

Spring 2018 Report on the State of the Loglan Language (official and provisional)

M. Randall Holmes

Version of 2/10/2017, 5 pm Mountain Time

1 Introduction

This document is intended to be a comprehensive report on the state of the Loglan language as of Spring 2018. The Phonetics Proposal is now the only section on phonetics, including permitted phonetic shapes of words, and later sections have been reconciled with it (at least, a first pass has been made at this). I do attempt to distinguish between things officially approved and things which are still provisional, whether proposed by me or by others. A Lexicography Proposal now replaces the section on Word Forms (containing a lot of the content of the old section, rearranged and edited). I would like to encourage a decision to accept all or most of the provisional features at once, at least in principle, rather than continuing a very slow piecemeal revision, but I do not insist on such an approach. By providing a reasonably structured overview of what I have done, I hope to encourage such an outcome, and the document should be useful in any case.

The document contains an overview of the issues I perceived when I started working on overhauling the language in 2013 along with a high-level description of what I have done about them (officially or provisionally). Anyone in the Loglan community who sees oversights in this section (or any section) is welcome to tell me about them!

This is followed by the entire text of the draft reference grammar (and it is in the context of this document that the reference grammar will continue to be maintained). The reference grammar is intended to give a complete description of the grammar and grammatical vocabulary of the language without the distraction of PEG notation.

This is followed by three Appendices, the list of proposals before the Loglan Academy, taken from the Academy agenda document, and augmented with further draft proposals, followed by an annotated text of the PEG grammar, similar to what is currently embedded in the agenda document, but freshly prepared for this document [in some places I did copy in large chunks of annotations from the old agenda document], and the complete trial.85 grammar for reference.

I am planning further editing of this document. For example, it would make sense to add comments about subsequent changes in the embedded copy of the trial.85 document.

The reference grammar needs considerable work; it will be edited in situ in this document.

I plan to add a small section before the annotated PEG grammar explaining how to read a PEG.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 2 | Version Notes | 6 |
| 3 | A brief report, Fall 2016 | 11 |
| 4 | A catalogue of issues | 19 |
| 5 | Introduction to the Reference Grammar Sections | 33 |
| 6 | The Phonetics Proposal 2017 | 34 |
| 6.1 | Sounds | 34 |
| 6.1.1 | Vowels | 34 |
| 6.1.2 | Consonants | 35 |
| 6.1.3 | Vowels pretending to be consonants | 36 |
| 6.1.4 | Consonants pretending to be vowels | 36 |
| 6.2 | Diphthongs and vowel grouping | 36 |
| 6.2.1 | Mandatory diphthongs | 36 |
| 6.2.2 | Optional diphthongs | 37 |
| 6.2.3 | Expanded notes on diphthong pronunciation | 37 |
| 6.2.4 | Grouping long streams of vowels in predicates and names | 39 |
| 6.2.5 | Grouping long streams of vowels in structure words . . | 39 |
| 6.2.6 | Vowel pairs with optional grouping revisited | 40 |
| 6.2.7 | Doubled vowels and stress | 40 |
| 6.3 | The Loglan syllable | 40 |
| 6.3.1 | Discussion of stress and notation for syllable breaks and degrees of stress | 40 |
| 6.3.2 | Every syllable has a vocalic unit | 42 |
| 6.3.3 | Like Gaul, every syllable has (up to) three parts | 42 |
| 6.3.4 | Words must resolve into syllables | 43 |
| 6.3.5 | Notation for consonant and vowel patterns | 43 |
| 6.4 | Word Forms | 44 |
| 6.4.1 | Word forms enumerated | 44 |
| 6.4.2 | Pauses and word boundaries | 44 |
| 6.4.3 | Structure words (cmapua) | 45 |
| 6.4.4 | Name words and the false name marker problem | 47 |
| 6.4.5 | Borrowed predicates | 50 |

