TOWARDS A LISP STANDARD

Herbert Stoyan%, Jerome Chailloux*, John Fitch¢, Tim Krumnack#, Eugen Neidl?, Julian Padget¢,
Guiseppe Attardi^, Thomas Christaller$, Jeff Dalton=, Mathieu Devin*, Bernard Lang*, Ramon Lopez de Mantaras:, Eric Papon?, Stephen Pope@, Christian Quiennec[, Luc Steels+

Abstract

This paper reports work in progress within the European LISP community on efforts to bring the LISP language to national and international standardization. The paper discusses the situation as seen by the authors and some of the conclusions that have been drawn.

1. Introduction

The main impetus for the work reported here presently comes from Europe. So far, most activities have also taken place in Europe. It was the idea of J.Chailloux, T.Christaller and S.Pope to form a group of European LISP implementers or LISP experts for starting to prepare for a LISP-standardization.

It is not our intention that this effort should continue to be a local European activity. It is yet relatively early in the standardization process and at the time of writing it has only been in progress for a few months. Already, however, a significant part of the US LISP research and implementation community has become interested, as well as several individuals working for commercial organisations.

Standardization is an active process and so, necessarily, the target will become clearer in the future time. This may also entail a revision of some of the material presented below. The product of standardization shouldn't be a new dialect nor a family of dialects but a criterion for dialect classification. The standard should contain the minimal requirements for a LISP implementation for calling it "following the standard".

% IMMD6, University of Erlangen, D-852 Erlangen, Germany
* INRIA, Domaine de Voluceau, Rocquencourt, BP 105, F-78153 Le Chesnay, France
¢ School of Mathematics, University of Bath, Bath Avon, BA2 7AY, United Kingdom
? Lab de Marcoussis, CROGE, Route de Nozay, F-91460 Marcoussis, France
Krupp-Atlas Elektronik, PSF 448545, D-28 Bremen, Germany
^ DELPHI SPA, via delle Vetraia 11, I-55049 Viareggio, Italia
$ GMD Birlinghoven, PSF 1240, D-5205 St.Augustin, Germany
= AIAI, University of Edinburgh, 80 South Bridge, Edinburgh, EH8 9NW, United Kingdom
: Centre d'Estudis Avancats, CSIS, Cami de Snta Barbara, Blanes, Girona, Espana
@ PCS GmbH, Pfaelzer Wald Str. 36, D-8 Munich 90, Germany
[STEI, Fort d'Issy, F-92131 Issy-Les-Moulineaux, France
+ AI Lab., Vrye Universiteit, Pleinlaan 2, B-1050 Brussels, Belgium

2. The development of LISP (or: Will a standard get accepted ?)

The LISP1 and LISP1.5 systems were implemented at one site. McCarthy's design of the LISP-description in LISP helped to maintain a de-facto LISP standard until 1965. LISP1.5 was still an important dialect at least until 1969. Afterwards really incompatible implementations were created - to mention only BBN-LISP (later: InterLISP [Te]) and LISP1.6 (later MacLISP). They were mere local dialects having at most 50 users each. The users were fond of their local implementation wizards and used a kind of sectarian pride.

There is some reason that McCarthy did not care much about the deficiencies of LISP1.5 (1962 he argued against the divergence between "normal" value and "functional" value [Mc]) and the spreading dialects because he hoped in LISP2. The dialect problem became pressing when this project failed.

Most of the existing dialects contained a second problem: They were in fact two dialects in one - an interpreter dialect and a compiler dialect. With the second LISP compiler a static binding scheme was used as semantics for compiled programs. It was a great achievement when the BBN-LISP-Compiler produced code which was semantically constistent with the interpreted program. This was done by using the dynamic binding scheme.

The first who became confronted with the situation was A.C.Hearn who had a evolving computer algebra system and hundreds of users all over the world using different machines running different LISP dialects. In consequence he proposed a "StandardLISP" already in 1969 [He]. But the problem of porting big application was not yet recognized. It would have been easy to standardize LISP in 1969.

In the seventies the scene changed. The dialects had to be ported. Only XEROX started a new approach by designing a "Virtual Machine" [Mo]. This was intended to be usable as basis for a new InterLISP-Standard and could have been used as model for a common LISP standardization. But it proved to be of restricted value because it was not usable as basis for a re-implementation of InterLISP. This dialect turned out to be nearly not transportable [BD]. The standardization problem was recognized now locally.

1978 Hearn and his colloborators in Salt Lake City made another step [Mea] in order to overcome the pressing dialect problem by defining a small basis as StandardLISP and by developing an implementation technique for it. The 2nd StandardLISP was ignored by the AI-connected LISP community again.

