The Incomplete Lojban Language

Chrestomathy included

John Woldemar Cowan

An unofficial publication, community edition (not by the LLG)

Version geklojban-1.2.16, Generated 2024-03-10

Table of Contents

Chapter 1
Formal grammars

1.1 YACC Grammar of Lojban

The following two listings constitute the formal grammar of Lojban. The first version is written in the YACC language, which is used to describe parsers, and has been used to create a parser for Lojban texts. This parser is available from the Logical Language Group. The second listing is in Extended Backus-Naur Form (EBNF) and represents the same grammar in a more human-readable form. (In case of discrepancies, the YACC version is official.) There is a cross-reference listing for each format that shows, for each selma'o and rule, which rules refer to it.

The Lojban machine parsing algorithm is a multi-step process. The YACC machine grammar presented here is an amalgam of those steps, concatenated so as to allow YACC to verify the syntactic ambiguity of the grammar. YACC is used to generate a parser for a portion of the grammar, which is LALR1 (the type of grammar that YACC is designed to identify and process successfully), but most of the rest of the grammar must be parsed using some language-coded processing.

Step 1 – Lexing

From phonemes, stress, and pause, it is possible to resolve Lojban unambiguously into a stream of words. Any machine processing of speech will have to have some way to deal with "non-Lojban" failures of fluent speech, of course. The resolved words can be expressed as a text file using Lojban's phonetic spelling rules.

The following steps assume that there is the possibility of non-Lojban text within the Lojban text (delimited appropriately). Such non-Lojban text may not be reducible from speech phonetically. However, step 2 allows the filtering of a phonetically transcribed text stream, to recognize such portions of non-Lojban text where properly delimited, without interference with the parsing algorithm.

Step 2 – Filtering

From start to end, performing the following filtering and lexing tasks using the given order of precedence in case of conflict:

i. If the Lojban word *zoi* (selma'o ZOI) is identified, take the following Lojban word (which should be end delimited with a pause for separation from the following non-Lojban text) as an opening delimiter. Treat all text following that delimiter, until that delimiter recurs *after a pause*, as grammatically a single token (labelled “YACC rule #699 (p. 10)” in this grammar). There is no need for processing within this text except as necessary to find the closing delimiter.

ii. If the Lojban word *zo* (selma'o ZO) is identified, treat the following Lojban word as a token labelled “YACC rule #698 (p. 10)”, instead of lexing it by its normal grammatical function.

iii. If the Lojban word *lo'u* (selma'o LOhU) is identified, search for the closing delimiter *le'u* (selma'o LEhU), ignoring any such closing delimiters absorbed by the previous two steps. The text between the delimiters should be treated as the single token “YACC rule #697 (p. 10)”.

iv. Categorize all remaining words into their Lojban selma'o category, including the various delimiters mentioned in the previous steps. In all steps after step 2, only the selma'o token type is significant for each word.

v. If the word *si* (selma'o SI) is identified, erase it and the previous word (or token, if the previous text has been condensed into a single token by one of the above rules).

vi. If the word *sa* (selma'o SA) is identified, erase it and all preceding text as far back as necessary to make what follows attach to what precedes. (This rule is hard to formalize and may receive further definition later.)

vii. If the word *su* (selma'o SU) is identified, erase it and all preceding text back to and including the first preceding token word which is in one of the selma'o: NIhO, LU, TUhE, and TO. However, if speaker identification is available, a SU shall only erase to the beginning of a speaker's discourse, unless it occurs at the beginning of a speaker's discourse. (Thus, if the speaker has said something, two adjacent uses of *su* are required to erase the entire conversation.

Step 3 – Termination

If the text contains a FAhO, treat that as the end-of-text and ignore everything that follows it.

Step 4 – Absorption of Grammar-Free Tokens

In a new pass, perform the following absorptions (absorption means that the token is removed from the grammar for processing in following steps, and optionally reinserted, grouped with the absorbing token after parsing is completed).

i. Token sequences of the form any - (ZEI - any) ..., where there may be any number of ZEIs, are merged into a single token of selma'o BRIVLA.

ii. Absorb all selma'o BAhE tokens into the following token. If they occur at the end of text, leave them alone (they are errors).

iii. Absorb all selma'o BU tokens into the previous token. Relabel the previous token as selma'o BY.

iv. If selma'o NAI occurs immediately following any of tokens UI or CAI, absorb the NAI into the previous token.

v. Absorb all members of selma'o DAhO, FUhO, FUhE, UI, Y, and CAI into the previous token. All of these null grammar tokens are permitted following any word of the grammar, without interfering with that word's grammatical function, or causing any effect on the grammatical interpretation of any other token in the text. Indicators at the beginning of text are explicitly handled by the grammar.

