記 号 処 理 34-4
(1 9 8 5.1 1.1 3)

A Common Lisp Subset Proposal

Masayuki Ida
Assis.Prof, Aoyama Gakuin University
Chair. Jeida Common Lisp Committee

1. Introduction

This paper describes a Common Lisp subset proposal design by the author. Along with the AI focussing, Common Lisp [Ste84] is going to be at the position of the standard bus for AI software. Many Common Lisp implementations appeared and further implementations are going to appear.

In general, when many implementors face to the rich functions of Common Lisp, some of them may wonder all the components can be fully used for their system. And if he is a researcher, he may want to add advanced features invented by him. It will lead the divergence of the language specification. Especially for micro computer-based implementation or implementations aiming at portability have tendency to make a original subset. Furthermore personal computer based implementaions which are already sold in market place have different subsetting. The author has an interest in subsetting since '84 and discussed subsetting on meetings of Jeida Common Lisp committee [ida85a,b,c].

2. History and the status related to subsetting

2.1 In USA

There is a subset subcommittee. But, nothing was decided till now.

There are two major opinions around subsetting. One is "we need one official subset", the other is "we don't care whether the subset is defined or no". I want to quote several opinions as follows;

> The Common Lisp committee has not defined a standard subset.and it appears that at least one subset,if not many such subsets, will appear. Unless these subsets are co-ordinated and widely adopted, then we will see one of the major goals of Common Lisp --transportability-- lost.
>
> by R.P.Gabriel and R.Brooks in [Gab84]

> It is possible to subset Common Lisp, of course. The question is, will it be "officially recognized" as a Common Lisp implementation? There was a great deal of debate on this at the Monterey meeting. Some felt that the purpose of Common Lisp to permit transportability will be defeated if there are subsets. Others felt that it is important to recognize at least one official subset for use on microcomputers, especially for teaching purposes.
>
> by Guy L. Steele Jr. in [Ste85]

> I am less enthusiastic about the creation of a standard subset -- I believe that virtual memory personal computers will be available soon and that Japan will not lag behind the U.S. in this area -- but I am still willing to help with your efforts on this.
>
> by S.E. Fahlman in [Fal85]

2.2 In Europe:

The author heard at IJCAI'85 that an european subset draft would be discussed and proposed in september 1985.

2.3 In Japan (around the author):

1) the introduction of GCLisp primitives based on [gol84] was presentedat june 11th meeting of Jeida committee [ida85].

2) The second memorandum[ida85b] was presented and the first discussion at July 9th '85 meeting of Jeida committee. 21 members agree to continue to discuss a subset, 3 members have no comments.

3) 'On the subsetting Common Lisp" [ida85c]at Sept.11 Jeida Common Lisp workshop. The basic policy of a private draft is presented and discussed.

4) at Sept. 13th WGSYM evening meeting. Discussions around Common Lisp and subsetting are there.

5) 'Memo for a subset Common Lisp proposal' was presented at Oct.15th meeting of Jeida committee.

6) This paper

3.Possible points of view for subsetting

There are three types of reasoning for subset as follows;

i) For pedagogical reason.

Because, full set Common Lisp is so huge that a kind of consensus is needed to determine what is minimal set of operations and data types to teach. The author compare primitives in several educational documents such as Sun Marco explorer, R.Brooks text, Lisp 2nd

edition, and APCL of the author, and reported in [ida85b]. For pedagogical applications, not so many functions are needed.

ii) For technical reason.
It is impossible to implement full set Common Lisp on small scale personal computers/hand-held computers. Then, if there is NO control for 'Common Lisp Dialects' on PCs, it will lead incompatibility among such implementations, and the effect of the no-control policy will be bad for Lisp community.

iii) For literature reason.
There should be two levels or so of standards. Because, many languages have two levels of standard. Then it is not wrong that Common Lisp has two level; One is full set and the other is subset.

This paper agrees the above three factors. The first one (pedagogical reason) has close relations to teaching style and philosophy of lecturers. Essentially, there is no standard for teaching philosophy. But the base is needed. The rests have relations to the infrastructure of Lisp community.

