9th European Lisp Symposium - personal report

Michał Herda

Faculty of Mathematics and Computer Science,

Faculty of Physics, Astronomy and Applied Computer Science

Jagiellonian University, Cracow

Summary

This is a personal report covering my experiences and personal discoveries made during the 9th European Lisp Symposium. I make notes and reflect on the general course the conference took and then elaborate on specific parts of it that caught most of my attention and gave me the most insight and inspiration. In particular, I write about a project of mine called the Common Lisp UltraSpec, research done on Clasp Common Lisp and the possibility to bootstrap one Common Lisp implementation off another.

1 Assessment

This year's European Lisp Symposium[1] was my first; not only it was the first Lisp conference I attended, it was the first true programming/research conference in my life. From the limited experience that I have, though, I can say that it set a very good impression inside my mind that I would like other conferences to follow.

I attended the conference as I am very much interested in Lisp and the design of programming languages and compilers. One of traits of Common Lisp is its ability to morph into a variety of other languages through its versatile macro system and its programmability - the reader, the compiler, the evaluator, all of these elements are written in Lisp itself and can be freely extended and expanded. Because of this, Common Lisp, aside from being a language on its own, is also a very competent metalanguage, meaning, a language for designing other languages. I find this trait of Common Lisp fascinating and worth much more research and study; I attended the ELS with hopes of finding inspiration and knowledge on this.

The audience of the conference was very heterogenous: there were people who borrow ideas and inspirations from Lisp and implement them into various other languages; there were people who bridge the gaps between Lisps and many other languages and programming structures; finally, there were people actually utilizing Scheme, Clojure, Common Lisp and other Lisp dialects in an immensely varied collection of use cases. After the conference I can say that without a doubt, despite common beliefs, Lispers or Lisp in

[1] `http://www.european-lisp-symposium.org/`

general are not extinct at all - according to my personal calculations, this year's ELS had about 60-70 attendees.

Organization-wise, the conference was really well thought-out. The location the conference was held at was good and close to many nearby hotels and the town center, which allowed me to give some of the guests a small tour of northern Cracovian Old Town by the end of the conference; Cracow has been the life centre for the last few years and I was able to share my knowledge of its history and sightworthy regions. There were no delayed talks and the catering kept all of us adequately fooded throughout the whole conference; the welcome party on Sunday and the farewell banquet on Tuesday were also very good.

I attended the conference with many more people from Poland. Among them were Kamil Ziemian - the second attendee from the SLiMaK students' association, many people from the community based around the `#lisp` IRC channel on Freenode and the `kraklisp` Lisp group that I coordinate, many other people from Poland and the rest of the world. Thanks to this, I was able to make many acquaintances and commence a series of interviews with the majority of the conference's speakers.

Summarizing the conference in a few sentences, I found the general opinions to be rather good. From what I was able to perceive through this last week since the ELS, many other attendeed returned home augmented and inspired by the occurrences and experiences of the Symposium - me included.

2 Research

2.1 Interpersonal matters

I have met many wonderful, inspiring and interesting people on the ELS and was able both to meet many members of the Lisp community; from what I observed, I also happened to make myself known for the reasons that follow.

As I heard during the first day of the ELS, *Lisp requires bravery.* I had a chance to prove it later during that day as I joined Nina Palińska, a singer and my girlfriend, at an impromptu recital of ancient and Slavic music to entertain our guests. The idea appeared about six hours before the event so there was little time for preparation, but the recital happened and, from what I could perceive, was well acclaimed by the conference participants.

During the conference, I was able to conduct a series of short impromptu interviews that are uploaded publicly to the `kraklisp` Youtube channel[2]. I have managed to interview the majority of people who gave speeches at the ELS; I hope that the material I've gathered will serve as another short reference of what happened during the ELS and a possible source of inspiration and information for people interested in Lisp.

[2]`https://www.youtube.com/channel/UCymtXMj1M7cKiV9TKLoTtEg`

2.2 Common Lisp UltraSpec

Thanks to the hospitality of the organizers and thought put into the flow of the conference, both days of the conference ended with series of lightning talks. I participated in the second one and talked about two interconnected things. The first was the current problematic state of Common Lisp documentation, the second was my proposed solution to that issue - a project that I named the Common Lisp UltraSpec[3].