Step 5 – Insertion of Lexer Lexemes

Lojban is not in itself LALR1. There are words whose grammatical function is determined by following tokens. As a result, parsing of the YACC grammar must take place in two steps. In the first step, certain strings of tokens with defined grammars are identified, and either

i. are replaced by a single specified “lexer token” for step 6, or

ii. the lexer token is inserted in front of the token string to identify it uniquely.

The YACC grammar included herein is written to make YACC generation of a step 6 parser easy regardless of whether a. or b. is used. The strings of tokens to be labelled with lexer tokens are found in rule terminals labelled with numbers between 900 and 1099. These rules are defined with the lexer tokens inserted, with the result that it can be verified that the language is LALR1 under option b. after steps 1 through 4 have been performed. Alternatively, if option a. is to be used, these rules are commented out, and the rule terminals labelled from 800 to 900 refer to the lexer tokens *without* the strings of defining tokens. Two sets of lexer tokens are defined in the token set so as to be compatible with either option.

In this step, the strings must be labelled with the appropriate lexer tokens. Order of inserting lexer tokens *IS* significant, since some shorter strings that would be marked with a lexer token may be found inside longer strings. If the tokens are inserted before or in place of the shorter strings, the longer strings cannot be identified.

If option a. is chosen, the following order of insertion works correctly (it is not the only possible order): A, C, D, B, U, E, H, I, J, K, M, N, G, O, V, W, F, P, R, T, S, Y, L, Q. This ensures that the longest rules will be processed first; a PA+MAI will not be seen as a PA with a dangling MAI at the end, for example.

Step 6 – YACC Parsing

YACC should now be able to parse the Lojban text in accordance with the rule terminals labelled from 1 to 899 under option 5a, or 1 to 1099 under option 5b. Comment out the rules beyond 900 if option 5a is used, and comment out the 700-series of lexer-tokens, while restoring the series of lexer tokens numbered from 900 up.

%token
A_501 eks; basic afterthought logical connectives
%token
BAI_502 modal operators
%token
BAhE_503 next word intensifier
%token
BE_504 sumti link to attach sumti to a selbri
%token
BEI_505 multiple sumti separator between BE, BEI
%token
BEhO_506 terminates BE/BEI specified descriptors
%token
BIhI_507 interval component of JOI
%token
BO_508 joins two units with shortest scope
%token
BRIVLA_509 any brivla
%token
BU_511 turns any word into a BY lerfu word
%token
BY_513 individual lerfu words
%token
CAhA_514 specifies actuality/potentiality of tense
%token
CAI_515 afterthought intensity marker
%token
CEI_516 pro-bridi assignment operator

%token

CEhE_517 afterthought term list connective

%token

CMENE_518 Lojbanized names; require consonant end, as well as a pause before and after them

%token

CO_519 tanru inversion

%token

COI_520 vocative marker permitted inside cmevla; must always be followed by pause or DOI