| | | |
|----------|---|-----------|
| 6.4.6 | Complex predicates | 53 |
| 6.5 | Appendix: Alien Text and Quotations | 56 |
| 7 | Lexicography (a formal Proposal as of 2/3/2018) | 57 |
| 7.1 | Name Words (and Alien Text) | 58 |
| 7.2 | Predicate Words | 60 |
| 7.3 | Structure Word Classes Introduced | 61 |
| 7.4 | Tight logical connectives: CA roots | 62 |
| 7.5 | Letters, acronyms, and pronouns | 62 |
| 7.5.1 | Remarks on acronyms | 64 |
| 7.6 | Numerals and quantifiers | 65 |
| 7.7 | Tense/location/relation operators | 69 |
| 7.7.1 | The system of tense and location words | 74 |
| 7.8 | Connectives | 78 |
| 7.8.1 | Logical connectives for sentence components | 78 |
| 7.8.2 | Sentence connectives and new utterance markers | 79 |
| 7.8.3 | Forethought logical and causal connectives | 80 |
| 7.9 | Articles | 81 |
| 7.9.1 | Constructions involving alien text and related articles (see the appendix to the Phonetics Proposal for some modifications) | 82 |
| 7.10 | Assorted grammatical particles, somewhat classified | 84 |
| 7.11 | Words which form free modifiers | 86 |
| 7.12 | Negation | 91 |
| 7.13 | Essays on word-making, and on what a word is exactly | 91 |
| 7.13.1 | Borrowing predicates | 91 |
| 7.13.2 | Making complex predicates | 92 |
| 7.13.3 | Name words | 93 |
| 7.13.4 | Essay: what is a word? | 94 |
| 8 | Grammatical Constructions | 97 |
| 8.0.5 | Note on Right Closers | 97 |
| 8.1 | Sentences and Utterances | 97 |
| 8.1.1 | The most basic sentences | 98 |
| 8.1.2 | Logically connected basic sentences (and final argu- ments moved to the front) | 100 |
| 8.1.3 | Free Modifiers and Utterances | 101 |
| 8.2 | A semantic note: scopes of quantifiers | 106 |

| | | |
|-----------|--|------------|
| 8.3 | Predicates | 107 |
| 8.3.1 | The basic building blocks of predicates: predunit classes | 107 |
| 8.3.2 | Description predicates | 108 |
| 8.3.3 | Sentence predicates, first pass | 109 |
| 8.3.4 | Sentence predicates, second pass | 110 |
| 8.4 | Clauses, arguments and term lists | 112 |
| 8.4.1 | Serial names and the false name marker problem | 112 |
| 8.4.2 | Arguments (including subordinate clauses) | 113 |
| 8.4.3 | A semantic note on multiple reference of arguments . . | 118 |
| 8.4.4 | Modifiers = relative clauses, prepositional phrases . . . | 119 |
| 8.4.5 | Terms, term lists, and termsets (including link sets) . . | 119 |
| 9 | Appendix: The Current and Recent Active Proposals (and some draft proposals of mine in preparation) with Comments | 122 |
| 10 | The PEG implementing the Phonetics Proposal, the new provisional grammar (the following one is preserved because the comments are valuable) | 140 |
| 11 | The previous PEG test Grammar, fresh annotations completed | 191 |
| 12 | Appendix: The Trial.85 Grammar | 256 |
| 13 | Appendix: old version notes for this document | 279 |

2 Version Notes

2/10/2018 removed nu- as an affix to A, CA, KA roots and added **nuu**, **nucu**, and **nuku** as root forms, producing the system in the Paradigms (which does not agree with LIP).

2/3/2018 Replaced the lexicography section with a formal proposal (which contains a lot of the old text). This is accompanied by some grammar modifications, notably allowing more PA words, but it is mostly a matter of organization. The copy of the PEG grammar which is embedded in this document is now somewhat out of date: one should look at the commented PEG source available on my page.

12/18/2017 Put a description of the tests for the start of a predicate word without lookahead in the main text. Copy editing. I should note that the embedded text for the new parser should be replaced by now: considerable edits and debugging have happened. I made a small tweak which once again allows many of the weird possible borrowings with many vowels before the CC pair; very likely all of these should be banned, but the rule which was banning about half of them was not the right context in which this should be done.

