

d) TOWARDS LISP 1.5

We have not very much information about people and the work done in the time from 1960 to 1962. What we know is, that in March 1960 the LISP implementation for IBM709 was started. Then we know the start and the end product - LISP 1 and LISP 1.5 respectively - and that some new people are involved now: T.P. Hart, M.I. Levin, B. Raphael were new. But Luckham and Park don't work very much with LISP.

From the old students only Bobrow, Slagle and Abrahams wrote a thesis using LISP. Park writes about this: "... I think that McCarthy, while he was ahead of all of us in seeing the significance of the thing, and in his research ambitions, was as surprised as anyone that something of such general significance to computer people had emerged. LISP made it all so easy. Paradoxically, for those of us looking for research topics, it make look things too easy, in some sense. McCarthy went on to capture a good deal of the theoretical importance in his "Mathematical Theory of Computation" work - but he pursued this almost entirely on his own.

The message caught on much more slowly for the rest of us, or at least for me. None of the research students listed as authors of LISP 1 went on to do doctorate work using computers or about programming. For myself, having been excited by LISP, and having listened with ill-judged scorn to McCarthy's initial work on his theory of computation, which seemed too straightforward for intellectual ambitious students like me to worry with, I ended up writing a thesis on mathematical logic ..." (31).

At the end of this 2-years period, Rochester is back to IBM, but H. Rogers Jr. and H. Teager had joined the group. Further persons are : D.A. Dawson, E.L. Ivie, P.G. Jenson and U. Shimony.

The new system was running better and better and in august of 1962 the new manual was completed. It seemed to describe not the old language/system but a better thing. The plans for the second variant were discussed at length at that time. So it seemed not quite correct to call the system LISP 2 - a medium name was searched for and LISP 1.5 used.

The differences between LISP 1 and LISP 1.5 are :

1. LISP 1.5 allows integers, has other internal representation for floating point numbers and allows both to be evaluated without quoting.
2. The set of arithmetic functions is completely different and remarkably larger.
3. LISP 1.5 allows arrays.
4. The user has to deliver doublets in LISP 1.5 : the system starts always with an empty alist. (In LISP 1 there were triplets of function, argumentlist and alist. Top-level : APPLY). Toplevel now : EVALQUOTE.
5. LISP 1.5 allows pointed pairs.
6. At the same time the structure of the alist and the arguments for SASSOC are altered.
7. With the exception of LIST, in LISP 1 no function could have an indefinite number of arguments. Now a serie of such functions exists.
8. LISP 1 did allow only LISP input. LISP 1.5 has now a large set of I/O functions.
9. The flexowriter system isn't in existence. Device-I/O is handled by a monitor.
10. LISP 1.5 introduced the \$\$x...x notation for unusual atoms.
11. In LISP 1 there were difficulties with multiple CAR-CDR's. Some functions work for this (DESC, MAKCBLLR, PICK). In LISP 1.5 all is solved by shifting the problem to the user.
12. LISP 1.5 uses TRACE instead TRACKLIST.
13. The compiler is completely new.
14. A new LAP.
15. LISP 1.5 introduces CSET and CSETQ for setting constants (both are EXPRs !). LISP 1 has neither and has two distinct indicators for constants (APVAL and APVAL1).
16. LISP 1.5 alters the names : what is in LISP 1 the property-list is in LISP 1.5 the association list. What was the association list is now the P-list.
17. A lot of internal changes, some of them visible for the user.

18. LISP 1.5 introduces the error function ERRORSET.

Very soon the LISP system is adopted inside the time-sharing system. We drop here to mention the work of McCarthy concerning time-sharing. In summer 1962 McCarthy moves to Stanford. The further development is characterized by the strong connection to CTSS at MIT and the work of McCarthy to build up similar facilities at Stanford University. The delivery of a PDP-6 in December 1964 to MIT is the start point for LISP 1.6 that later (1967) is coming to Stanford. The concrete history between 1962 and 1966 is not full clear.

After 1966 we have for MACLISP J.L. White's paper (32).

