The Medley Interlisp Revival

Andrew Sengul
andrew@interlisp.org
Interlisp.org

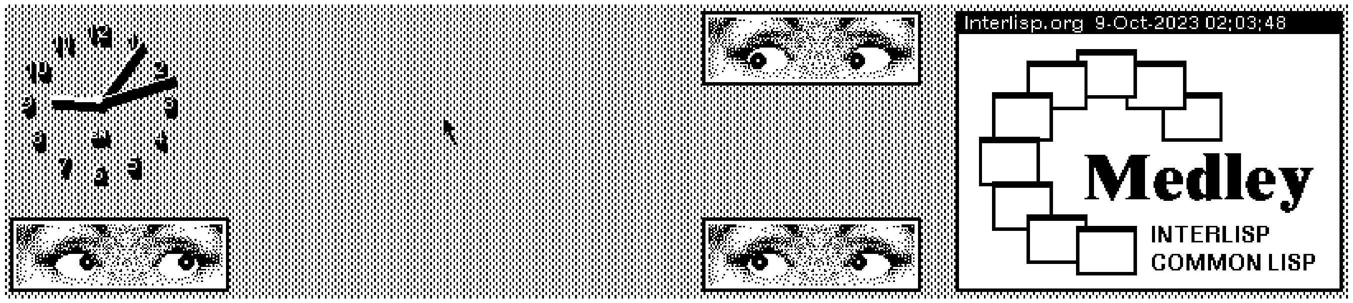


Figure 1: A screenshot from Medley featuring compact graphical software applications.

ABSTRACT

The Medley Interlisp revival is a project to restore Medley Interlisp for use on modern computers. Interlisp began as a Lisp environment for researchers sponsored by DARPA, and after gaining display capabilities it was renamed Interlisp-D. Xerox spun out sales and development of Interlisp-D with the "Medley" software release, which eventually became the product name. Medley development ended in the 1990s and was revived in 2021 by a team including some of the original PARC developers. Their effort is aimed at both preserving the Interlisp software created in the past and expanding the scope of what these tools can do to further realize the promises of interactive, graphically augmented development.

CCS CONCEPTS

• **Software and its engineering** → **Open source model**; **Software evolution**; *Maintaining software*; *Documentation*; Software reverse engineering; • **Theory of computation** → **Interactive computation**; • **Human-centered computing** → *Interaction design theory, concepts and paradigms.*

KEYWORDS

Lisp, Interlisp, GUI, interactive programming, software archaeology, software restoration, education, nonprofit, FOSS

ACM Reference Format:
Andrew Sengul. 2024. The Medley Interlisp Revival. In *Proceedings of the 17th European Lisp Symposium (ELS'24).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.5281/zenodo.11090093

1 INTRODUCTION

Interlisp.org, a US non-profit organization, is working to resurrect and restore the Medley Interlisp system and has made progress in modernizing and enhancing it. Medley Interlisp was the last version of the Interlisp system developed by Xerox Palo Alto Research Center (PARC). Accomplishments thus far have included reducing barriers to entry, making it easier to build versions for modern computing platforms, the creation of an online Medley system usable through a web browser, and one-click installers for major operating systems. The Medley software has been released under a free open source, downloadable at https://interlisp.org.

The system's original release came near the end of a fruitful period for interactive software development stretching from the late 1960s through the early 1990s. In the mid-1980s, Xerox PARC tapered off development of Medley and moved the project to Xerox AI Systems (a Xerox subsidiary). In the late 1980s XAIS moved the system to Envos, which closed shortly thereafter and led to a company called Venue acquiring the rights to Medley Interlisp. Circa 2018, Ron Kaplan and Nick Briggs resurrected Medley Interlisp and began transporting it to modern computing platforms to support work in natural language research.

In 2020, the Medley Project was formed with the goal of modernizing the Interlisp ecosystem, opening the way for present-day developers to experience it on modern computing platforms like Windows, MacOS and Linux. Interlisp.org was formed to organize these efforts, provide versions of Medley Interlisp to interested users and provide a repository for source code and documentation. Interlisp.org has been successful in this endeavor as Medley Interlisp now runs on the indicated platforms as well as through a browser-accessible online interface and on ARM systems. A Docker container release is available for added portability. Interlisp.org has also resurrected several applications including ROOMS, Notecards, LOOPS and others. It has assembled an online Zotero repository collecting Lisp documentation along with Interlisp papers, books, and technical material. Selected source code is also available for contributed programs.

