

Lightning Talks II

ELS 2016

- **01 ILLITHID - *Mark Evenson***
- 02 Sugaring Lisp for the 21st Century - *Vsevolod Dyomkin*
- 03 Common Lisp UltraSpec - *Michał Herda*
- 04 PRIMSMA - *Philip Gagner*
- 05 Managing Packages - *James Anderson*
- 06 CEPL Reaches Beta - *Chris Bagley*
- 07 QueryFS: a Solution Looking for More Problems - *Michael Raskin*
- 08 LISP/c - *Michał Herda (on behalf of Jonathan Baca)*
- 09 Electricity is Orange - *Devon McCullough*

MARK <EVENSON.NOT.ORG@GMAIL.COM>
ELS 2016 / KRAKOW, POLAND

**ILLITHID: A REPORT ON USING GOING
INTO PRODUCTION WITH ABCL**

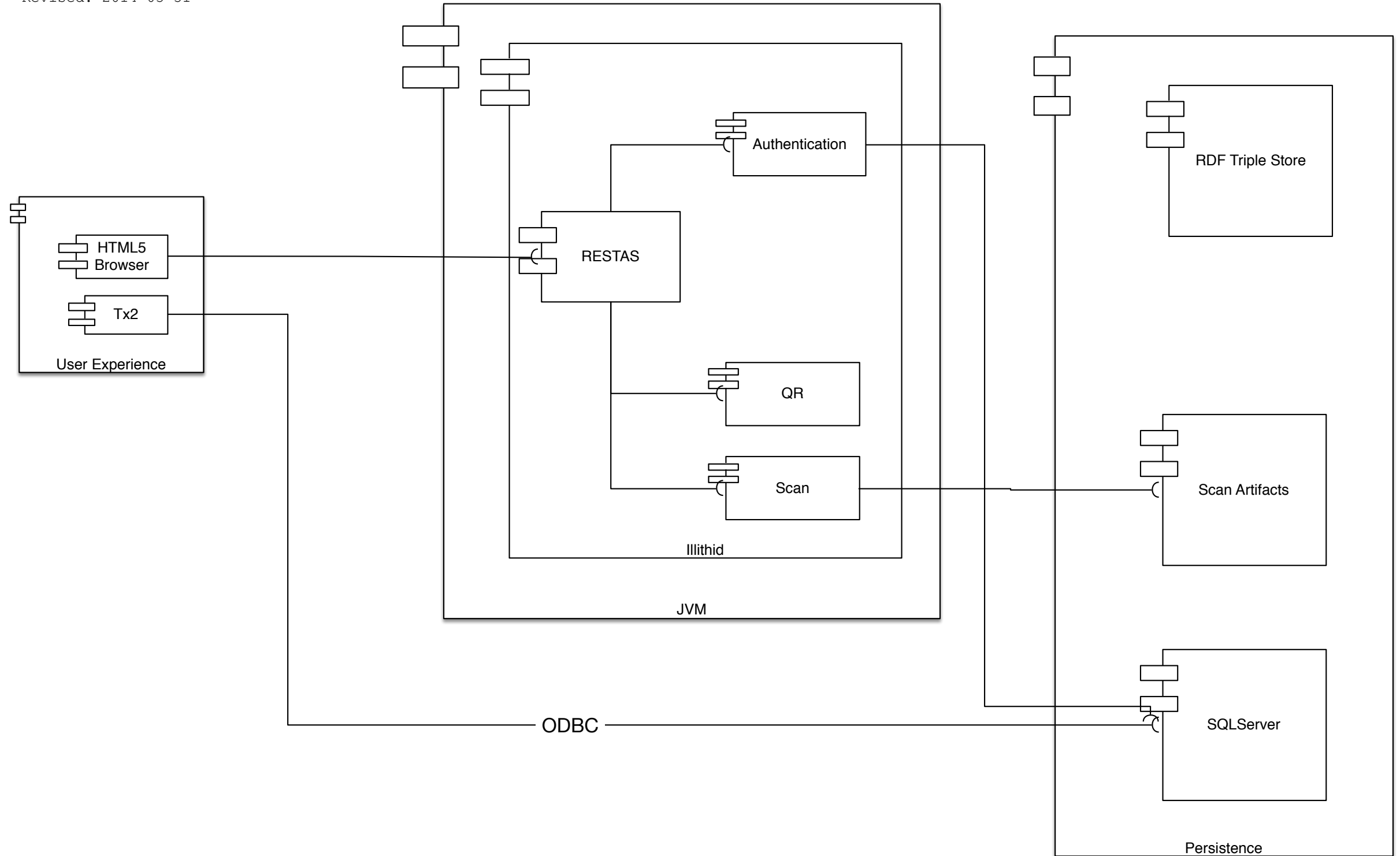
ILLITHID

- ▶ (“Picasso”) Electronic Health Record system
 - ~20 year old codebase University at Buffalo School of Dental Medicine
 - used by ~1000 Students ~600 Dentists
 - In-house custom software based on PowerBuilder (4GL with SQL persistence)
 - Application has both record keeping/pedagogical (grading)
- ▶ (“Pablo”) Incremental “Brownfield” project begun in Fall 2013; first production in May 2014
 - ▶ Move to HTML5 client (while obeying HIPAA regulation)
 - ▶ Move to use of OWL2 Ontologies for data description
 - ▶ 1) To facilitate research across anonymized data sets
 - ▶ 2) To enable interoperability with other medical data systems

- ▶ Illithid: “REST Broker” built on ABCL
 - ▶ QuickLisp: Hunchentoot, RESTAS, CL-WHO, ParenScript, LParallel, CXML, XPath
Total of around ~60 ASDF systems
 - ▶ Java Libraries: Render to PDF, SQL connectivity
 - ▶ LSW2 (Alan Ruttenberg <https://github.com/alanruttenberg/lsw2>)
OWL2-based Reasoner; data persisted as RDF
 - ▶ Ontotext GraphDB (OWLIM)
 - ▶ Microsoft SQL Server

ILLITHID

UBSDM "Pablo" Picasso
Mark Evenson
Created: 2014-03-31
Revised: 2014-03-31



- ▶ Problem

Need a simplified deployment strategy for Microsoft Windows because system operators “Don’t want to know about Lisp” (i.e. no management via REPL)

- ▶ Solution

Use Java Servlet Container (Tomcat) instances that manages the application lifecycle (start, stop, logging, deployment descriptor for database location)

- ▶ Need to provide a single Java deployment artifact (“Web Application Archive”) to system operators