12/16/2017 Added footnotes about the actual method of detecting the start of a predicate word, and the fact that this method enforces the rule about y-hyphenating initial CVC djiifoa without any special additional tests.

12/15/2017 Some light editing.

12/14/2017 I have merged some bits of the original sections on phonetics and orthography and the phonetic shapes of words into the Phonetics Proposal and eliminated the old sections, and I have read through the later parts of the reference grammar and made corrections determined by the Phonetics Proposal (of course I may have failed to notice some that need to be made!)

12/13/2017 This now contains a more or less readable version of the now official PEG based on the phonetics parser. This embedded text will not be updated often (it is rather laborious to do so) but it is available on the site in the latest version as a PEG. Its embedded comments are probably worth reading.

I still need to reconcile the phonetics proposal with the rest of the reference grammar!

12/12/2017 Massive provisional modifications to the text to temporarily signal the changes caused by the phonetics proposal. I am inserting the phonetics proposal as a section (superseding the original phonetics description;

I'll eventually reconcile them into a single text), and inserting the new PEG as an appendix without eliminating the old commented PEG. I'm planning to develop a "literate programming" format so that I can insert the new PEG, with LaTeX comments in pretty format, and build Python parsers straight from this document!

12/3/2017 Fixed a bug which allowed broken mono vowel pairs in borrowed predicates.

12/2/2017 Update with fancy punctuation improvements. Quotation marks (double) can be used in classes LI and LIE (in class LI they need to be directly adjacent to the words **li** and **lu** (on either side of the word) and a **li** must be so adorned iff the matching **lu** is so adorned. In class LIE, double quotes may appear before and after the quoted text and will appear before if and only if they appear after. The words **kie** and **kiu** may be adorned with directly adjacent parentheses (and in a given matched pair of these words, both or neither will be so adorned).

I did not, but should, make these updates in the alternative parser.

This update has the unintended effect that it is legal to prefix (but not postfix) a double quote to other alien text items (the bodies of foreign names with **lao**, and foreign or onomatopoeic predicates with **sao** and **sue**. TA

8/5/2017 Installed the alternative openers and closers for subordinate clauses (JIZA etc., GUIZA etc.) in the main version `loglan.py` as well as the alternative version. This makes class `argmod1` more elaborate. Being able to tag these pieces of text as strange is not a bad thing, I suppose.

5/23 More work on closures and development of alternative parsers to detect unintended parses in the Leith novella and other Loglan texts. Information about the grammar defined by the alternative parser `loglannew.py` is included. This update covers a lot of experimentation: it would not be surprising if I've missed something.

5/10 continuing work on closers: **guu** now works somewhat differently. The "empty termset **guu**" no longer exists. The closing **guu** is no longer part of the termset; it is now usually supplied by `barepred`, or possibly by other contexts. `kekpredunit` is closed by **guu**. In practice, **guu** does the same thing it always did, but this is realized by the parser in a different way. Now **gui** closes afterthought connected sequences of subordinate clauses rather than individual such sequences, and **guia** is not needed.

5/9 Added new closers. Also added the fix to `linkargs1` which removes the need for double closure if an instance contains both **je** and **jue** clauses.

5/8 Removed all **gap**'s closing names, which have unambiguous right clo-

asures already (a final predunit in a serial name might need to be closed with *gue*, but that is unusual). Further refinement of placing of gaps where names are also involved.

5/7 Various fixes to utterance structure. Negation with full sentence scope can now be applied to *sen1*'s, obtaining *sen1*'s.

5/5 Some fixes to avoid problems with VCV literals. One must pause before them when they appear as words, but not when they appear as units in acronyms (that would actually be bad). Incidental repair to NI2 with regard to IE.

Debugged some problems with NO at the beginning of utterances.

5/4 unified the identical FinalConsonant rules. Fixed stringnospaces so that commas can appear in non-final position in blocks of alien text. Additional tweak to punctuation in NOUI. Added the Greek legacy vowels Vzi.