2 THE PROJECT

Interlisp.org received the source code for Medley Interlisp and its applications from Venue Corporation. Interlisp.org consists of a group of volunteers – original developers, former users, computing historians interested in software preservation, and people interested in software archaeology. This group has focused on:

- Modernizing Medley's infrastructure and source code;
- Adapting the system to run on modern platforms;
- Reducing the barrier to entry for new users;
- Resurrecting and restoring Medley Interlisp applications of historical interest; and
- Conducting outreach to potential users, students, software historians, and others to foster better understanding of how symbolic computing evolved.

Notable is Medley's support for two Lisp dialects: Common Lisp (CL) and Interlisp. These are implemented as compilers for the respective dialects supporting interaction through a read-evaluate-print loop, or REPL. While the functions in these dialects are implemented in different ways their data structures are identical, so numbers, symbols, linked lists and some other data types can be shared between the two dialects. This makes it possible to create blended applications in which data is passed between functions written in either dialect.

2.1 Adaptation to Modern Platforms

Interlisp.org is engaged in restoring the Medley Interlisp ecosystem, including tools, utilities, and applications, and provides public versions of source and binary code for modern computing platforms along with documentation. All of these artifacts are available through Interlisp.org's GitHub and Zotero repositories. The open source version for modern computing platforms consists of:

- Maiko: the emulation software that implements the Interlisp Virtual Machine;
- Medley: source and compiled versions of Medley Interlisp, its tools and utilities, and selected applications;
- Installers for modern computing platforms: Windows 10+, MacOS and Linux, including WSL and ARM; and
- https://online.interlisp.org: a browser-accessible online version of Medley Interlisp.

Additionally, Interlisp.org provides public access to its Zotero repository, which contains a collection of documentation for Interlisp and other Lisp dialects.

2.2 Reducing Barriers to Entry

Easy access to Medley Interlisp will help (re)introduce potential users to Interlisp in a way that allows them to explore its features while building their expertise. Since Interlisp was developed before current conventions for mouse and window-based interaction as well as Unicode standards for text encoding, Medley has been modernized to give users a look and feel that will be more familiar in the context of today's computer interfaces.

Interlisp.org created an online version of Medley Interlisp using Docker and Amazon Web Services, providing for users to experience the system through a web browser without running any of the Medley system's components on their local computing platform. When they are ready, if they choose they can install a version of Medley Interlisp onto a local computer system and download files created using the online platform to be used on the same system.

During 2023, online Interlisp had 428 registered users accounting for 1,685 sessions, along with 2,588 anonymous guest sessions.

3 INTERLISP IN PERSPECTIVE

The history of Interlisp is interwoven with the early history of artificial intelligence as many AI researchers had access to DEC PDP-10/DECsystem-10 computers which could run relatively large programs on a platform with 256K words of real memory [1]. Interlisp provided a residential programming environment in which the software development functions of edit, compile, link, and execute could be performed without leaving the Interlisp environment [16], along with a suite of tools for writing documentation.

In Interlisp, a user edits and evaluates Lisp objects that reside in an image in memory. The code is saved to files that are more like code databases than traditional source files. Users do not modify the files but load their contents into memory to edit, compile, and execute. The File Manager provides a simple interface that saves code in a memory image to disk, along with archiving unsaved code changes when a user closes a session so unfinished work can be resumed later. The beginning of a file specifies metadata describing the "file environment" and readtable associated with the file. The File Manager coordinates the development tools and readable code management tasks. It notices the changes to Lisp objects edited with SEdit or manipulated in memory, tracks what changed functions and objects need to be saved to symbolic files, and carries out the actions for building programs such as compiling or listing them (working in some ways like Unix's make).

This model is unique even today, with a few programming systems like Smalltalk and Symbolics Genera offering similar capabilities but nothing exactly matching Medley's model. Although Interlisp was not the first version of Lisp (there were several on mainframes and early minicomputers), it is safe to say that it influenced most succeeding versions. Medley's development coincided with the creation of the Common Lisp standard and its design influenced decisions by the CL committee, which included Medley developer Larry Masinter. Records of their thought process can be found in their email threads [8]. Interlisp was implemented on top of a virtual machine, preserving vertical integration, and many of its utilities were also written in Interlisp. Conversely, most other Lisp systems were written in imperative programming languages like C for performance reasons.