- ▶ ABCL-SERVLET <<https://bitbucket.org/easye/abcl-servlet>>

Scaffolding for a Java Servlet that boots ABCL, and executes configurable Lisp Code

Creating Java Deployment Artifact 1/2

- ▶ All software components encapsulated by ASDF
- ▶ (ABCL-ASDF) ABCL extension to ASDF uses Maven to describe Java software library dependent which downloads the Java dependencies upon first run

```
(:module maven  
  
  :components  
  
  ( (:mvn "org.apache.xmlgraphics/fop/1.0")  
    (:mvn "org.apache.pdfbox/pdfbox/1.8.4")
```


Creating Java Deployment Artifact 2/2

- ▶ (ASDF-JAR) ABCCL extension to locate ASDF systems within the deployment artifact
 - ▶ Simple packaging strategy:
Assuming that all ASDF systems have a "root" description file, locate the ASDF root on the filesystem, recursively copy all components (including Quicklisp)
- ▶ Result: 108 Mib "illithid.war" deployment artifact
- ▶ Bonus: able to connect to production application via Swank



- 01 ILLITHID - *Mark Evenson*
- **02 Sugaring Lisp for the 21st Century - *Vsevolod Dyomkin***
- 03 Common Lisp UltraSpec - *Michał Herda*
- 04 PRIMSMA - *Philip Gagner*
- 05 Managing Packages - *James Anderson*
- 06 CEPL Reaches Beta - *Chris Bagley*
- 07 QueryFS: a Solution Looking for More Problems - *Michael Raskin*
- 08 LISP/c - *Michał Herda (on behalf of Jonathan Baca)*
- 09 Electricity is Orange - *Devon McCullough*

Sugaring Lisp for the 21st Century

Vsevolod Dyomkin
@vseloved

European Lisp Symposium
2016-05-09

Common Complaints about Common Lisp

(from the POV of outsiders)

- ~~* No libraries~~
 - ~~* No threads, sockets,
— whatever in the standard~~
 - * “Historical cruft” (poor names, verbosity, inconsistency, lack of modularity, not generic enough etc.)
-

“Modernization” Efforts

* CDR <https://common-lisp.net/project/cdr/>
Common Lisp Document Repository

* clt13 <http://ilc2009.scheming.org/node/48>
“Codifying Modern Common Lisp”

* cl21 <https://github.com/cl21/cl21>
“Common Lisp in the 21st Century”

Beyond Utilities

Default solution – utilities
(<http://cliki.net/utilities> - 36
entries of which at least 10+ are
general-purpose ones).