5/1 In what looks like a major move, but isn't really, eliminating unused vestiges of the difference between sentence predicates and description predicates. The 1990's elimination of the rule that metaphors could not have kekked head modifiers left the restriction that the metaphors with kekked head modifiers could not have further initial modifiers appended with **go**. This restriction is lifted, and the number of rules is significantly reduced, as parallel forms of sentence predicates before **sentpred** itself are simply eliminated in favor of the description predicate classes.

4/28 Moved **ie** into class SA. Also fixed comma2 so capitalization is enforced across optional breaks.

4/27 Added the option of articulating a y-hyphenated CVCy djifoa as CV-Cy as well as CVC-y. For some reason, I already thought I could. This allows the syllable-final allophone of h to be avoided. Also fixed a problem that it was failing to require a pause between a (C)VV cmapua and a following vowel initial predicate.

3/18 There is another very subtle 3/18 fix, allowing freer use of links of the form **je pa** or **jue pa** without unnecessary closures. It is entirely a small change to the text of JELINK and JUELINK without explicit comment in the text. Just a bug fix. Such links are not found in existing texts, as this is a new proposal in TLI Loglan, but I have tested it a bit.

3/18 experimentally allowing conversion of PA roots with **nu-** and negation with **-noi**. The conversion and negation forms for KOU words remain the same. I believe that both of these moves are provably harmless. Negation with initial **no-** as for the KOU roots has significant phonetic danger attached.

3/9/2017 fixes a bug with use of **kouki** connectives with sentences or predicates: the **kou** was mistaken for a modal. Also adds **ciu** and **mou** to KOU to allow construction of words given in Paradigm K. This allows new forethought connectives as an unintended side effect.

2/11/2017 added a footnote about **kia**, the word cancelling operator. I was sad to have to correct my translation of the original sense of **Na crina!** from the satisfyingly weird “Be a raindrop now!” to the accurate “Be rained on now!”. The new observative meaning is of course “It’s raining now!”.

11/19/2016 Supports cleanup of the grammar which should have no effect at all on parser behavior. The rule `__LWbreak` is replaced by `!(connective)` everywhere, and most occurrences of `!(Oddvowel)` are eliminated. These are not needed because VV attitudinals are now required to be pause-initial. We do not thus raise alarms of a `cmapua` syllable is followed by an odd number of vowels: we usually raise alarms if it is followed by a vowel at all. The only place where `Oddvowel` is needed is in the definitions of the vowel units of `cmapua` words and in the definition of letterals.

11/5/2016 Moved the version notes section to this location. Moved a lot of the version notes to the appendix. Removed the old annotated PEG grammar (to another document); having both was very confusing when editing.

11/4/2016 Tiny change to the grammar of **period** removes a rare error.

10/17/2016 General editing to remove anachronisms.

10/1/2016 Added some semantic notes about topics of interest for logical analysis of Loglan sentences.

9/24/2016 The rule about inserting an explicit pause between a finally stressed `cmapua` and a following predicate is restricted to consonant initial predicates. With a vowel initial predicate, the pause is already required and need not be indicated by a comma. This also fixes a bug.

9/14/2016 Dealt with cleanup items, and also made some changes in the grammar of serial names required by the recent change of the way phonetic pauses are handled.

Monosyllables ending in o or i cannot be followed by another copy of their last letter. This doesn’t prevent vowel sequences but changes their grouping.

The strong quotation function has been simplified to be exactly parallel to the LAO construction, using **y** as the separator of alien text blocks.

inverse vocatives were pulled out as a separate class.

Serial names had to be debugged; they meet the English specification in the grammar without essential changes.

In a `gasent2`, the terms after the **ga** may optionally have the first one

separated from the others by **gio**, and the tags **argumentA**, **argumentB**, etc. will fall on the sutori arguments, not the first one, whether **gio** is present or not. This means that no more than four non-case-tagged arguments are needed in class terms.

9/11/2016 Finished reannotation of the latest PEG grammar, and moved the old annotation appendix to the end (it may still contain valuable remarks).

The new appendix contains various remarks about projected final cleanup of the grammar. At this point I am regarding it as a deliverable, apart from those cleanup items and of course any actual bugs. I am not particularly interested in making further major modifications until there has been some discussion of this version.