3.1 A Model of Interactive Software Development

Medley Interlisp was created to provide a computing environment for research into and development of large-scale applications. To this end, Xerox PARC users as well as others developed tools to facilitate software development through an interactive graphical user interface (GUI) in a collaborative environment [6]. Among these utilities were MasterScope, Spy, DWIM, CLISP, and LOOPS. Over 100 such tools are collected in the LispUsers library of contributed software, some of which have yet to appear in modern software development toolsets.

Numerous utilities were developed at institutions apart from Xerox PARC, such as NASA, several DARPA contractors and commercial firms. We are searching to identify these tools, acquire source code where possible, adapt them to run on modern computing platforms and make them available along with appropriate licenses and documentation through the GitHub and Zotero repositories.

4 MEDLEY INTERLISP ECOSYSTEM

Components of the Medley Interlisp Ecosystem include:

- Maiko: the emulator software for the Interlisp virtual machine;
- Medley Interlisp: the Interlisp source code and its utilities and tools;
- Applications: including several Interlisp applications, such as ROOMS, Notecards, LFG, STRADS, IDA, as well as other Lisp system applications; and
- Documentation: a comprehensive collection of books, papers, technical memoranda, and manuals regarding Interlisp, and extending to other Lisp variants.

Interlisp.org welcomes contributions of source code and documentation to add to its repositories. The following sections briefly describe these components and work done to restore them.

4.1 Maiko

Maiko is the emulator implementing the virtual machine within which Interlisp runs.[1] Maiko was initially developed by Fuji Xerox but acquired by Xerox PARC, which continued to maintain and enhance it. Written in Kernighan and Ritchie C, Interlisp.org developers have modernized it, making it ANSI C compatible. A few of these modifications included resolving issues of signed vs. unsigned characters, adding prototypes for functions, ensuring all parameters have types, fixing some incorrect translations of Lisp code to C code and optimizing all virtual machines' opcodes.

A major goal of this modernization process was to facilitate moving the Medley Interlisp ecosystem to modern computing platforms. Numerous programming changes were made to ensure that Maiko could run on Windows 10/11, recent MacOS versions and Linux and WSL-based platforms, as well as in the form of a hosted public installation accessible through web browsers.

4.1.1 Running Natively on Windows 10/11. Previously, running on Windows required the use of the Medley Docker container or WSL. Both involved significant effort to set up along with knowledge not possessed by most Windows users. Native support was developed using Cygwin and SDL2, allowing the use of a one-click installer in the form of an `.exe` file.

4.1.2 Support for AArch64. The build scripts for the Maiko virtual machine were extended to support the AArch64 (ARM) platform. This effort established a model for generating build scripts for other platforms, such that any system which has an ANSI C compiler can host a version of Medley Interlisp.

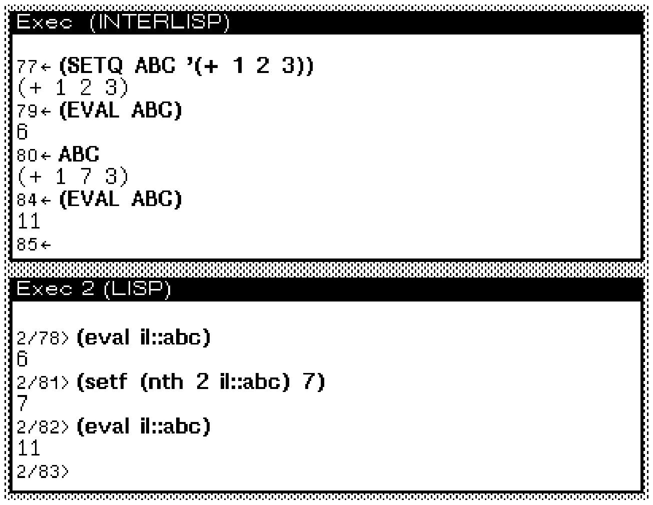
4.1.3 Major Platform Installers. Installing Medley was formerly a multi-step process requiring a degree of expertise with the administrative tools of a given platform. A single-step installer using a "one-click" approach was developed for MacOS, Windows (native), Windows running WSL and Cygwin and many Linux distributions, allowing a user to quickly and easily install a Medley release and do meaningful work.

[1] https://github.com/Interlisp/maiko

4.2 Medley Interlisp

Medley Interlisp includes the basic functions implementing the Interlisp language and environment as well as a selection of development tools. Because these utilities were written in Interlisp, many were found to run without major changes once the basic system became operational.