Most don't go far enough – added
value is only convenience, not
language evolution and growth

RUTILS

est. 2009

14 core + 7 contrib packages

~ 250 exported symbols

(cf. ~1000 symbols in package CL)

* backward-compatible

* practical

* modular

* pro-choice

<https://github.com/vseloved/rutils>

Example Code

```
(defmethod select-transition ((parser amrparser) graph)
  (with (((stack buffer) @ parser))
    (s0 (? stack 0))
    (b0 (? buffer 0)))
  (flat-map (lambda (k)
    (unless (or (in# (pair k b0) graph)
                (in# (pair b0 k) graph))
      (let ((fs (append common-fs
                        (extract-fs parser
                                   nil s0 b0))))
        (mapcar ^ (make-trans :fn % :fs fs)
                 '(amr:reattach
                   amr:reenter))))))
    (set-difference @parser.tokens
                    (list nil s0 b0))))))
```

```
CL-USER> (select-transition (make 'parser) #h(equal))
```

How You Can Benefit

* Use it

;; conventional way

```
(ql:quickload :rutils)
```

```
(use-package :rutils)
```

```
(named-readtables:in-readtable rutils-readtable)
```

;; radical way

```
(ql:quickload :rutilsx)
```

```
(use-package :rutilsx)
```

```
(named-readtables:in-readtable rutilsx-readtable)
```

How You Can Benefit

- * Use parts of it
(`use-package :rutilsx.threading`)
 - * Borrow ideas ...like `cl21` did
usually called cross-pollination
-

Open Issues

How to gracefully handle:

- * map/mapcar
 - * slot-value names across packages in @obj.slot
-



- 01 ILLITHID - *Mark Evenson*
- 02 Sugaring Lisp for the 21st Century - *Vsevolod Dyomkin*
- **03 Common Lisp UltraSpec - *Michał Herda***
- 04 PRIMSMA - *Philip Gagner*
- 05 Managing Packages - *James Anderson*
- 06 CEPL Reaches Beta - *Chris Bagley*
- 07 QueryFS: a Solution Looking for More Problems - *Michael Raskin*
- 08 LISP/c - *Michał Herda (on behalf of Jonathan Baca)*
- 09 Electricity is Orange - *Devon McCullough*

Yet Another Rant About The State Of Common Lisp Documentation

(save-lisp-and-die "secret-alien-technology.core")

Michał „phoe” Herda

LispWorks



Common Lisp HyperSpec™

The very definition of class.

Welcome to the *Common Lisp HyperSpec*.
I hope it serves your need.

--[Kent Pitman](#), X3J13 Project Editor



Here are some useful [starting points](#):

<u>* Highlights</u>	Contents..... Chapter 1 Chapter 2	Master Index	M N	Symbol Index	S T	Glossary, <i>n.</i> Index of terms.	x3j13 issues
---------------------	---	-----------------	--------	-----------------	--------	--	-----------------



A [text-only version of this cover sheet](#) is available.

Copyright 1996-2005, LispWorks Ltd. All Rights Reserved.



clhs array



Wszystko

Grafika

Mapy

Filmy

Wiadomości

Więcej ▾

Narzędzia wyszukiwania

Okolo 6 490 wyników (0,17 s)

CLHS: Function MAKE-ARRAY - LispWorks

www.lispworks.com/.../lw51/CLHS/.../f_mk_ar.htm ▾ Tłumaczenie strony

Syntax: make-**array** dimensions &key element-type initial-element initial-contents adjustable fill-pointer displaced-to displaced-index-offset. => new-**array**.

CLHS: Section The Arrays Dictionary - LispWorks

www.lispworks.com/documentation/.../c_arrays.htm ▾ Tłumaczenie strony

15.2 The **Arrays** Dictionary. System Class **ARRAY** · Type **SIMPLE-ARRAY** · System Class **VECTOR** · Type **SIMPLE-VECTOR** · System Class **BIT-VECTOR**.

CLHS: System Class ARRAY

clhs.lisp.se/Body/t_array.htm ▾ Tłumaczenie strony

An **array** contains objects arranged according to a Cartesian coordinate system. An **array** provides mappings from a set of fixnums $\{i_0, i_1, \dots, i_{r-1}\}$ to corresponding ...

Ta strona była przez Ciebie odwiedzana.

CLHS: Function ADJUST-ARRAY

clhs.lisp.se/Body/f_adjust.htm ▾ Tłumaczenie strony

Syntax: adjust-**array** **array** new-dimensions &key element-type initial-element initial-contents fill-pointer displaced-to displaced-index-offset. => adjusted-**array**.