3 A brief report, Fall 2016

I believe that at this point I have largely achieved what I set out to do in 2013. In this (hopefully) short report I will set out what I think I have done and indicate what I am likely to turn my hand to in the future.

My belief in 2013 (which had been my belief for many years; it was in 2013 that I set out to do something about it) was that the state of the language definition and the software that we had inherited was unsatisfactory and an overhaul was necessary.

I addressed this by writing a new parser, unifying the levels of phonology, lexicography and grammar in a way that LIP did not. This addressed several problems. The lexicography was underspecified by the existing documentation: it was defined internally to LIP by a demonstrably buggy lookup table, and nowhere systematically defined in our Sources. There were some demonstrable ambiguities in the language caused by unintended interactions between the lexicography and the grammar.

I chose to use PEG formalism (Parsing Expression Grammar, due to Bryan Ford) following the example of a Lojban worker. This does mean that I do not have automated ambiguity checking of the kind supported for certain restricted BNF grammars, exploited by the original Loglan grammar writers. A PEG grammar is always unambiguous in some technical sense, because a priority scheme is used to choose between alternatives, but care must be taken in the ordering of alternatives to be sure that the intended alternative is always chosen.

The phonology is now precisely defined in a way which is integrated with the lexicography and grammar. This definition created very little which was novel: a complete reparse of all the words in the dictionary uncovered only a handful of words that needed to be corrected, and most of these were questionable from the standpoint of the earlier language definition as well. Doubled vowels which induce stress were eliminated from borrowings, which caused a change in one word. The reform which eliminated the **slinkui** test was refined, but it appears that the refinement was already intended by the original workers, since the dictionary appeared to be in agreement with the refinement already!

I defined the Loglan syllable, which had never been exactly defined, but which was clearly an important concept, notably because of the role of the penultimate stressed syllable in the definition of predicate words. I imposed syllable structure on name words, and required (as is suggested in Loglan 1

(1989)) that syllabic (“vocalic”) consonants appearing in names be doubled. This caused spelling changes in a few names, and phonetic changes in a few names because doubled non-syllabic consonants and groups of three non-syllabic consonants in final position became illegal.

As a side-effect of having precisely defined the syllable, it became possible to refine the parser to allow explicit expression of syllable breaks and stresses. This made it possible to define a phonetic transcript mode of the parser, in which no breaks appear except actual comma-marked pauses and all stresses are marked. This did require that I ensure that whenever a phonetic pause was required, an explicit comma pause could be written, and that no whitespace was required that did not represent a pause.

The availability of phonetic transcripts made it possible to do real investigations of the false name marker problem. The end of the Loglan name has always been easy to determine, since names are the only consonant final words, with the consonant always followed by a pause. It would be equally easy to determine the beginning of a Loglan name if one always had to pause at the beginning as well as the end of a name, but this seemed awkward to our Founders. It is not required that one pause before a name if it is preceded by a name marker word (such as **la** or **hoi**). This created problems with determination of the left boundary of a name if a phonetic copy of a name marker word appeared in the name (and forbidding **la** in names is very awkward). The general solution is that names which contain phonetic copies of name marker words must themselves be marked, and occasions where names can appear unmarked except by a preceding pause are very restricted: unmarked vocative uses of names were eliminated, because they cause disastrous difficulties, as the availability of phonetic parsing allowed me to determine (it was already known!)

Another reform related to names was a cleanup of serial names, requiring that predicate components of serial names be marked (with **ci**) and eliminating the need for two grades of pause to distinguish serial names from short sentences. The availability of phonetic transcripts made it much easier to test our solutions to the serial name and false name marker issues (which have additional technical detail which can be seen below).

James Jennings made a very interesting remark in recent conversation on the Loglanist list to the effect that the definition of Loglan word classes in terms of patterns of consonants and vowels was the original sin of the language and has led it into endless difficulties. It is an interesting view: my take is that the original decisions along these lines, combined with the later

changes in the morphology of predicates, certainly made the specification of the phonology quite complex, but they also give the language a distinctive flavor which I find interesting.