4.2.1 Common Lisp Support. With the groundswell of support for Common Lisp in the mid-1980s, Xerox PARC extended the Interlisp environment to support Common Lisp, specifically as of Common Lisp: The Language, Version 1 (CLtL1), along with CLOS and CL's condition system. The infrastructure supporting Interlisp and Common Lisp is fully integrated such that functions from both dialects are available within the same system in separate software packages, albeit with some rough edges we are working out.



```
77← (SETQ ABC '(+ 1 2 3))
(+ 1 2 3)
79← (EVAL ABC)
6
80← ABC
(+ 1 7 3)
84← (EVAL ABC)
11
85←
```

```
2/78> (eval il::abc)
6
2/81> (setf (nth 2 il::abc) 7)
7
2/82> (eval il::abc)
11
2/83>
```

Figure 2: Code evaluated in both Interlisp and Common Lisp addressing a common data structure.

For example, Figure 2 shows two Exec windows – one using the Interlisp readtable addressed through the package `IL:` and one using the Common Lisp readtable addressed through the package `CL:`. The pictured code creates a list in Interlisp, modifies it in Common Lisp and evaluates it in both dialects.

The Common Lisp integration with the Interlisp tools was incomplete. Substantial work has made it easier to use some of the Interlisp tools like HELPSYS and MasterScope with CL. Also, some of the functions and directives in CLtL2 are not available (such as 'declaim' versus 'proclaim'). As we discover these we are determining how to provide the missing functionality, but some might not be available until later in 2024.

4.2.2 Editing and Browsing Support. Since the early 1980s Medley Interlisp has used a 16-bit internal representation of characters in strings and atoms in the form of Xerox XCCS codes [9], which were mapped into appropriate glyphs for display and printing. We have since generalized character reading/writing functions as part of our external formatting project. If a file's external format is specified as Unicode when a stream is opened, Unicode byte sequences are read into 16-bit Unicode codes, which are translated into their XCCS equivalents before being delivered to the calling function. Support for ISO8859 and certain Japanese conventions are also provided. This system removes the need for most programmers using Medley to directly deal with character encoding.

TEdit, the text editor, was extended with Unicode support in a way providing for better efficiency, reliability, and maintainability of the system [14]. TEdit reads all characters into an internal editing buffer and creates pointers to the bytes on the file that represent those characters. It only interprets those bytes when it needs to display that section of the file, move characters from one place to another in the file, or copy them to some other application. Thus a TEdit session on a large file opens quickly and only occupies a small amount of memory.

Work on TEdit has encompassed a major portion of the modernization effort over the past three years because assumptions about the XCCS file format were threaded all through the core TEdit implementation. Every location of XCCS code usage had to be tracked down and this revealed a variety of bugs, inconsistent behaviors and maintainability issues, which required substantial refactoring. As of Spring 2024 this work is finally coming to the end, bringing significantly greater robustness and reliability to the Medley Interlisp system.

Additional changes allowed the ingestion of Xerox Alto Bravo-format files, making it possible for legacy documents to be converted to PDF through invocation of an external converter. Also, HELPSYS was extended to allow lookup and display of the Common Lisp Hyperspec and other Medley documentation.

4.2.3 UnixUtils. Medley was enhanced to allow it to reach out to the host environment platform to accomplish system-level tasks that are not available in Medley. These include ShellBrowser, which opens a URL in the specified browser, and ShellOpen, which opens a host-resident viewer for a specified file.

4.2.4 PDFStream. Medley incorporated a native imagestream implementation for producing PostScript™ hardcopy files. The PDF format is not supported as it was not yet invented when this system was developed, but an interim PDF solution from 2023 allows creating a PS file and executing a UnixUtils shell script to convert it to PDF via Ghostscript's ps2pdf utility. Medley's FileBrowser was extended to automatically open PDF files in a separate window using a host-resident PDF viewer.

4.2.5 Github Integration. Github is being used to manage the coordination of multiple developers across several time zones and countries in extending Maiko, Medley, and the tools and utilities. A major effort was the integration of Github functions with the Interlisp File Manager. GITFNS is a set of functions that includes a menu-driven interface to compare Lisp source files on a function-by-function basis, supporting Interlisp's characteristic "residential" approach to development.