Content

Chapter 1 (Introduction)	
1.1 Scope, Purpose, and History	1-
1.1.1 Scope and Purpose	1-
1.1.2 History	1-
1.2 Organization of the Document	1-
1.3 Referenced Publications	1-
1.4 Definitions	1-
1.4.1 Notational Conventions	1-
1.4.1.1 Font Key	1-
1.4.1.2 Modified BNF Syntax	1-
1.4.1.2.1 Splicing in Modified BNF Syntax	1-
1.4.1.2.2 Indirection in Modified BNF Syntax	1-
1.4.1.2.3 Additional Uses for Indirect Definitions in Modified BNF Syntax	1-
1.4.1.3 Special Symbols	1-
1.4.1.4 Objects with Multiple Notations	1-1
1.4.1.4.1 Case in Symbols	1-1
1.4.1.4.2 Numbers	1-1
1.4.1.4.3 Use of the Dot Character	1-1
1.4.1.4.4 NIL	1-1
1.4.1.5 Designators	1-1
1.4.1.6 Nonsense Words	1-1
1.4.2 Error Terminology	1-1
1.4.3 Sections Not Formally Part Of This Standard	1-1
1.4.4 Interpreting Dictionary Entries	1-1
1.4.4.1 The "Affected By" Section of a Dictionary Entry	1-1
1.4.4.2 The "Arguments" Section of a Dictionary Entry	1-1
1.4.4.3 The "Arguments and Values" Section of a Dictionary Entry	1-1
1.4.4.4 The "Binding Types Affected" Section of a Dictionary Entry	1-1
1.4.4.5 The "Class Precedence List" Section of a Dictionary Entry	1-1
1.4.4.6 Dictionary Entries for Type Specifiers	1-1
1.4.4.6.1 The "Compound Type Specifier Kind" Section of a Dictionary Entry	1-1
1.4.4.6.2 The "Compound Type Specifier Syntax" Section of a Dictionary Entry	1-1
1.4.4.6.3 The "Compound Type Specifier Arguments" Section of a Dictionary Entry	1-1
1.4.4.6.4 The "Compound Type Specifier Description" Section of a Dictionary Entry	1-1
1.4.4.7 The "Constant Value" Section of a Dictionary Entry	1-1
1.4.4.8 The "Description" Section of a Dictionary Entry	1-1
1.4.4.9 The "Examples" Section of a Dictionary Entry	1-1
1.4.4.10 The "Exceptional Situations" Section of a Dictionary Entry	1-1
1.4.4.11 The "Initial Value" Section of a Dictionary Entry	1-1
1.4.4.12 The "Argument Precedence Order" Section of a Dictionary Entry	1-1
1.4.4.13 The "Method Signature" Section of a Dictionary Entry	1-1
1.4.4.14 The "Name" Section of a Dictionary Entry	1-1
1.4.4.15 The "Notes" Section of a Dictionary Entry	1-2
1.4.4.16 The "Pronunciation" Section of a Dictionary Entry	1-2
1.4.4.17 The "See Also" Section of a Dictionary Entry	1-2
1.4.4.18 The "Side Effects" Section of a Dictionary Entry	1-2
1.4.4.19 The "Supertypes" Section of a Dictionary Entry	1-2