Except in the narrow area of names, I made very few changes in the phonology, and almost none which actually affected existing words. This was not true at the next stage, the lexicography of the language. By this I mean the lexicography of structure words: the large classes of predicate and name words are actually defined at the level of phonology. I needed to make precise definitions of certain classes of structure words, and these definitions often do not coincide with those implicit in LIP, though the commonly used words are supported.

An important issue is exactly what a **word** is. There is a definition in NB3: a word is a grammatical construction in which one cannot pause, effectively. Unfortunately, there is a counterexample to this in Loglan 1 (1989): one is allowed to pause after a borrowing affix in the middle of a long predicate word! I introduced other exceptions to this: my language definition allows pauses (with some restrictions) in the articulation of PA and NI words (compound tenses and numerals). But it is a useful concept even if it has exceptions. Lojban has achieved a state in which (at least in theory) one can pause anywhere in a stream of structure word syllables without changing the meaning of the utterance. I do not believe that this is the case in TLI Loglan: we do have some multisyllable structure words in the internals of which one is not permitted to pause (enumerated in the reference grammar). The most famous example, the **lepo** words, has been fixed.

I gave precise definitions to the large classes of compound structure words (PA and NI words notably, and by extension word classes into which PA or NI words can enter as components). These definitions do not agree exactly with LIP, but they do support commonly used words. I forbade **noi**-initial compound tenses (PA words) which can be shown to lead to ambiguity (because all other uses of **noi** are word-final); I introduced a different construction of negative compounds to replace the forbidden words (**noipacena** becomes **panocena**).

The classes of logical and sentence connectives with suffixed PA words (APA and IPA words) presented serious difficulties. Lojban has forbidden them. I did not: they are common in existing Loglan text, and the IPA words specifically (words like **irau**) are very common. What I did, which follows a style which JCB uses in NB3 though not perfectly consistently, is require that such words be closed with an explicitly written comma pause, and added

the additional alternative of closing them with **-fi**.¹ The difficulty is that the suffixed PA attached to such words could otherwise be understood as the initial “preposition” in a sentence modifier. In parsing Alex Leith’s *Visit to Loglandia*, I found that situations where one had to add pauses either before or after a PA unit to clarify its relation to a preceding logical connective or **i** were not uncommon. My solution allows ancient texts to be fixed with pauses, and supports and perhaps positively suggests use of **-fi** to close such words and remove any danger of ambiguity.

The status of acronyms presented serious problems. We had been reduced to requiring that a sequence of letteral pronouns be separated by explicit pauses to prevent them from being mistaken for an acronym, which struck me as absurd. A minor feature of the language should not inconvenience a major feature in this way. My solution (which seems sensible semantically as well) was to forbid multiletteral pronouns² and reclassify acronyms as **names** rather than predicates, which has the effect that in their main use they are left marked by a name marker and right marked by a pause. Acronyms used as dimensions in NI words are left marked with a new marker **mue** and also required to be followed by a comma marked pause. The effect is that it is impossible for a letteral pronoun or numeral to be confused with a component of an acronym, and it is not necessary to pause between letteral pronouns appearing as successive arguments of predicates.

A further issue which I class as lexicographic is the status of constructions which incorporate alien text into the language. I made a proposal to change the strong quotation mechanism, which I realized subsequently was basically the same as a phonetic solution already given for **lao** names, originally Linnaean names but now a general construction for foreign names (following an excellent observation of Steve Rice). A multi word foreign name has the form **lao Albert y Einstein**, where the blocks of alien text are set off with phonetic pauses and separated by **y** as shown (explicit commas not necessary). The original worker suggested that the **y** appear in speech but not in writing; my parser requires that it be written. My solution for strong quotation (replacing an original proposal which is neither BNF nor PEG parsable) is basically the same: “War and Peace” becomes **lie War y and y Peace**

¹Originally I closed APA connectives with **gu**, but this caused conflicts with other uses of **gu**.

²LIP permits multiletteral pronouns but it seems quite clear to me that JCB’s discussion of letterals in NB3 does not support this. I do allow single letters with one-digit numerical subscripts as pronouns.