The history of INTERLISP is clear in outline (as everybody can see in the preface of the manual (33)), but the connection to BBN's former work in LISP (as reported by Berkeley (34)) is unknown. Most of LISP people know the Berkeley-Bobrow book (35) about LISP. Bobrow has planned two further books (The Nature, Use and Implementation of the Computer Language LISP, Cambridge 1967; and LISP Applications, 1970) but neither seems to be appeared. Very unclear is the history of UNIVAC-LISP. We have only the author (Eric Norman), the location: University of Wisconsin, and a guessed time: 1969.

The first international echo on the LISP development came from G.B. Here in London a group around C. Strachey registered the paper (21) and during a tea-discussion Mike Woodger proposed D. Jenkins from the Royal Radar Establishment at Malvern to implement LISP. This was done by Jenkins and Woodward in 1960-1961. They report in their paper very little about the system on TREAC and more about LISP itself (36).

The next LISP implementation was done by H. McIntosh at the University of Florida in 1962. Very short after that L. Hawkinson wrote a LISP-system for the IBM709 at Yale University. Both of them went then to Mexico City and combined their work in LISP. In december 1963 to January 1964 the 1st International LISP Conference was held in Mexico City. Very little is known about; we have only a paper of M. Minsky concerning garbage collection and a paper of D. Edwards concerning secondary storage.

To come to an end, we mention only the beginning in some other countries: in 1965 J. Cohen implements LISP in ALGOL (38). In 1966 J. Kent implements LISP on a CDC3600 in Oslo, Norway. 1967 he implements with the help of J. Bolce LISP on IBM/360 in Canada. This work was concluded at Stanford in 1967. In 1966 V. D. Poel and V. D. Mey started in the Netherlands using a PDP-8. Delft is now a location where a lot of different implementations were made (mostly on PDP and X8). Actual work with LISP in Sweden seems to be started 1968 after foundation of the Datalogilaboratoriet under the direction of E. Sandwall by transferring and upgrading the CDC3600-system of Kent.

After the IFIP-symposium on symbol manipulation (39) some new LISP activities came out. In Poland S. Waligorski embedded LISP in ALGOL on a GIER-computer. 1968 Lawrow started in Moskow with some help from Berkeley on the BESM-6. In the same year we have the letter of K. Bahr to the CACM (40) indicating, that LISP is in Germany. In Eastern Germany we started 1970. In Czechoslovakia they started 1971. The same holds for Hungaria. In Italy a lot of different LISP systems seem to be

imported. Some work is known concerning MAGMA-LISP (41).

I am writing a book about LISP, which will contain not very much about programming, a chapter on application fields, a chapter on basic ideas, a chapter on history, a chapter on comparison of "famous" systems and a last chapter on LISP implementation. For the historical chapter every help will be recommended.

REFERENCES

- (1) J. McCarthy: LISP History. SIGPLAN Symposium on History of Programming Languages, Los Angeles, June 1978.
- (2) J. McCarthy: Talk about LISP history, held in Summer 1974 at MIT.
- (3) J. McCarthy, R. Brayton, D. Edwards, P. Fox, L. Hodes, D. Luckham, K. Maling, D. Park, S. Russel: LISP Programmer's Manual, March 1 1960, MIT, Comp. Center and RLE, Cambridge, Mass.
- (4) J. McCarthy: Inversion of Turing Machines, in J. McCarthy and C. Shannon (Eds.): Automata Studies, Princeton, 1956.
- (5) P. McCorduck: History of AI, Advance Papers of 5th IJCAI, 1977
- (6) A. Newell, H. Simon: The Logic Theory Machine - A Complex Information System, IRE Trans. Information Theory, Vol IT-2, No. 3, Sept. 1956, pp 61-79.
- (7) Preliminary Report: Specification for the IBM Mathematical Formula TRANslating System, FORTRAN, IBM Corp., Prog. Res. Group., Appl. Sci. Div., 1954.
- (8) The FORTRAN Automatic Coding System for the IBM 704 EDPM, Programmer's Manual, IBM Corp., 32-7026, Oct. 1956.
- (9) A. Newell, J.C. Shaw, H. Simon: Programming the Logic Theory Machine, Proc. WJCC, Feb. 1957.
- (10) A. Newell, F.M. Tonge: An Introduction to IPL V, Comm. ACM 3,4,1960.
- (11) H. Gerlernter, J.R. Hansen, C.L. Gerberich: A FORTRAN Compiled List Processing Language, Journ. ACM 7,2, 1960.
- (12) H. Gerlernter, N. Rochester: Realization of a Geometry Theorem Proving Machine, ICOI Proc., Paris 1959.
- (13) H. Gerlernter, J.R. Hansen, D. Loveland: Empirical Explorations of the Geometry Theorem Proving Machine, Proc WJCC 1960.
- (14) J.R. Hansen: The Use of the FORTRAN Compiled List-Processing Language, IBM Th. Watson Res. Center, Res. Rpt.:RC-282, June 1960.