%token

CU_521 separator between head sumti and selbri

%token

CUhE_522 tense/modal question

%token

DAhO_524 cancel anaphora/cataphora assignments

%token

DOI_525 vocative marker

%token

DOhU_526 terminator for DOI-marked vocatives

%token

FA_527 modifier head generic case tag

%token

FAhA_528 superdirections in space

%token

FAhO_529 normally elided “done pause” to indicate end of utterance string

%token

FEhE_530 space interval mod flag

%token

FEhU_531 ends bridi to modal conversion

%token

FIhO_532 marks bridi to modal conversion

%token

FOI_533 end compound lerfu

%token

FUhE_535 open long scope for indicator

%token

FUhO_536 close long scope for indicator

%token

GA_537 geks; forethought logical connectives

%token

GEhU_538 marker ending GOI relative clauses

%token

GI_539 forethought medial marker

%token

GIhA_541 logical connectives for bridi-tails

%token

GOI_542 attaches a sumti modifier to a sumti

%token

GOhA_543 pro-bridi

%token

GUhA_544 GEK for tanru units, corresponds to JEKs

%token

I_545 sentence link
%token
JA_546 jeks; logical connectives within tanru
%token
JAI_547 modal conversion flag
%token
JOI_548 non-logical connectives
%token
KEhE_550 right terminator for KE groups
%token
KE_551 left long scope marker
%token
KEI_552 right terminator, NU abstractions
%token
KI_554 multiple utterance scope for tenses
%token
KOhA_555 sumti anaphora
%token
KU_556 right terminator for descriptions, etc.
%token
KUhO_557 right terminator, NOI relative clauses
%token
LA_558 name descriptors
%token
LAU_559 lerfu prefixes
%token
LAhE_561 sumti qualifiers
%token
LE_562 sumti descriptors
%token
LEhU_565 possibly ungrammatical text right quote
%token
LI_566 convert number to sumti
%token
LIhU_567 grammatical text right quote
%token
LOhO_568 elidable terminator for LI
%token
LOhU_569 possibly ungrammatical text left quote
%token
LU_571 grammatical text left quote
%token
LUhU_573 LAhE close delimiter
%token
ME_574 converts a sumti into a tanru_unit
%token
MEhU_575 terminator for ME
%token
MOhI_577 motion tense marker
%token
NA_578 bridi negation
%token
NAI_581 attached to words to negate them
%token

NAhE_583 scalar negation
%token
NIhO_584 new paragraph; change of subject
%token
NOI_585 attaches a subordinate clause to a sumti
%token
NU_586 abstraction
%token
NUhI_587 marks the start of a termset
%token
NUhU_588 marks the middle and end of a termset
%token
PEhE_591 afterthought termset connective prefix
%token
PU_592 directions in time
%token
RAhO_593 flag for modified interpretation of GOhI
%token
ROI_594 converts number to extensional tense
%token
SA_595 metalinguistic eraser to the beginning of the current utterance
%token
SE_596 conversions
%token
SEI_597 metalinguistic bridi insert marker
%token
SEhU_598 metalinguistic bridi end marker
%token
SI_601 metalinguistic single word eraser
%token
SOI_602 reciprocal sumti marker
%token
SU_603 metalinguistic eraser of the entire text
%token
TAhE_604 tense interval properties
%token
TEI_605 start compound lerfu
%token
TO_606 left discursive parenthesis
%token
TOI_607 right discursive parenthesis
%token
TUhE_610 multiple utterance scope mark
%token
TUhU_611 multiple utterance end scope mark
%token
UI_612 attitudinals, observationals, discursives
%token
VA_613 distance in space-time
%token
VAU_614 end simple bridi or bridi-tail
%token
VEhA_615 space-time interval size

%token

VIhA_616 space-time dimensionality marker

%token

VUhO_617 glue between logically connected sumti and relative clauses

%token

XI_618 subscripting operator

%token

Y_619 hesitation

%token

ZAhO_621 event properties – prospective, etc.

%token

ZEhA_622 time interval size tense

%token

ZEI_623 lujvo glue

%token

ZI_624 time distance tense

%token

ZIhE_625 conjoins relative clauses

%token

ZO_626 single word metalinguistic quote marker

%token

ZOI_627 delimited quote marker

%token

ZOhU_628 prenex terminator (not elidable)