Tilde Right-Bracket (format directive)	22-31
Tilde Right-Paren (format directive)	22-33
Tilde S (format directive)	22-26
Tilde Semicolon (format directive)	22-33
Tilde Slash (format directive)	22-28
Tilde T (format directive)	22-28
Tilde Tilde (format directive)	22-22
Tilde Underscore (format directive)	22-27
Tilde Vertical-Bar (format directive)	22-21
Tilde W (format directive)	22-27
Tilde X (format directive)	22-23
time	26-48
time	25-13
time zone	26-48
token	2-5, 26-48
top level form	26-48
trace	25-11
trace output	26-48
trace-output	21-47
translate-logical-pathname	19-28
translate-pathname	19-29
tree	14-1, 26-48
tree structure	26-48
tree-equal	14-14
true	26-48
truename	20-2, 26-48
truename	20-5
truncate	12-22
two-way stream	26-48
two-way-stream	21-8
two-way-stream-input-stream	21-40
two-way-stream-output-stream	21-40
type	26-48
type	3-74, 25-16, 25-17
type declaration	26-49
type equivalent	26-49
type expand	26-49
type specifier	26-49
type-error	4-33
type-error-datum	4-33
type-error-expected-type	4-33
type-of	4-30
typecase	5-62
typep	4-31
unbound	26-49
unbound variable	26-49
unbound-slot	7-74
unbound-slot-instance	7-74
unbound-variable	10-19
undefined consequences	1-15
undefined function	26-49
undefined-function	5-86
Underscore (format directive)	22-27
unexport	11-20
unintern	26-49
unintern	11-21
uninterned	26-49
union	14-50
universal time	25-3, 26-49
unless	5-59
unqualified method	26-49
unread-char	21-17
unregistered package	26-49
unsafe	1-14, 26-49
unsafe call	3-40, 26-49
unsigned-byte	12-16
:unspecific	19-5
unspecified consequences	1-15
unspecified values	1-15
untrace	25-11
unuse-package	11-23
unwind-protect	5-41
:up	19-6
update-instance-for-different-class	7-28
update-instance-for-redefined-class	7-29
upgrade	26-49
upgraded array element type	15-2, 26-50
upgraded complex part type	26-50
upgraded-array-element-type	15-24
upgraded-complex-part-type	12-50
upper-case-p	13-17
uppercase	26-50
use	26-50
use list	26-50
use-package	11-24
use-value	9-56
user	26-50
USER package	A-1
user-homedir-pathname	25-27
valid array dimension	26-50
valid array index	26-50
valid array row-major index	26-50
valid fill pointer	26-50
valid logical pathname host	26-50
valid pathname device	26-51
valid pathname directory	26-51
valid pathname host	26-51
valid pathname name	26-51
valid pathname type	26-51
valid pathname version	26-51
valid physical pathname host	26-51
valid sequence index	26-51
value	26-51
value cell	26-51
values	4-23, 5-69
values-list	5-70
variable	26-51
variable	25-17
vector	15-1, 26-51
vector	2-24, 15-6, 15-27
vector-pop	15-28
vector-push	15-28

- <http://www.lispworks.com/documentation/HyperSpec/...>
- <http://www.sbcl.org/manual/...>
- <http://ccl.closure.com/manual/...>
- <http://www.clisp.org/...>
- <http://bauhh.dyndns.org:8000/lim-spec/...>
- <http://bauhh.de/clxman/...>
- <http://metamodular.com/CLOS-MOP/...>
- <http://www.gigamonkeys.com/book/...>
- 50+ more websites with library-specific docs

(save-lisp-and-die "secret-alien-technology.core")

Fin

(save-lisp-and-die "secret-alien-technology.core")

Thanks for listening!

A Possible Solution To The State Of Common Lisp Documentation

Common Lisp UltraSpec

Michał „phoe” Herda

%%% ===== MAPCAR
%%% ===== MAPLIST
%%% ===== MAPC
%%% ===== MAPL
%%% ===== MAPCAN
%%% ===== MAPCON

\begincom{mapc, mapcar, mapcan, mapl, maplist, mapcon}**\ftype**{Function}

\label Syntax::

\DefunWithValues mapc {function **\rest** **\plus**{lists}} {list-1}
\DefunWithValues mapcar {function **\rest** **\plus**{lists}} {result-list}
\DefunWithValues mapcan {function **\rest** **\plus**{lists}} {concatenated-results}
\DefunWithValues mapl {function **\rest** **\plus**{lists}} {list-1}
\DefunWithValues maplist {function **\rest** **\plus**{lists}} {result-list}
\DefunWithValues mapcon {function **\rest** **\plus**{lists}} {concatenated-results}

\label Arguments and Values::

\param{function}---a **\term**{designator} for a **\term**{function}
that must take as many **\term**{arguments} as there are **\param**{lists}.

\issue{DOTTED-LIST-ARGUMENTS:CLARIFY}

\param{list}---a **\term**{proper list}.

\param{list-1}---the first **\param**{list} (which must be a **\term**{proper list}).

\endissue{DOTTED-LIST-ARGUMENTS:CLARIFY}

==== Function MAPC, MAPCAR, MAPCAN, MAPL, MAPLIST, MAPCON ====

The mapping operation involves applying `//function//` to successive sets of arguments in which one argument is obtained from each `//[[CL:Glossary:sequence]]//`. Except for `**mapc**` and `**mapl**`, the result contains the results returned by `//function//`. In the cases of `**mapc**` and `**mapl**`, the resulting `//[[CL:Glossary:sequence]]//` is `//list//`.

`//function//` is called first on all the elements with index `'0'`, then on all those with index `'1'`, and so on. `//result-type//` specifies the `//[[CL:Glossary:type]]//` of the resulting `//[[CL:Glossary:sequence]]//`. If `//function//` is a `//[[CL:Glossary:symbol]]//`, it is `**[[CL:Functions:coerce]]**d` to a `//[[CL:Glossary:function]]//` as if by `**[[CL:Functions:symbol-function]]**`.