It is unnecessary to duplicate information from the original UltraSpec manifesto that I posted about three months ago, although interested readers might want to read original post[4]. To sum up, the original documentation is dated, scattered, lacks hyperlinking, corrections, versioning and extensibility. I am trying to address these problems by creating a basis upon which the existing documentation can crystallize again in a reformatted form that's augmented and reviewed, while also being tightly knitted to all of its other modules by a net of hyperlinks.

My personal research on the details of the form and way of the final documentation continues as, after the lightning talk, I can see the UltraSpec being mentioned in various contexts. I am glad to see a small discussion arising around the UltraSpec with all the hopes its worries emerging around the it; they provide me with insight and tell me which pitfalls I might want to avoid in the future.

2.3 Compiler design

A very interesting insight came from professor Christian Schafmeister from Temple University, USA, who spoke about his research on nanomaterial design. The basis for his research is the general unfitness of proteins as a building material for nanostructures and the development of an artificial alternative - spiroligomers, or, as he calls it, *molecular Lego*. His frustration with libraries and tools written in Python and pure C++ led him to create a new Common Lisp/C++ implementation that suits his design goals - Clasp[5].

Clasp is a new Common Lisp implementation that seamlessly interoperates with C++ code, making data and functions from both languages available in each other and compiling both languages into homogenous intermediate representation that is then processed by LLVM to produce executables. Along with Clasp, Christian is developing CANDO[2][6] - a tool and Common Lisp extension specialized in molecular design, modeling, optimization, and simulation.

Developing Clasp is particularly intriguing to me as this research project is particularly innovative. This is unexplored area; there has been no previous research upon the matter of creating a compiler for a high-level language like Common Lisp meant to freely interoperate with libraries written in C++ and retaining most of its speed.

Christian had two presentations for the ELS, but time permitted him to show only one of them. The talk that he gave at the conference was a direct presentation of the capabilities of CANDO and the current outcome of his research - a nanoparticle capable

[3]`http://phoe.tymoon.eu/clus/`

[4]`http://phoe.tymoon.eu/clus/doku.php?id=articles:manifesto`

[5]`https://github.com/drmeister/clasp`

[6]`https://github.com/drmeister/cando`

of filtering water through reverse osmosis while being much more resilient and durable when compared to current protein-based solutions.

The other presentation contained more information about Clasp and its implementation details. Christian showed it to me and a few other people after the conference ended along with a simple demonstration of how one is able to write usual C++ code and expose it to the Common Lisp language. He also showed us slides showing the current process of compiling Common Lisp and C++ code to the AST, and pruning it of unnecessary parts for boosting the speed a function.

Christian also talked about how it is possible to improve Clasp. Currently, Clasp uses the Cleavir Common Lisp compiler, created by Robert Strandh as a part of SICL Common Lisp implementation. Christian talked about how utilizing Cleavir gave Clasp a giant speed boost, placing it within a single order of magnitude of C++ for simple computation tests while keeping the code safe through type checks. He also mentioned how there is still space and need for improvement within the compiler - there is the possibility of direct type inference during compile-time for further boosting the performance of the resulting code.

Further chat with Christian on the IRC channel ensures me that his research is in fact exploring the unknown; the questions that he asks and which are fundamental to his research are not answered in classical books about compiler design and need intensive study on their own. Despite him being a professor at the Department of Chemistry, I suspect that his current study in design and architecture of compilers might change computer science - a discipline quite far from molecular chemistry. There is a saying that the omnipresence of C/C++ language impeded compiler design in general by about ten years; by creating a new compiler that is able to leverage C++'s strengths and impressive codebase while covering its weak spots with Common Lisp expressiveness, techniques and philosophy, Christian might be currently creating a work of compiler art that I am glad to be able to watch and will be honored to participate in in the future.

2.4 Language bootstrapping

One more person I was able to meet at the ELS was professor Robert Strandh from Université de Bordeaux, France; it was the first chance for me to meet him in person. I had met Robert on the `#lisp` IRC channel months before that; he needed an apprentice while I needed a teacher and this turn of events led me to ask him for mentorhood, which he accepted.

I had a chance to speak with Robert after his second ELS talk, *A Modern Implementation Of The LOOP Macro*[3][7]. Robert developed this `loop` implementation as a part of working on his SICL Common Lisp implementation. A part of his talk that struck me particularly hard is, Robert openly admitted that he utilized a full implementation of Common Lisp in his implementation of `loop` - which included `loop` itself. This apparent circularity puzzled me hard enough to ask him for explanation - how it was possible and why he utilized this in such a way.