All the various dialects brought in new features whose pros and cons could be studied by using the dialects. A dialect which went in the direction of conceptual consolidation was SCHEME [Su] by taking a clear decision for static scoping and - in the spirit of McCarthy's ideas - by dropping the separation between a normal and a functional value of symbols. [Fr] and [Cea] represent the current state of development of that dialect. Another approach towards a complete revision and refinement of LISP was B.Smith's 3-LISP [Sm]. Smith argued not only for static scoping a. ' cleaner evaluation (to be understood as normalization) but offered a way to understand FEXPR's.

The definition of CommonLISP [Sea] was done for overcoming the diversification in MacLISP. It was based on the LISP culture of the MIT nearly exclusively.

3. Implementation techniques

Concurrently, there has been a lot of progress on implementation methods for LISP. The definition of virtual machines (programming environment and interface to host operating system), abstract machines (target code for compilers, so that only an assembly phase is needed to complete the process) and machine descriptions (aids automation of the generation of an assembler) have been instrumental in this process. Practical evidence of this is the PSL dialect [Gr] for which a very sophisticated new compiler has been developed [KE] and from which the Portable Common LISP Subset (PCLS) [KU] is being built, and LeLISP [Ch] with its combined virtual and abstract machine LLM3 [IN].

A LISP standard should be easily transportable by using such methods.

4. The question of semantical description

LISP1.5 was described by McCarthy's famous APPLY-EVAL-functions as a kind of meta-linguistic interpreter. A description of the growing LISP dialects with their large sets of special functions has become increasingly difficult. Obviously, SCHEME is easier in this respect [Mu], [Cea], and subsequently a compiler based on a denotational semantic description of SCHEME was written [Cl]. If we use as basis of the LISP standard a similar language, then the formal description should be possible. That would be a safe basis.

5. The question of different power

LISP systems and their environments vary from very large (InterLISP [Te], ZetaLISP [WM]) to very small (UOLISP [Marti, 198?]). The former require specialized hardware or at least dedicated hardware of about a VAX power, whilst the latter exist on Z-80s (or comparable machines). Such a linear spectrum, however, fails to capture the essence of the differences in power between these systems which live on such widely differing scales.

This problem suggests that the 'measurement' or classification ordering should be done using at least two comparisons: one compares the set of features of the language implementation and the other the features of the support environment. It seems that the different LISP-systems form a lattice. The bottom of the lattice corresponds to a LISP system with a minimal set of functions, simple parameter passing conventions and only file/teletype input/output.

The main reason for establishing a consistent relationship between adjacent nodes of this lattice is, of course, portability and compatibility.

It is interesting to move up in the lattice because different people have different facilities and the transition from one system to another ought to be as painless as possible. Given that an important part of research and development is collaboration and teamwork an homogeneous software environment is considerably important.

Moving down might be important in the future as more commercial applications of LISP become apparent because there will be a need for 'delivery systems' which are limited capability environments containing a just sufficient subset of the facilities of the development environment.

6. Our Conclusions

(i) It is possible to describe LISP in a clean formal way and the standardization of the language should take advantage of that. This could be done by ensuring that the lowest level of standardisation has a formal definition and is therefore self-consistent. Then this property will be inherited by later levels of the standard if they are properly related.

(ii) LISP is needed to run on a variety of machines with widely differing requirements for the degree of support environment that standardization on a single level is unreasonable. The standardization of LISP should encompass several levels of complexity in both the language and the environment. The highest level of conformance with the standard, at least along the language axis, could correspond closely to CommonLISP.

(iii) The sophistication of the run-time support environment is so intimately bound to the language that it is just as much a matter for the application of standards as the langauge itself.

Such a programme will not necessarily be easy; for example there may be operations which cannot be accepted as defined in the CommonLISP reference, because the specified behaviour contravenes the formal specification of the language. In some cases this may mean that the formal definition is too limited, in other cases it may mean that something is inconsistent in CommonLISP, thus the revision will be a two way process. It has already been observed that parts of the Common LISP reference are unsatisfactory [Se] as experience with implementation increases, and so it is being revised.

Each level of the standard should be viewed as the implementation language for the next and so any program which works at level n will also work at level n+i, but not necessarily at level n-i (i=1..).

7. The situation of the standardization effort

Programming language standardization is an international matter. The ISO creates working groups which design a language description and this is proposed as a standard to the national standardization organisations. If they agree, a vote is made and the standard will formally accepted.