`**mapcar**` operates on successive `//[[CL:Glossary:element|elements]]//` of the `//lists//`. `//function//` is applied to the first `//[[CL:Glossary:element]]//` of each `//list//`, then to the second `//[[CL:Glossary:element]]//` of each `//list//`, and so on. The iteration terminates when the shortest `//list//` runs out, and excess elements in other lists are ignored. The value returned by `**mapcar**` is a `//[[CL:Glossary:list]]//` of the results of successive calls to `//function//`.

`**mapc**` is like `**mapcar**` except that the results of applying `//function//` are not accumulated. The `//list//` argument is returned.

`**maplist**` is like `**mapcar**` except that `//function//` is applied to successive sublists of the `//lists//`. `//function//` is first applied to the `//lists//` themselves, and then to the `//[[CL:Glossary:cdr]]//` of each `//list//`, and then to the `//[[CL:Glossary:cdr]]//` of the `//[[CL:Glossary:cdr]]//` of each `//list//`, and so on.

`**mapl**` is like `**maplist**` except that the results of applying `//function//` are not accumulated: `//list-1//` is returned.

Function MAPC, MAPCAR, MAPCAN, MAPL, MAPLIST, MAPCON

The mapping operation involves applying *function* to successive sets of arguments in which one argument is obtained from each *sequence*. Except for **mapc** and **mapl**, the result contains the results returned by *function*. In the cases of **mapc** and **mapl**, the resulting *sequence* is *list*.

function is called first on all the elements with index 0, then on all those with index 1, and so on. *result-type* specifies the *type* of the resulting *sequence*. If *function* is a *symbol*, it is *coerced* to a *function* as if by *symbol-function*.

mapcar operates on successive *elements* of the *lists*. *function* is applied to the first *element* of each *list*, then to the second *element* of each *list*, and so on. The iteration terminates when the shortest *list* runs out, and excess elements in other lists are ignored. The value returned by **mapcar** is a *list* of the results of successive calls to *function*.

mapc is like **mapcar** except that the results of applying *function* are not accumulated. The *list* argument is returned.

maplist is like **mapcar** except that *function* is applied to successive sublists of the *lists*. *function* is first applied to the *lists* themselves, and then to the *cdr* of each *list*, and then to the *cdr* of the *cdr* of each *list*, and so on.

mapl is like **maplist** except that the results of applying *function* are not accumulated; *list-1* is returned.

mapcan and **mapcon** are like **mapcar** and **maplist** respectively, except that the results of applying *function* are combined into a *list* by the use of **nconc** rather than *list*. That is,

```
(mapcon f x1 ... xn) ≡ (apply #'nconc (maplist f x1 ... xn))
```

and similarly for the relationship between **mapcan** and **mapcar**.

Syntax

- **mapc** *function* &rest *lists+* → *list-1*
- **mapcar** *function* &rest *lists+* → *result-list*



Live demo/manifesto:

<http://phoe.tymoon.eu/clus/>

(save-lisp-and-die "secret-alien-technology.core")

Thanks to Shinmera for the hosting!

What am I aiming for?

- Editable
- Complete
- Downloadable
- Mirrorable/Clonable
- Versioned
- Modular
- Updatable
- Portable
- Unified
- Community-based



`(save-lisp-and-die "secret-alien-technology.core")`



The Actual End

(save-lisp-and-die "secret-alien-technology.core")

Thanks for listening!



- 01 ILLITHID - *Mark Evenson*
- 02 Sugaring Lisp for the 21st Century - *Vsevolod Dyomkin*
- 03 Common Lisp UltraSpec - *Michał Herda*
- **04 PRIMSMA - *Philip Gagner***
- 05 Managing Packages - *James Anderson*
- 06 CEPL Reaches Beta - *Chris Bagley*
- 07 QueryFS: a Solution Looking for More Problems - *Michael Raskin*
- 08 LISP/c - *Michał Herda (on behalf of Jonathan Baca)*
- 09 Electricity is Orange - *Devon McCullough*



PRIMSMA

- PRISMA is a new AI project of SCG, supported by Thompson Reuters' EIKON and ELEKTRON.
- Thompson Reuters is the largest non-military public data harvester in the world*.
- T-R data includes shipping, insurance, energy, scientific papers & data, most only for special use.

*Not counting CERN and other specialized harvesters

Why do my slides have a cute dragon on them?