4. Basic issues and decisions

4.1 General issues

1) Is this proposal for personal computer based implementation? ---> Yes and No
YES: As prof. Fahlman said in [Fah85], personal computer will grow to have much more memory space and secondary storages. Large size personal computer will be possible to run full set Common Lisp. But, there is a price range factor. There will be still low-end computers with popularity. Price of the machine which may run this subset can range among a few thousand dollars.
No: The decision upon defining subset should not include the restrictions drived from the current technology.

2) How small or slim?
A subset which is very small, cannot play the actual roles. For example, Minimal Basic seems to be dead now for almost every scene. Then this subset of CommonLisp should not go to extremes. The author have an image in mind the size is about half of full set. And may be more than 50%, but not greatly less than 50%.

3) What is the basic policy?
Don't cut arms and legs of Common Lisp

4.2 Some technical issues

1) Do we assume compiler? ----> Yes
Interpreter plays a role of command interpreter/debugger. As a matured language, a compiler should be equipped.

2) DO we keep the type hierarchy? ---> Yes
Omit some types, but, preserve hierarchy (for example, character type is independent from number)

3) Do we omit sequence type? ---> No
(for user friendlyness)

4) Do we need complex numbers? ---> No

5) Do we need the visibility of system constants and variables?
---> No (some variables and constants are left)

6) Do we need complete function sets of numbers? ---> No
preserve types except Complex numbers. delete almost all the irrational functions

7) Do we need hash-table? ---> No

8) Do we need dynamic properties of array? ---> No
Only simple and static array. no fill-pointers, no adjustable array, no bit array. three dimension, not seven, is enough.

9) How about the format feature? ---> there are two ways
1. limit the field specifier set,consulting fortran case.
subset F77 has I, F, E, L, A, '...', H, X, /, P, N, Z, BZ ,BN.
subset F77 does not have colon, S, SP ,SS, T, TL, TR, G, and list directed editing.
2. exclude all the spec of format from subset
---> include format with restricted syntax

10) How about lambda list keyword? ---> make it slim
only &optional and &rest

11) Will include the object oriented facility? ---> No.
CommonLOOPS[Bob85] will be the standard after some discussions. But not include in this subset.

12) Do we include go/prog/tagbody? --->No
"go" structure should be rewritten with other constructs. In Common Lisp, go is not always equivalent to a simple jump instruction. prog is obsolete. If there is no "go", tagbody is meaningless.

13) Keyword arguments ? -> restrict to the special cases. So parser will be easier (general parsing mechanism for keyword is not needed)

5.Summary of the subset spec

Total: 330 functions, 2 constants, 7 variables (Table 1). Fig.1 shows the type hierarchy of this subset.

Table 1 Functions of this subset

	No.of functions
program structure	2
predicate	27 + 2 const.
control structure	51
macro	5
declaration	2
symbol	10
package	1
number	51
character	21
sequence	17
list	62
hash-table	0
array	6
sting	9
structure	1
evaluator	1
stream	1 + 7 var.
input/output	22
file system	11
error	4
miscellaneous	26

Notation of the following summary:
1) listed elements are included in subset spec.
2) single braketted elements ([...]) are OMITTED.
3) double braketted elements ([[...]]) have RESTRICTED syntax/semantics
4) triple braket ([[[...]]]) contains comments

Chapter 2 Data Type

number, integer, fixnum, bignum, ratio, float, short-float,single-float, double-float, long-float, [omit: complex]
character, standard-char, string-char, [[[comment: bits-attribute and font-attribute may be zero]]]
symbol,list,cons,null
array, [[restricted: max rank=3, array means simple-array]], [omit: bit vector]
[hash table]
[[readtable;restricted to only one readtable]]
[package, pathname, stream, random-state]
structure, function
2.15 type overlap,inclusion,disjointeness
[[[preserve data type hierarchy of full set. hierarchy figure is defined as in [ida84]]]]

Chapter 3. Scope and Extent

[[[same scoping as full set]]]

Chatpter 4. type specifier

standard type specifiers of Fig.4.1 are all active, coerce, typeof, readtable, simple-array, simple-bit-vector, simple-string, simple-vector
[bit, bit-vector,complex,hash-table, stream,]
[And omit functions such as deftype, type specifier combination, predicating specifier(satisfies)]