At present, the ISO has agreed with the existence of an "ad-hoc working committee" headed by R.F.Mathis. He planned to submit a "New Work Item Proposal" in March. For forming an regular Working Group two ballots had to be made. At present, the plan seems not to be followed. Instead, an ANSI working group has been founded in March (1986) for proposing a CommonLISP standard. The members are: A.Bawden, D.G.Bobrow, S.Fahlman, R.P.Gabriel, M.Griss, D.Moon, J.A.Rees, G.L.Steele, and D.Weinreb.

In April, AFNOR (the French standardizations organization) has proposed to organize a new work item for LISP in ISO and take the technical secretariat.

As it stands, there are two lines of work: The ANSI group intends to standardize CommonLISP on the basis of the "aluminium book" [Sea]. The European group proposes a general revision of LISP resulting in a standardization of sound and clean concepts. The result can be CommonLISP but has not to be.

It is clear that the standardization of LISP - which has to be done in cooperation between LISP people in the USA and in other countries (including Europa) suffers from the bad experiences of past standardization efforts. The PEARL-standardization was cancelled by means of bad tricks of the US ISO members. ADA is anything but a success. The fight for a PASCAL standard is still going on. The idea of making an "CommonLISP standard" is a hint. We know that for a world-wide LISP standard we need the support of our friends in the USA. We hope that our friends see the picture from a similar view. The question should not be how much money was invested but how reasonable the standard will be!

8. The European Proposal

At present, the European group is developing a proposal [SCFKNP] for the discussion with the ANSI group which is featuring a 3 levels standard. The basic level is a minimal LISP in the spirit of SCHEME, the first level is a medium sized LISP with a well-choosen set of basic functions and the second level is a "big" LISP in the spirit of CommonLISP. The language levels are intended to be completed by environment levels.

The intention of this level approach is to enable the standardization of LISPs suitable for teaching, of LISPs implementable on minicomputers as well as large LISPs which are thought to be necessary for industrial applications. We believe that there are at least three different user groups which could be served this way. (One might have doubt that a very big language is the best for industrial application; there are negative examples (LISP2, PL1, ADA). A smaller thing seems to be more handy and understandable.) It would be a good thing if the language learned in LISP courses is a real part of the language used in big applications.

The relation between the levels is not only upwards compatibility. We hope to be able to describe in a level all features of the next higher level - somehow, not necessarily in an efficient way. This should help to have a clear level 2. If the level 0 is described in a formal manner - what we believe is important - then level 1 and level 2 should be as clean as possible. There was the idea that the level 0 interpreter has already all of the important features. In this case the standardization effort for the higher levels would be smaller - they could be regarded to be extensions only. But we believe this approach would destroy our hope in a clean minimal level 0.

We propose to standardize as level 0 a small minimal LISP. It contains only a few data types (numbers - integers and floats, characters, symbols, conses, vectors and strings, functions, streams). For each of them only a minimum of functions (classified into: type checking, construction, selection, modification, comparison, conversion) is part of the standard. To make the level-0-language

really useful a device for introducing new user data types is proposed. This device enables not only the introduction of a new type (via a kind of DEFSTRUCT) but additionally a means to put the new types into the type system.

The type system maintains a (circle free) graph of types. Types with subtypes are union-types. The type system does not provide inheritance of any kind. What is inherited is rather the set of functions which has to be implemented at some point than function definitions. One can introduce new union types, add new member types to an union typ, delete a type, ask for inherited required functions, ask for implemented functions.

The control part of the level-0-language is characterized by the lexical scoping mechanism, thr usual "structured" control structures (IF or COND, AND, OR, PROGN, LET, CATCH and THROW). The "call-with-current-contionuation" would be an alternative for gloabl exits.

The question has been asked, why SCHEME itself was not taken for level 0. We believe the Scheme of the RRRS [Cea] should not taken:

1. because of its strange arithmetics (the exact-inexact dichotomy),
2. because functions are not first-class-objects (a FUNCTIONP is missing and selector/modifiactor functions),
3. because its strange access to the environment (only during function definition; there's no EVAL, variable names don't matter - which is a problem in LISP and therefor the name "lexical binding" seems to be a misnomer - "static binding" that's what it is),
4. because it is not minimal (a lot of functions is called "essential" which in fact might be defined),
5. because you can't expand the type system,
6. because there should be Macros.

There are points were Scheme is cleaner than our level 0 proposal. We are still in discussion and hope for further discussion rounds. We didn't take:

a. a Boolean data type,
b. a difference between the empty list and false,
c. a difference between any data except false and true,

Moreover, we thought about following B.Smith [Sm] in some way but didn't find a good compromise (the upward compatibility!).