%token

BIhE_650 prefix for high-priority MEX operator

%token

BOI_651 number or lerfu-string terminator

%token

FUhA_655 reverse Polish flag

%token

GAhO_656 open/closed interval markers for BIhI

%token

JOhI_657 flags an array operand

%token

KUhE_658 MEX forethought delimiter

%token

MAI_661 change numbers to utterance ordinals

%token

MAhO_662 change MEX expressions to MEX operators

%token

MOI_663 change number to selbri

%token

MOhE_664 change sumti to operand, inverse of LI

%token

NAhU_665 change a selbri into an operator

%token

NIhE_666 change selbri to operand; inverse of MOI

%token

NUhA_667 change operator to selbri; inverse of MOhE

%token

PA_672 numbers and numeric punctuation

%token

PEhO_673 forethought (Polish) flag
%token
TEhU_675 closing gap for MEX constructs
%token
VEI_677 left MEX bracket
%token
VEhO_678 right MEX bracket
%token
VUhU_679 MEX operator
%token
any_words_697 a string of lexable Lojban words
%token
any_word_698 any single lexable Lojban words
%token
anything_699 a possibly unlexable phoneme string
The following tokens are the actual lexer tokens. The _900 series tokens are duplicates that allow limited testing of lexer rules in the context of the total grammar. They are used in the actual parser, where the 900 series rules are found in the lexer.
%token lexer_
A_701 flags a MAI utterance ordinal
%token lexer_
B_702 flags an EK unless EK_BO, EK_KE
%token lexer_
C_703 flags an EK_BO
%token lexer_
D_704 flags an EK_KE
%token lexer_
E_705 flags a JEK
%token lexer_
F_706 flags a JOIK
%token lexer_
G_707 flags a GEK
%token lexer_
H_708 flags a GUhEK
%token lexer_
I_709 flags a NAhE_BO
%token lexer_
J_710 flags a NA_KU
%token lexer_
K_711 flags an I_BO (option. JOIK/JEK lexer tags)
%token lexer_
L_712 flags a PA, unless MAI (then lexer A)
%token lexer_
M_713 flags a GIhEK_BO
%token lexer_
N_714 flags a GIhEK_KE
%token lexer_
O_715 flags a modal operator BAI or compound
%token lexer_
P_716 flags a GIK
%token lexer_
Q_717 flags a lerfu_string unless MAI (then lexer_A)
%token lexer_

```
R_718 flags a GIhEK, not BO or KE
%token lexer_
S_719 flags simple I
%token lexer_
T_720 flags I_JEK
%token lexer_
U_721 flags a JEK_BO
%token lexer_
V_722 flags a JOIK_BO
%token lexer_
W_723 flags a JOIK_KE
%token lexer_
X_724 null
%token lexer_
Y_725 flags a PA_MOI
%token lexer_A_905 : lexer_A_701 utt_ordinal_root_906
%token lexer_B_910 : lexer_B_702 EK_root_911
%token lexer_C_915 : lexer_C_703 EK_root_911 BO_508
%token lexer_D_916 : lexer_D_704 EK_root_911 KE_551
%token lexer_E_925 : lexer_E_705 JEK_root_926
%token lexer_F_930 : lexer_F_706 JOIK_root_931
%token lexer_G_935 : lexer_G_707 GA_537
%token lexer_H_940 : lexer_H_708 GUhA_544
%token lexer_I_945 : lexer_I_709 NAhE_583 BO_508
%token lexer_J_950 : lexer_J_710 NA_578 KU_556
%token lexer_K_955 : lexer_K_711 I_432 BO_508
%token lexer_L_960 : lexer_L_712 number_root_961
%token lexer_M_965 : lexer_M_713 GIhEK_root_991 BO_508
%token lexer_N_966 : lexer_N_714 GIhEK_root_991 KE_551
%token lexer_O_970 : lexer_O_715 simple_tense_modal_972
%token lexer_P_980 : lexer_P_716 GIK_root_981
%token lexer_Q_985 : lexer_Q_717 lerfu_string_root_986
%token lexer_R_990 : lexer_R_718 GIhEK_root_991
%token lexer_S_995 : lexer_S_719 I_545
%token lexer_T_1000 : lexer_T_720 I_545 simple_JOIK_JEK_957
%token lexer_U_1005 : lexer_U_721 JEK_root_926 BO_508
%token lexer_V_1010 : lexer_V_722 JOIK_root_931 BO_508
%token lexer_W_1015 : lexer_W_723 JOIK_root_931 KE_551
%token lexer_X_1020 null
%token lexer_Y_1025 : lexer_Y_725 number_root_961 MOI_663
```

Lojban Words Glossary

All definitions in this glossary are brief and unofficial. Only the published dictionary is a truly official reference for word definitions. These definitions are here simply as a quick reference.

le'u
placeholder definition

lo'u
placeholder definition

sa
placeholder definition

si
placeholder definition

su
placeholder definition

zo
placeholder definition

zoi
placeholder definition

Lojban Words Index

General Index