Chapter 5. Porgram structure (2 functions)

defun,defvar
[omitted functions from Fig 5.1; tagbody, macrolet, flet, compiler-let, eval-when,go]
[&key,&allow-other-keys,&aux from lambda-list keyword, lambda-list-keyword, lambda-parameters-limit, defparameter, defconstant]

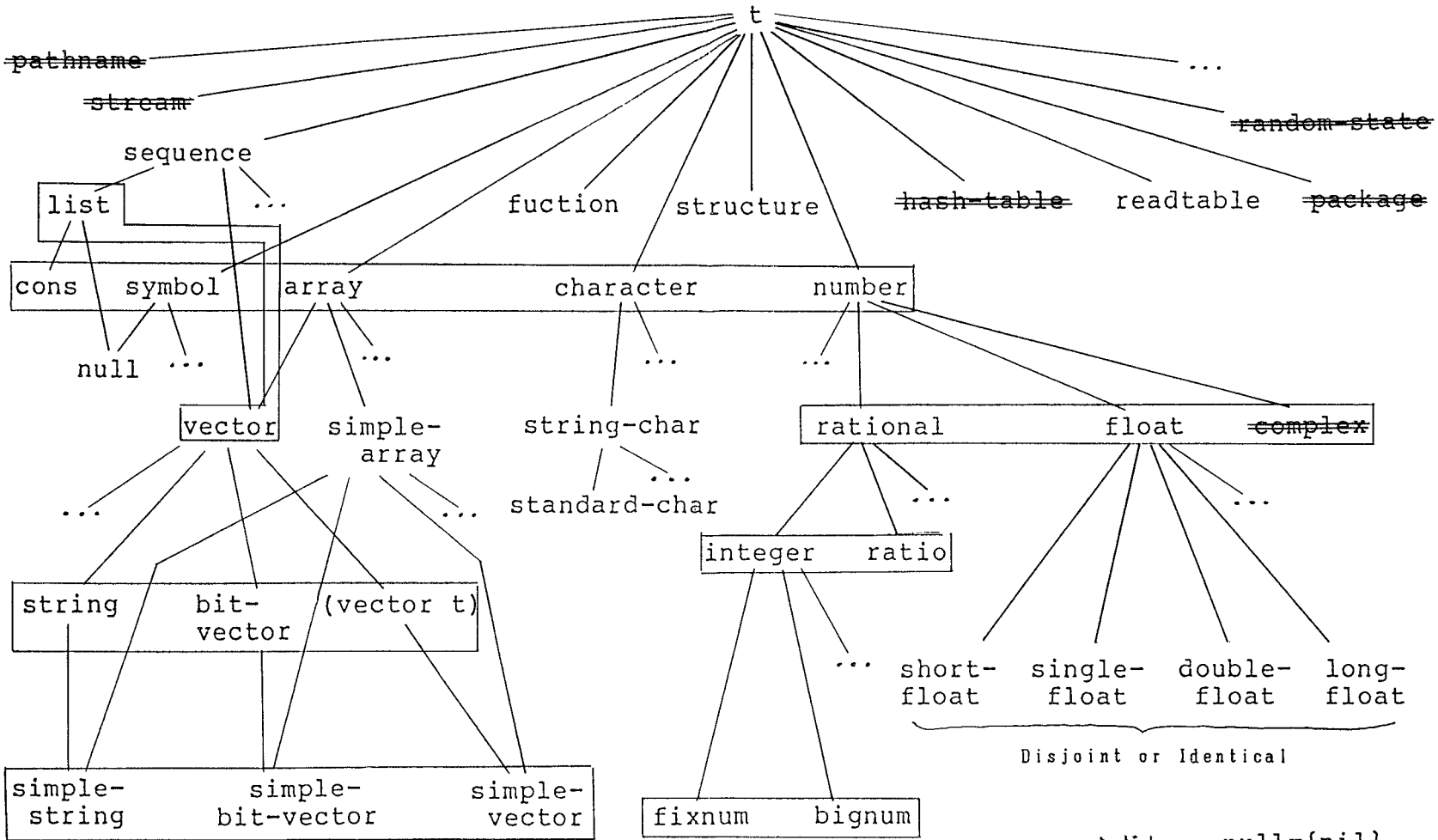
Chapter 6. predicates (27 functions, 2 constants)