- (15) J. McCarthy: Letter to the Editor, Comm. ACM 2,8, 1959.
- (16) A. Church: The Calculi of Lambda-Conversion, Princeton, 1941.
- (17) J. McCarthy: Programs with Common-Sense, Proc. of Symp. on Mechan. of Thought Processes, Nat. Phys. Lab., Teddington GB, Nov. 1958.
- (18) E. Dijkstra: Making an ALGOL translator for the X1, reprinted in: Ann. Rev. Automatic Programming, R. Goodman (Ed.), Pergamon Press, 1963.
- (19) G.E. Collins: A Method for Overlapping and Erasure of Lists, Comm. ACM 3,12,1960.
- (20) Quaterly Progress Report no 53, april 1959, RLE, MIT, Cambridge, pp 122-152.
- (21) J. McCarthy, Recursive Functions of Symbolic Expressions and their Computation by Machine, Comm ACM, 3, 3, 1960.
- (22) Toshiaki Kurosowa, A New Save and Fast GC-algorithm, Tokyo, 1976.
- (23) S. H. Goldberg, Solution of an Electrical Network Using a Digital Computer, MIT, MS Thesis, 1959.
- (24) J. Moses, Simplification - a Guide to the Perplexed, 2nd Symp. on alg. and symb. manipulation, ACM, 1971.
- (25) R. A. Saunders, LISP - on the Programming System, in (34).
- (26) D. R. Jenkins, Woodward, Atoms and Lists, Comp.J., 4 april 1961.
- (27) J. McCarthy, LISP - a Programming System for Manipulating Symbolic Expressions, Annual Meeting of ACM, MIT, Cambridge, sept. 2-4, 1959.
- (28) Quarterly Progress Report no 56, january 15, 1960, RLE, MIT, Cambridge, pp 158-161.
- (29) see (3).
- (30) R. A. Saunders, The LISP Listing for the Q-32-Compiler, in (34).
- (31) D. Park, personal communication, 1978.
- (32) J. White, LISP : Program is Data - a Historical Perspective on MACLISP, MACSYMA 1977 users conference, Berkeley.
- (33) W. Teitelman, INTERLISP Reference Manual, Palo Alto, 1974.
- (34) E. C. Berkeley, D. G. Bobrow (eds), The Programming Language LISP, its Operation and Applications, Cambridge, 1964.
- (35) E. C. Berkeley, LISP - an Introduction, Computers and Automation, Sept 1964.
- (36) see (26).
- (37) M. Minsky, A LISP Garbage Collector Algorithm Using Serial Secondary Storage, AI-memo 58, MAC-M-129, MIT, 1963.

- (38) D. J. Edwards, Secondary Storage in LISP, AI-memo 63, MAC-M-128, MIT, dec. 27, 1963.
- (39) D. G. Bobrow (ed), Proc IFIP Conf. on Symbol Manip. Lang., Pisa, sept 1966, N.Y. 1968.
- (40) K. Bahr, Letter to the editor, Comm.ACM, 11, 6, 1968.
- (41) Montangero, Pacini, Turini, MAGMA-LISP, Adv. Papers 4th IJCAI 1974.