Level 1 could be seen as a "kernel" of a modern LISP. It should have the size of LeLISP [Ch] or PSL [Gr]. The important things which are added to the level-0-langage: More data structures, more basic functions (e.g. property lists for symbols), dynamic binding, multiple values etc. Adding dynamic binding with going up a level seems to achieve more cleaness. This topic was hardly discussed and still some of us have the feeling that dynamic binding should discarded completely. At present, the introduction of dynamic binding is planned to be restricted to a special form: DYNAMIC-LET. This enables a lexical detection of dynamic variables.

At this point we want to make a short statement on name spaces. In CommonLISP, symbols can be used for at least 15 different purposes and one con construct a program which culminates in an area where one and the same symbol carries at least 9 meanings: 2 functional (function/special form/macro), block name, tag, catcher, variable, data type, package name, pathname, datum, p-list name, ...

We believe the language should not permit this: there should be always only one (static visible) meaning.

Level 2 is a luxurious LISP of the size of CommonLISP. It could be a cleaned CommonLISP - with the open questions answered, the incomplete descriptions made complete, the dark spots illuminated, the unclear points made clear and the contradictions resolved. At present this level is thought to introduce packages, the complete lambda-keyword-list, and a voluminuous I/O subsystem (including formats, window basic functions etc.).

There is a document describing our proposals which is still not complete. It contains at present a good overview on level 0, a short draft for level 1 and some hints for level 2.

9. Literatur

[BD] R.L.Bates, D.Dyer, , M.Koonen: Implementation of InterLISP on the VAX 2nd ACM LISP Confer 1982

[Ch] J.Chailloux et al.: Le LISP, Portable and Efficient LISP System. INRIA Rocquencourt, Juil. 1984

[Cea] W.Clinger (Ed.): The Revised Revised Report on Scheme. MIT AI Memo 848, Cambridge, Aug. 1985

[Cl] W.Clinger: The Scheme 311 Compiler - An Exercise in Denotational Semantics. 3rd ACM LISP Conference, 1984

[Fr] D.Friedman et al.: Scheme84 Interim Reference Manual. Indiana University Computer Science Techn. Rep. 137, Feb. 1985

[Gr] M.Griss et al.: PSL - a portable LISP system. 2nd ACM LISP conference, 1982

[He] A.C.Hearn: StandardLISP. 1st LISP Bulletin, SIGPLAN Notices, 1969

[IN] J.Chailloux: La Machine LLM3. INRIA Rapport Technique No.55, Juin 1985

[KE] R.Kessler et al.: EPIC - a Retargetable Highly Optimizing LISP-Compiler. to app.: ACM SIGPLAN Conference on Compiler Construction, 1986

[KU] R.Kessler et al.: A Portable CommonLISP Subset with High Performance. to appear: 4.ACM LISP Conference, 1986

[Ma] J.Marti: UO-LISP Reference Manual, Nordwest Computer Algorithms, Los Angeles, 1984

[Mea] J.Marti et al.: Standard LISP Report. University of Utah, Salt Lake City, 1978

[Mc] J.McCarthy: A New Eval Function. MIT AI Memo 34, Cambridge, 1962

[Mo] J.S.Moore, The InterLISP Virtual Machine Specification. XEROX Parc, CSL-76-5 Palo Alto, Mar. 1976

[Mu] S.Muchnick, U.Pleban: A Semantic Comparison of LISP and Scheme. 1st LISP Conference, 1980

[Sm] B.Smith: Reflection and Semantics in a Procedural Language. MIT PhD Thesis, Dept. of EE and CS, LCS TR 272, Cambridge, 1983

[Sea] G.L.Steele (Ed.): CommonLISP - The Language. Burlington, 1984

[Se] G.L.Steele: Private Communication with J.Padget on CommonLISP, 1986

[St] H.Stoyan: Early LISP History. 3.ACM LISP Conference, 1984

[SCFKNP] H.Stoyan, J.Chailloux, J.Fitch, T.Krumnack, E.Neidl, J.Padget: A Proposal for an IsoLISP Standard. Draft. May 1986

[Su] G.J.Sussman, G.L.Steele: Scheme, an Interpreter for Extended Lambda Calculus. MIT AI Memo 349, Cambrige, 1975

[Te] W.Teitelman: InterLISP Reference Manual. XEROX Parc, 1978

[WM] D.Weinreb, D.Moon: LISP Machine Reference Manual. MIT, Cambridge, 1980