nil, t, typep, subtypep, null, symbolp, atom, consp, listp, numberp, integerp, rationalp, floatp, characterp, stringp, vectorp, simple-vector-p, simple-string-p, arrayp, packagep, functionp, compiled-function-p, commonp, eq, eql, equal, not, and, or
[complexp, bit-vector-p, simple-bit-vector-p, equalp]

Chapter 7. Control Structure (51 functions)

quote, function, symbol-value, symbol-function, boundp, fboundp, special-form-p, setq, psetq, set, makunbound, fmakunbound, setf, apply, funcall, progn, prog1, prog2, let, let*, progv, labels, if, when, unless, cond, case, block, return-from, return, loop, do, do*, dolist, dotimes, maplist, mapcar, mapc, mapl, mapcan, mapcon, values, values-list, multiple-value-list, multiple-value-prog1, multiple-value-bind, multiple-value-setq, multiple-value-call, catch, unwind-protect, throw
[psetf, shiftf, rotatef, defsetf, define-setf-method, define-modify-macro, get-setf-method, get-setf-method-multiple-value, call-argument-limit, compiler-let, flet, macrolet, prog, go, typecase, tagbody]

setf accessible forms;
aref, car, cdr, c...r, fill-pointer, first, get, getf, nth, nthcdr, rest, second, symbol-function, symbol-plist, symbol-value, third, access function of defstruct,



ただし、 null={nil}、

□ は Disjoint、

すべての型の副型として、nil 型がある。

図1 Common Lisp Subset Type Hierarchy

[omitted access form; elt,fourth--tenth,svref,gethash,documentation, c....r, macro-function, (char, schar, bit, sbit, subseq) function calls, the-declaration, apply]

Chapter 8. Macro (5 functions)
macro-function, defmacro, macro, macroexpand, macroexpand-1
[[defmacro keywords only to &optional and &rest]]
[omit *macro-expand-hook*, &key, &allow-other-keys, &aux, &body, &whole, &environment]

Chapter 9. Declarations (2 functions)
Only (declare (special ...)) acceptable
[omit all the spec of declarations but above]
[[["the" acceptable but not effective]]]

Chapter 10. Symbols (10 functions)
get, remprop, symbol-plist, getf, remf, symbol-name,
make-symbol, copy-symbol, gensym, keywordp
[get-properties, gentmp, symbol-package]

Chapter 11. Package (1 function)
[All the spec of Package features are omitted,but below]
11.1 consistency rules, intern

Chapter 12. Number (51 functions)
zerop, plusp, minusp, oddp, evenp, =, /=, <, >, <=, >=, max, min, +, -, *, /, 1+, 1-, incf, decf, gcd, lcm, exp, expt, log, sqrt, abs, signum, sin, cos, tan, rational, rationalize, numerator, denominator, float, floor, ceiling, truncate, round, mod, rem, logior, logxor, logand, logeqv, lognot, logtest. logbitp, ash
[[complex related features of above fuctions are omitted]]
[complex and byte-manipulation functions, random-numbers, implementation parameters described in 12.10.
conjugate, isqrt, phase, cis, asin, acos, atan, pi, sinh, cosh, tanh, sinh, acosh, atanh, ffloor, fceiling, ftruncate, fround, decode-float -- integer-decode-float, complex, realpart, imagpart, lognand, logandc1, logandc2, logorc1, logorc2, boole, bool- constants byte, byte-size, byte-position, ldb, ldb-test, mask-field, dpb, deposit-field, random, *random-state*, make-random-state]

Chapter 13. Characters (21 functions)
standard-char-p, alpha-char-p, upper-case-p, both-case-p, digit-char-p, char=, char/=, char<, char<=, char-code, code-char, make-char, char-upcase, digit-char, char-int, int-char, char-downcase, char-name, name-char, char-bit, set-char-bit