4.2.6 Mouse and Keyboard Usage. Originally, Interlisp supported the three-button mice available with many Xerox computers. These have largely disappeared so the interface APIs have been extended to support mice with two buttons (as often used with Windows), one button (as with Macs) or a touchpad. Function keys on several popular keyboards have been mapped to codes emitted by Xerox-type mice to allow access to the original functionality; a "meta" key allows emulation of the middle mouse button of a three-button mouse. Work is ongoing to implement more scroll wheel and middle mouse button functionality.

Only a few keyboard models were available when Interlisp was developed and every application had a unique way of associating input keycodes with program functions. This is impossible now; users want uniform treatment, no matter what keyboard they use. Interlisp.org is working to broadly organize keyboard encoding and communication because keyboard handling is buried in several different locations within Maiko and Medley. Different utilities interpret certain keystrokes in a variety of ways.

Our goal is to build a unified keyboard model that will accommodate a large number of commercial keyboards and unify keyboard handling across tools and applications. This is likely to involve translators interpreting "niche" keyboard types as more common keyboards and/or translators from actual keycodes to an internal model. We expect this to increase the speed with which we can port Medley to different computing platforms.

4.3 Applications

Numerous applications have been built using versions of Interlisp including many early AI tools as well as ROOMS[4], NoteCards[15], a workbench for writing LFG grammars[7], Intelligent Database Assistant (IDA), LOOPS, and the Strategic Automated Discovery System (STRADS) [5]. Recently, an affiliate of Interlisp.org discovered an archive of the Stanford AI Laboratory containing sources for Doug Lenat's AM and Eurisko programs[2], written in an early version of Interlisp. Interlisp.org has demonstrated they can be loaded into a Medley Interlisp environment with minimal changes and are documenting steps to make them usable again.

Interlisp.org is also collecting open source Common Lisp programs and using them to test the Medley Common Lisp implementation, with some changes to make them easier to use, and providing them through our GitHub repository for public use. Some of these applications include ATMS, BB1 and NIKL. Work will continue throughout this year to get them running in Medley reliably and provide minimal documentation (or more, if possible).

4.4 LOOPS

The Lisp Object-Oriented Programming System (LOOPS) is unique among programming systems in that it combines four different paradigms for software development:

- Imperative/Functional Programming
- Object-Oriented Programming
- Aspect-Oriented Programming
- Rule-Based Programming

[2]https://white-flame.com/am-eurisko.html

The integration of these paradigms provides the software architect and developer with a comprehensive toolkit for building large applications [2], allowing for the choice of a data representation and problem solving approach that best meets the needs of a given application. The Medley Interlisp tools and utilities were extended to operate with the LOOPS constructs seamlessly, and Medley's interface tools allow the creation of graphical displays reflecting the values of variables to which they are attached.

Researchers at PARC developed the Truckin' game to help users understand how to program in a multiparadigm environment and visualize what was happening as the game evolved [13]. Truckin' simulated the activity of truck drivers working to make deliveries on time, accounting for geography, varying types of goods and the need to refuel [12]. LOOPS is documented in three books that address the Basic System, the Tools and Utilities, and the Rule-Based System; the latter volume details Truckin' and its development.

4.5 Documentation

Interlisp.org has access to a wide variety of documentation about Interlisp and Common Lisp, including original Xerox PARC manuals, memoranda and program and application documentation from the Computer History Museum's PARC archive. Much of this documentation was written by the original developers who already knew how to use the system and can be obtuse for new users. Interlisp.org has released additional volumes on the usage of Medley Interlisp and LOOPS since 2021, all of which are available at Interlisp.org. These include:

- Interlisp: The Language and its Usage
- Medley Interlisp: The Interactive Programming Environment
- Medley Interlisp: Interactive Programming Tools
- LOOPS Volume I: The Basic System †
- LOOPS Volume II: Tools & Utilities †
- LOOPS Volume III: Rule-Based Systems ††

More work documenting Interlisp and Common Lisp applications is planned for the next two years, with a focus on supporting new users.

4.5.1 Community Outreach. We revamped the Interlisp.org website over the past year to make it easier to navigate, offering visitors an array of options to support further Medley Interlisp development. The website provides access to most of the material that Interlisp.org has collected, with more information being added as we locate sources. Our collection of new and recovered documents extends beyond the website to the GitHub and Zotero repositories.

