A subjective view of Lisp

Christian Queinnec
Laboratoire d'Informatique de l'École Polytechnique
91128 Palaiseau Cedex — France
Internet: queinnec@poly.polytechnique.fr

January 14, 1990

The editor of Lisp Pointers has been asking me for a long time to write down my view of Lisp. I was even given permission to flame. This paper is the result and, naturally, is entirely my own opinion.

Since the first rumors of a possible standardization of Lisp started to circulate (end of 85), many fights have taken place. They were never scientific nor technical: they were mainly commercial. Before trying to explain the state we have reached, I will try to define the ecological niche of Lisp.

Lisp is old. Very old. Its remarkable conception makes it a terrific local extremum. Unlike the monsters born around the same time (Fortran, COBOL, Basic) which simply swallowed linguistic features they lacked, Lisp has evolved until it has become a sophisticated family of languages.

Lisp has been the jewel—despite Joel Moses muddy analogy— of AI laboratories. Lisp has also bred a large community people for whom Lisp was the right language, enriched by a prodigious software sedimentation, which lead to exciting environments and dream hardware. But the boosting of AI which took place in the '80s is now past and the number of practising Lispers has probably gone back to what it was before.

Three niches are apparent to me: hypercomplex systems, extensible systems and education.

- Hypercomplex systems are on the edge of research. They are carried over by a few people which want to devote their time on their problems and need powerful environment and high throughput. Usually they are software masters and know their tools perfectly.
- Extensible systems are systems offering numerous primitives that the user may freely compose even in unexpected way. Gnu Emacs is perfect example. The glue is Lisp because it is simple, regular and composable without restriction. Extensions can be written simply and thus provide answers to unexpected combinations of primitives.
- Education needs Lisp for many reasons: Lisp has no syntax, is simple to implement, allow to study in vivo many linguistic aspects. However the functional family and in particular ML are taking an increasing part of this niche.

The word "Lisp" gathers many different desiderata in the previous niches. Let us summarize them.

hypercomplex systems	needs powerful language, needs large libraries and rich integrated environment	few users
Extension systems	needs simple but extensible language, must be convivial with the external world, needs efficiency and small core image	many users
Education	structured design, clear semantics, syntactically and semantically extensible, efficiency is not a problem	some users

Hypercomplex systems extend to "simply" complex systems. They both need Lisp as design language since they require rich data types, paradigms and libraries.

Among existing Lisps, COMMON LISP or more exactly its current implementations, seems well adapted for hypercomplex systems: nearly everything is in them, even twenty screwdrivers ! Of course the official language lacks some features like multiprocessing, windowing ... but these are offered everywhere and hypercomplex systems need not be ported every other day. Learning COMMON LISP is long, its complexity makes it entangled for universities which cannot afford the price of the documentation reproduction.

Another contender is Le-Lisp[1]. All incarnations deriving from a common core make applications more portable across computers, operating and windowing systems.

Scheme meets well the requirements of education and research. Scheme papers are more numerous than Lisp papers (check your recent L&FP proceedings). Scheme is a very clear language and its tutors follow a zen philosophy. Anything unessential or controversial (i.e. not so well understood) is thrown away. The latest Scheme report is an admirable document and as semantical analysis progresses, slicing molecules then atoms and afterthat quarks I dare say that the revised$^{\infty}$ report on Scheme will converge to λ-calculus. Perfection depends on time but does not serve industrial interest. Having no libraries or macros eliminate Scheme from the industrial world which would rather tend to incorporate CL-like features.

A plethora of Lisps exist for extension systems. Standardization does not tamper the production of small Lisp kernels such as ELI, ELK, GnuLisp (probably the most run Lisp today), AutoCadLisp, TinyLisp, Scheme-ToC, Winterp ...

I thus see a powerful language which size steps from the 465 pages of the aluminium book to the 746 of the second edition. Such fat would likely kill the healthiest superman ! Had these addenda been libraries it would not have hurt but many new linguistic features were added (`symbol-macrolet`, `compiler-macroexpand` ...) which need is evident and ancestral but which I dislike since too many patches incline me to a reconception. Beside this dinosaur one can see many agile young mammals the eldest being Scheme and the newest being Haskell and EuLisp.

ISO is not an academic forum for debating dreams. ISO is stuck with companies problems, investments and money returns. Each decision is weighted against the gain and loss nations will instantaneously make if taking a decision, its opposite or no decision at all. No real industrial Lisp system vendor can afford gratuitous changes to its implementation and philanthropy is only a bet on future gains. I certainly would not keep shares of companies acting differently.

The sensibility to the standardization depends on the niche. For industrial use, it seems obvious that a standard is strongly needed. Users can live with any Lisp (they already proved it for almost 30 years) but they just need one as soon as possible. For vendors, standardization does not seem so urgent although it must be conformant to their implementation. Moreover only vendors participate to the ISO standardization. For education, standardization is unnecessary or even undesirable since it may impede (or kill) the community. Scheme was once the "free style" garden of Lisp. Will it keep this rôle or will research migrate towards other fields ? I do not have the answer, but I do have the feeling that while Lisp was born in the USA —as emblazoned on RPG's and Dussud's T-shirts at one ISO meeting— it might well also be buried there.

[1] Le-Lisp is a trademark of INRIA.