[char-code-limit, char-font-limit, char-bits-limit, graphic-char-p, string-char-p, alphanumericp, char-equal, char-not-equal, char-lessp, char-greaterp, char-not-greaterp, char-not-lessp, char>, char>=, 13.5 char control bit functions]

Chapter 14. Sequences (17 functions)
subseq, reverse, nreverse, some, every, notany, notevery, sort
[[as to fill pointer:: elt,length]]
[[as to :start - :end :: remove, remove-if, remove-if-not, remove-duplicates, find, position, count]]

:count, :from-end, :strat, :end, make-sequence, concatenate, map, reduce, fill, replace, delete, delete-if, delete-if-not, delete-duplicates, substitute, substitute-if, substitute-if-not, nsubstitute, nsubstitute-if, nsubstitute-if-not, find-if, find-if-not, position-if, position-if-not, count-if, count-if-not, mismatch, search, stable-sort, merge]

Chapter 15. list (62 functions)
car, cdr, c..r, c...r, cons, tree-equal, endp, list-length, nth, first, second, third, rest, nthcdr, last, list, list*, append, copy-list, copy-alist, copy-tree, nconc, push, pushnew, pop, butlast, nbutlast, ldiff, rplaca, rplacd, pairlis

[[keyword not allowed; subst, nsubst, sublis, member, member-if, member-if-not, adjoin, union, nunion, intersection, nintersection, set-defference, nset-difference, set-exclusive-or, nset-exclusive-or, assoc,assoc-if, assoc-if-not, rassoc, rassoc-if, rassoc-if-not]]

[c....r, fourth -- tenth, make-list, revappend, nreconc, subst-if, subst-if-not, nsubst-if, nsubst-if-not, subsetp, acons, tailp]

Chapter 16. Hash-table (0 functions)
[ALL]

Chapter 17. Array (6 functions)
vector, aref, array-rank, array-in-bounds-p, array-total-size

[[make-array(limit to three dimensions (not seven), all the keyword arguments but :initial-contents are deleted)]]

[array-rank-limit, array-dimension-limit, array-total-size-limit, svref, array-elemet-type, array-dimension, array-dimensions, array-row-major-index,

adjustable-array-p, bit, sbit, bit-and, bit-or, ..., bitorc2, bit-not, array-has-fill-pointer-p, fill-pointer, vector-push, vector-push-extend, vector-pop, adjust-array, 17.4 all (bit-array),17.5 all (fill pointer),17.6 all (augumentation of dimension)]

Chapter 18. string (9 functions)
char, string-trim, string-left-trim, string-right-trim, string-capitalize

[[subseq(:start and :end) related restriction :: string=, string/=, string<, string<=]]

[schar, string>, string>=, string-equal, string-lessp, ...,make-string, string-upcase, string-downcase, nstring-upcase, nstring-down-case, nstring-capitalize]

Chapter 19. structure (1 functions)
[[defstruct]]

[:type,:read-only (in defstruct slot option), :conc-name, :constructor, :copier, :predicate, :include, :print-function, :type, :named, :initial-offset (in defstruct option), by-position constructor]

Chapter 20. Evaluator (1 function)
eval
[evalhook, applyhook, *evalhook*, *applyhook*, constantp, +, ++, +++, -, *, **, ***, /, //, ///]

Chapter 21. stream (1 function, 7 variables)
standard-input, *standard-output*, *error-output*, *query-io*, *debug-io*, *terminal-io*, *trace-output*

[[close (No :abort arg)]]
[make-synonym-stream, make-...-stream, get-output-stream-string, with-input-from-string, with-output-to-string,streamp, input-stream-p, output-stream-p, stream-element-type]

Chapter 22. input/output (22 functions)

22.1.4 dispatching macro character
#',#(,#+,#-,#B,#D,#S,#X,#nA,#|
[#,#*,#:,#.,#,,#O,#nR,#n=,#n#,#<,#)]