[7]`http://metamodular.com/loop.pdf`

He ended up explaining the process of bootstrapping parts of a language, starting with a common myth that my mind was confined in as well - if one needs to write machine code, one either has to do assembly or C; if one needs to write web browser code, one has to write Javascript.

He explained to me how it is possible to utilize an already existing Common Lisp implementation and its facilities, the provided `loop` macro and the Common Lisp Object System (abbr. CLOS) to *compile* his implementation of the `loop` macro into Lisp code that, when compiled, does not utilize any of the original `loop` macro or the CLOS facilities - it becomes simple Lisp code that utilizes mostly `tagbody` and `go` - Lisp version of the `goto` instruction known from many other languages.

Then he pointed out that once this new `loop` macro is compiled, it is no longer required to have CLOS or a different `loop` implementation on the new system. The whole `loop` is compiled into a form that only requires implementation of basic Lisp special forms: the aforementioned `tagbody` and `go`, conditionals like `cond` or `if` and basic mutation operators such as `setq`, `rplaca` or `nconc` - and only these basic parts of a Lisp system need to be implemented on the target system.

Let me use JSCL[8] as an example of a JavaScript implementation that lacks a `loop` macro. Given what Robert told me, it is posssible for it to have its own `loop` macro bootstrapped off an existing Common Lisp system with only very basic Common Lisp special operators implemented in JavaScript, which is notable on its own a very big chance of easing its completion.

At the time of writing this document, JSCL still lacks implementations of `loop`, `format` and CLOS. Through aforementioned bootstrapping, I consider this a chance to implement modern and extensible versions of these complicated functions within a new and fertile ground that is Common Lisp running on JavaScript platforms; as the world slowly consolidates its client-side computing towards web browsers and, therefore, JavaScript as the main scripting language of browsers, this makes it possible for programmers to program their code that targets web browsers in Common Lisp. As I am currently gaining experience in JavaScript, I hope that in the future I will be able to complete JSCL this way or aid someone with achieving the same.

3 Afterword

On behalf of me and Kamil Ziemian, I would like to thank the authorities of the Faculty of Mathematics and Computer Science of Jagiellonian University, the funders of our participation in the conference.

I would also like to thank the organizers of the European Lisp Symposium, as it would obviously be impossible to write this document without their wonderful work.

Many thanks also go to professor Christian Schafmeister and professor Robert Strandh for the aforementioned insight and knowledge they shared with me and the support I feel right now as I decide to embark on working with Lisp as my language of choice.

[8] `https://github.com/davazp/jscl`

My thanks also go to all the people who attended the ELS, the `#lisp` and `#lisp-pl` communities, the `kraklisp` group and the SLiMaK and KSI students' associations for their wonderful atmosphere of cooperation and friendliness, which allows me to participate in their daily life with comfort and relaxation.

Nina Palińska of Pedagogical University, Cracow, deserves applause for her wonderful singing during the welcome reception of the ELS held at Sunday.

Finally, I would like to thank two of my teachers, dr hab. Daniel Wilczak and dr Włodzimierz Moczurad from the aforementioned Faculty of Mathematics and Computer Science of Jagiellonian University. Dr hab. Daniel Wilczak was the first person to show me the basics of Scheme and to light up my interest in Lisps, while, thanks to dr Włodzimierz Moczurad, I gained insight into the functioning of Lisps as I was able to implement a very simple and bug-ridden Lisp in Haskell as an assignment for the functional programming course.

And one more thank to one certain figment of my imagination that appeared within my mind one day and, since then, gives me ideas and motivation necessary to continue going forwards.

References

Available in the proceedings for the 8th European Lisp Symposium[9]:

[1] *Christian Schafmeister. Clasp - A Common Lisp that Interoperates with C++ and Uses the LLVM Backend. Temple University, USA, 2015.*

Available in the proceedings for the 9th European Lisp Symposium[10]:

[2] *Christian Schafmeister. CANDO - A Common Lisp based Programming Language for Computer-Aided Nanomaterial Design and Optimization. Temple University, USA, 2016.*

[3] *Robert Strandh. A Modern Implementation Of The LOOP Macro. University of Bordeaux, France, 2016.*

[9] `http://www.european-lisp-symposium.org/editions/2015/ELS2015.pdf`

[10] `http://dept-info.labri.fr/~idurand/ELS/ELS2016/2016.pdf`