- C'mon, doesn't that answer itself?*


(And because the dragon is a symbol of Romania)

T-R EIKON "SCREEN"

Top News | Front Page

Quicklinks Main Banking & Finance Breakingviews **Commodities Markets** Company News Industry News Markets News National & Regional News National Language News Sport & Lifestyle

NEWS | FRONT PAGE



ECB's Draghi signals ready to act, defends banker meetings

MOST READ 3 hours ago

BRUSSELS/ FRANKFURT, Nov 12 (Reuters) - The head of the European Central Bank underlined on Thursday the bank's readiness to extend money printing, warning that a key measure of economic health – price inflation – was flagging.

Draghi and Yellen speak before the G20 finance ministers and central bankers family portrait during the IMF/World Bank 2014 Spring Meeting in Washington
JOSHUA ROBERTS

ECB's Draghi says indications that inflation will recover are weakening
Germany takes aim at ECB policy
Bank of Korea holds rates, resists argument for cut

Steady U.S. jobless claims consistent with healthy labor market

36 mins ago

WASHINGTON, Nov 12 (Reuters) - New U.S. applications for unemployment benefits last week held steady at levels consistent with sustained labor market strength that could encourage the Federal Reserve to raise interest rates next month.

Japan machinery orders point to tepid rebound from feared recession
Japan Inc sees no turnaround soon for economy likely in recession

Draghi stimulus hint knocks euro, commodities buckle

GLOBAL MARKETS 1 hour ago

LONDON, Nov 12 (Reuters) - A signal from European Central Bank president Mario Draghi that further policy easing is coming next month pushed the euro and government bond yields lower on Thursday but failed to lift stocks, which buckled under the weight of gloomy corporate news.

US futures little changed as Yellen comments awaited
MORNING BID-AMERICAS-Weak oil

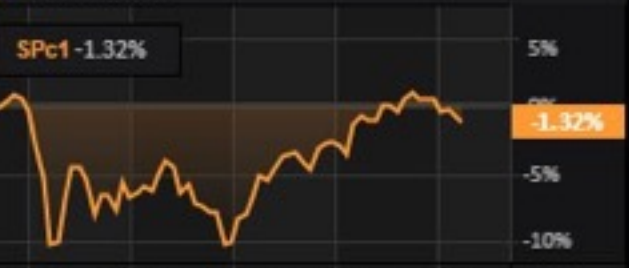
LATEST NEWS

- Nigeria 2017 bond yield spikes 341 bps on expected tighter liquidity 18 secs ago
- EARNINGS POLL-Heidelberg Druck Q2 EBIT seen up 64 percent 1 min ago
- Talanx says dividend a priority even if it misses profit goal 1 min ago
- Manulife profit misses expectations, hurt by energy losses 5 mins ago
- CANADA FX DEBT-C\$ tumbles to a six-week low with weak oil 5 mins ago
- WEEKAHEAD AFRICA FX-Naira seen stable after new disclosure rule, Kenya's shilling under pressure 6 mins ago
- UK regulators to publish review of HBOS failure on Nov. 19 7 mins ago
- Russia's Rosneft eyes oil supplies to South Africa - TASS cites ministry 7 mins ago
- U.S. postal workers union backs Sanders' presidential bid 7 mins ago
- Syrian army seizes rebel-held town in Aleppo province- state TV 8 mins ago
- Oil falls as investors fret about oversupply 8 mins ago
- RI 177-Harvard Announces Regenerative

MARKET SNAPSHOT 3 MONTHS

S&P FUTURES


SPc1 -1.32%



Aug 19 Sep 09 Sep 29 Oct 19 Nov 06

REUTERS INSIDER

ANOTHER PROMISE TO ACT FROM DRAGHI



ANOTHER PROMISE TO ACT FROM DRAGHI 12-NOV-2015 08:40

U.S. MORNING CALL: KOHL'S SURGES 12-NOV-2015 08:30

ROLL ASIA 12-NOV-2015 08:30

ANALYSIS & INSIGHT

(inject AI T-R-EIKON)

- EIKON currently can access only a fraction of available data
- PRISMA will add new data sources, but more importantly, new ways of viewing the new and old data
- What primary programming languages?
 - Javascript
 - C#
 - And Lisp for all the hard stuff

Changing the EIKON market

- T-R views EIKON mainly as a financial analyst tool
- We view EIKON as a platform for delivering economic and political analysis
- T-R has agreed to market our software on EIKON

Changing the EIKON market



- EIKON has ~300,000 subscribers, mostly financial and governments
- We want to dramatically expand their market into political and economic analysis
- We want to distribute human analysis and replace the analysts with Lisp code
- T-R has agreed to market our software on EIKON

Why Bucharest?