22.1.5 [*readtable*, copy, readtable, readtablep, set-syntax-from-char, set/get-macro-character, make-dispatch-macro-character, set/get-dispatch-macro-character]
22.1.6 [*print-...* all omitted]
22.2 input
read, read-line, read-char, unread-char, listen, read-char-no-hang, clean-input, read-byte
[*read-default-float-format*, read-preserving-whitespace, read-delimited-list, peek-char, parse-integer]
[[read-from-string (no keyword arg)]]

22.3 output
write-byte, prin1, print, pprint, princ, prin1-to-string, princ-to-string, write-char, terpri, fresh-line, format, y-or-n-p, yes-or-no-p

22.3.3 format specifier:
~A, ~S, ~D, ~X, ~C, ~F, ~E, ~%, ~&, ~|, ~~, ~<newline>, ~{, ~}
[~B, ~O, ~nR, ~P, ~G, ~$, ~T, ~*, ~?, ~(str~), ~[str0~;...strn~], ~<str~>]

Chapter 23.file system interface (11 functions)

23.1 file name pathname
file name should be string
[all the pathname related definitions are omitted]
23.2 file open/close
open
[[file name should be string (:direction is Only input or output, :element-type is only string-char or unsigned-byte, :if-exists omitted, :if-does-not-exist omitted)]], with-open-file
23.3 rename-file, delete-file, probe-file, file-write-date, file-author, file-position, file-length
23.4 [[load(no keyword)]], [*load-verbose*]
23.5 directory

Chapter 24. error (4 functions)
error, cerror, warn, break
[*break-on-warning*, check-type, assert, etypecase, ctypecase, ecase, ccase]

Chapter 25. miscellaneous (26 functions)
compile, compile-file, documentation, trace, untrace, step,time, describe, inspect, room, ed, dribble, appropos, appropos-list, get-decoded-time, get-internal-run-time, get-internal-real-time, sleep, lisp-implementation-type, *features*, lisp-implementation-version, machine-type, machine-version, machine-instance, software-type, software-version

[disassemble, get-universal-time, decode/encode-universal-time, internal-time-units-per-second, short-site-name, long-site-name, identity]

This subset draft will be submitted to Jeida Committee and discussed. Also sent to some US researchers. The summary of this draft is on fj.lang.lisp newsgroup of junet. (The author's junet address for the matter is 'ida@ccut') The author

will appreciate opinions from all the persons through network, post mail and face to face communication.

Acknowledgements

The author want express his gratitude to all the persons who intersted in subsetting and encouraged him, especially; For the members of Jeida Common Lisp committee , members of WGSYM of IPSJ, Dr. Guy L. Steele (Thinking Machines Inc.), Prof S.E.Fahlman (CMU), Dr. Gerald Barber and Mr.Stan Curtis (Gold Hill Computers Inc.) who introduced GCLisp to the author, Dr. Masinter (Xerox PARC), and Prof. Tuji (Aoyama Gakuin University) who gave him the informations on [bro84].

References

[Bob85] Bobrow D.G., et.al;
CommonLOOPS, ISL-85-8 Xerox PARC,1985
[bro84] R.Brooks; Text for CS122 stanford univ.
[Fal85] S.E. Fahlman; private communication to the author dated May 14, 1985
[Gab84] Gabriel R.P., Brooks R.;
A Critique on Common Lisp, pp1-8, Proc. of 1984 Lisp conference, Aug.1984
[gol84]; Gold Hill Computers Inc.:
golden common lisp reference manual version 1.00
[ida85a] M.Ida: Memo for GCLisp primitives, LC2-2
Jeida Common Lisp committee, June 11th 1985
[ida85b] -: One view point toward Common Lisp subset,
LC-3-2, Jeida Common Lisp Committee, 1985 July 9th
[ida85c] -:On a Common Lisp subset;
preprint of Common Lisp workshop, Sept.11, 1985
[Ste84] Guy L. Steele Jr. et. al.;
Common Lisp: the language, Digital Press 1984 (Japanese edition by M.Ida, Kyouritu Pub. Corp. 1985)
[Ste85] -; private communication to the author dated March 27,1985
(also appeared in M.Ida;"Some topics of Common Lisp" appendix 7.
WGSYM IPSJ June 1985)
[Win84] P.H.Winston, B.K.P.Horn;
Lisp 2nd edition, 1984, addison wesley