Interlisp.org continues its outreach to the broader Lisp and computer science community through technical presentations. Three talks were presented in 2023:

- BALISP: In March 2023, the project's efforts were presented to the Bay Area Lisp meetup group. The slides are available on the project's Google Drive[3] and the talk on Youtube[4].
- Software Preservation Network (SPN): On Nov. 2, 2023 Larry Masinter presented to the SPN Idea's Workshop technical details of our work as well as suggesting future collaborative projects across the community.
- BCS Computer Conservation Society (CCS): Steve Kaisler presented a talk entitled "Software Archaeology: The Medley Restoration Project" to the CCS Monthly Meeting on Nov. 16, 2023 in London, England. It included a brief history of Interlisp, a review of some applications, and discussion of challenges in in modernizing Medley Interlisp (some of which have been presented in this paper).

Articles on the Medley Interlisp project have appeared in The Register [11], Hackaday [10] and Hacker News [3].

†Draft available at Interlisp.org

††In progress, forthcoming

[3]https://drive.google.com/file/d/1xpXSoEnc5PPnIa7BHcionBbc8v-Nxp7N/view?usp=sharing

[4]https://www.youtube.com/watch?v=N1MobfEaoWY

ACKNOWLEDGMENTS

We acknowledge the leadership and historical perspective of some of the original developers - Larry Masinter, Nick Briggs, Frank Halasz, Ron Kaplan and the other members of Interlisp.org. We also acknowledge the estate of John Sybalsky for granting rights to use and distribute the source code for Medley Interlisp, and Eric Kaltman and his students at UC Channel Islands for efforts in organizing the Zotero repository. Funding for this effort has been provided by members of Interlisp.org.

REFERENCES

[1] Daniel G. Bobrow and Bertram Raphael. New programming languages for artificial intelligence research. *ACM Comput. Surv.*, 6(3):153–174, sep 1974. ISSN 0360-0300. doi: 10.1145/356631.356632. URL https://doi.org/10.1145/356631.356632.

[2] Daniel G. Bobrow and Mark Stefik. *The LOOPS Manual.* Xerox Corporation, Palo Alto, CA, USA, 1983.

[3] Paolo Amoroso et al. My encounter with Medley Interlisp., January 2023. URL https://news.ycombinator.com/item?id=34300806.

[4] D. Austin Henderson and Stuart Card. Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Trans. Graph.*, 5(3):211–243, Jul 1986. ISSN 0730-0301. doi: 10.1145/24054.24056. URL https://doi.org/10.1145/24054.24056.

[5] Stephen H. Kaisler. A knowledge based system for geopolitical analysis. In *Proceedings of the 57th MORS Symposium.* Military Operations Research Society, Jun 1989.

[6] Stephen H. Kaisler. *Medley Interlisp: The Interactive Programming Environment.* Interlisp.org, Palo Alto, CA, USA, 2021.

[7] Ronald M. Kaplan and John T. Maxwell. *LFG Grammar Writer's Workbench.* Xerox Corporation, Palo Alto, CA, USA, Mar 2003.

[8] David Moon, Kent M. Pitman, Larry Masinter, Brad Miller, Scott Fahlman, Warren Harris, and Jon L White. Issue: Eval-other (version 1), 1988. URL https://github.com/masinter/parcftp-cl/blob/main/cl/cleanup/old-mail/eval-other.mail.

[9] Greg Nuyens. *Font/Character documentation.* Xerox Corporation, Palo Alto, CA, Mar 1986.

[10] Maya Posch. Reviving Interlisp with the Medley Interlisp project, July 2023. URL https://hackaday.com/2023/07/09/reviving-interlisp-with-the-medley-interlisp-project/.

[11] Liam Proven. Revival of Medley/Interlisp: Elegant weapon for a more civilized age sharpened up again, November 2032. URL https://www.theregister.com/2023/11/23/medley_interlisp_revival/.

[12] Mark Stefik. Truckin' and the knowledge competitions, 2017. URL https://www.markstefik.com/?page_id=359.

[13] Mark Stefik, Daniel G. Bobrow, Sanjay Mittal, and Lynn Conway. Knowledge programming in loops: Report on an experimental course. *The AI Magazine*, pages 3–13, 1983.

[14] Xerox Artificial Intelligence Systems. *Interlisp-D: A Friendly Primer.* Xerox Corporation, Pasadena, CA, USA, Nov 1986.

[15] Xerox Special Information Systems. *NoteCards™ Release 1.2 Reference Manual.* Xerox Corporation, Pasadena, CA, USA, Apr 1985.

[16] Warren Teitelman. Interlisp. *SIGART Bull.*, page 8–9, dec 1973. ISSN 0163-5719. doi: 10.1145/1056786.1056787. URL https://doi.org/10.1145/1056786.1056787.