- It's a great city!
- Located in a country whose GDP is increasing, and debt shrinking as %GDP, top 1/3 in most economic indicators
- Lots of technology and a history of innovation*

*The jet aircraft was invented in Romania, but not implemented in Lisp

Why Romania?



- Cost of living is low, yet much to see and do
- Situated between Western Europe and Middle East
- We have 2 teams of human analysts in Romania now
- Romania built a Lisp Machine in 1984 (Professor George Stefan is one of our advisors)

DIALISP - A LISP MACHINE

G.Stefan
A.Paun
V.Bistriceanu
A.Birnbaum

Functional Electronics Laboratory
Polytechnical Institute of Bucharest
ROMANIA

ABSTRACT

High performance facilities to interpret LISP represent an ever increasing request even for minis.

This paper presents a LISP hardware structure conceived to be implemented in a general purpose mini system called DIAGRAM.

The LISP structure had to be adapted to the system technological requirements and size.

The data structure and the instruction set concerning the basic machine are also presented.

1. Introduction

The system comprising the LISP machine is shown in Fig.1, where:

- IOM is a microcomputer controlling the system input-output devices;
- MPM1 is a minicomputer on a PCB, operating as the system central unit, running high level languages (Fortran, Basic, a.s.o.). It operates with a general purpose arithmetic processor;
- MPM0 is a physical structure identical

with MPM1, controlling the alphanumeric and graphic display on a black and white or color CRT monitor. It can operate with a numerical processor specialized in bi- and tridimensional graphic transformations;

- DIALISP, the topic of this paper, is the LISP hardware interpreter.

2. General structure of LISP Hardware Interpreter (DIALISP)

2.1. Structural Options

The access time of the available memory devices used in high capacity memory arrays is 300-500 ns.

Using TTL devices, processing structures (RALU, CROM,...) having cycle time between 150 and 300 ns can be obtained.

Hence, using a cache memory the efficiency may be rather poor even when the processes associated to the LISP interpretation frequently access the memory.

Hardware facilities offered by bit-slices controlled by a stack state-machine (SSM) have been used to optimize DIALISP cost and size. Thus the whole structure is built on a single PCB, but the operating speed is

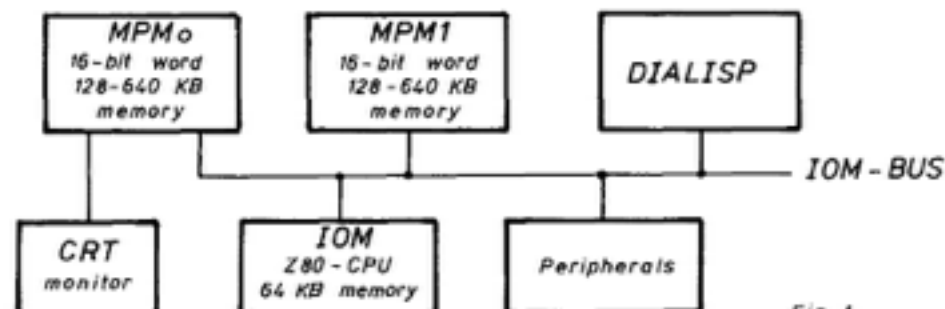


Fig. 1

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1984 ACM 0-89791-142-3 84.008 0123 \$00.75

limited due to the small number of fast registers and to the use of arithmetic comparing functions instead of logic ones, as in a dedicated structure.

Using a SSM to control the structure instead of a typical CROM configuration permits higher speed and the implementation of a large micro-stack.

2.2. Duality

An important option for DIALISP takes

What Positions?

- An Experienced Lisp Programmer willing to share knowledge
- Lisp Programmers who want to do real AI with Text Processing of Big Data
- Enthusiastic programmers willing to learn Lisp

What Else?

- Contact me at gagner@schloerconsulting.com
- Have your friends contact me too (if they're Lispers)
- Salaries, bonuses, etc. are negotiable



- 01 ILLITHID - *Mark Evenson*
- 02 Sugaring Lisp for the 21st Century - *Vsevolod Dyomkin*
- 03 Common Lisp UltraSpec - *Michał Herda*
- 04 PRIMSMA - *Philip Gagner*
- **05 Managing Packages - *James Anderson***
- **06 CEPL Reaches Beta - *Chris Bagley***
- **07 QueryFS: a Solution Looking for More Problems - *Michael Raskin***
- **08 LISP/c - *Michał Herda (on behalf of Jonathan Baca)***
- **09 Electricity is Orange - *Devon McCullough***