(my original proposal used **cii** instead of **y** and had complexities for nested quotation: this has now been eliminated). The use of **lao** for foreign names in general has the further advantage that we can require foreign names with **la** to be spelled as they are pronounced: **lao Einstein** vs. **la Ainctain**.

Finally, the grammar which we inherited from LIP was quite explicit and fairly readily translated into a PEG. What did require cleaning up was the ordering of alternatives so that the correct one would be chosen first. The fact that the original grammar was ambiguity checked ensured that there **was** a choice of order which would work!

The major change in the grammar proper which I made in the end, though I resisted it initially, was the complete elimination of PAUSE/**gu** equivalence. Pauses surplus to phonetic requirements³ are always read as free modifiers in the present grammar (which is also the case in Lojban). I did attempt to implement the use of pauses in certain situations as **gu**, following JCB, but I found in the end that it was impossible. Most uses of this in the NB3 corpus were eliminable in favor of use of the special closures **gue**, **gui**, **guo**, **guu**. Many of the uses which JCB employed were clearly impossible: it cannot be the case that pauses **next to “gu”** are semantically significant, as a tendency to pause next to these words must be regarded as inevitable, due to their function, and parsing many of the NB3 corpus examples clearly depends on understanding such pauses as **gu**. As a result of this restriction which I placed on the equivalence, my version of PAUSE/**gu** equivalence was so different from the LIP version that it was easier for me to parse the Visit to Loglandia with no such equivalence at all than with the one I had implemented.

There are some other significant changes in the grammar which do not affect existing text, or not very much, but which would affect complex utterances.

The construction **lemi hasfa** is now grammatically parallel to **le la Djan, hasfa**, which was not true in the original grammar (**lemi** was a word construction), though speakers might very well regard these as analogous. Similarly **lena** is no longer a word. I believe that this is also the case in Lojban. This grammar modification allows some utterances not allowed in TLI Loglan, and probably still deprecated: I am not very fond of the possessive **le la Djan, hasfa** and similar things, but note for example that as one could

³and now some of those which are phonetically required: pauses before logical connectives are now freemods.

say **lemina hasfa** before, one can now say **le la Djan, na hasfa**, “John’s present house”.

The attachment of common argument lists to logically connected predicates, as in **Mi cluva, e donsu le bakso guu, la Meris**, is handled by an elegant and highly left recursive rule in the trial.85 grammar, which is basically impossible to realize in a PEG. Also, the behavior of the ACI series of connectives when linking predicates is simply weird in the trial.85 grammar. These two issues are handled together in my grammar in a somewhat different way which is not likely to be detected by a user. The use of ACI connectives is much more sensible (they simply link more tightly than the standard A connectives); the possibility of linking common final termsets to logically connected predicates is more limited in my grammar than in the original trial.85, but in a way which is unlikely to limit the possible range of utterances in practice.

The notorious **lepo** problem was solved, though not because I was particularly unwilling to accept the difference between **lepo sucmi ditca** and **le, po sucmi ditca** found in Loglan (1989). The problem I discovered was deeper. The trial.85 grammar severely restricted the use of predicates of the form **po mi blanu** (event predicates built from sentences) in a quite unreasonable way – such predicates could only appear at the very top level and could not for example enter into metaphors at all. I allowed such predicates to be of class `predunit1`, which seemed inevitable on reflection. This then led me to the view that the phrase **le po mi blanu** should not be viewed as containing a predicate **po mi blanu** (in Lojban it is viewed as containing such a predicate and one sometimes has to close such an expression twice, once to close the predicate and once to close the description). I ruled that the constructions `LE PO SENTENCE` and `PO SENTENCE` are disjoint (the first does not contain an instance of the second), and both closable with `GUO`. Where a metaphor `LE (PO SENTENCE) PREDA2` might seem to introduce danger of an ambiguity, I require the use of `GE`: `LE GE PO SENTENCE GUO PREDA2` is required if one wants the `GUO` to close `PO SENTENCE` rather than closing the entire description. This will not affect existing text with complicated predicates `PO SENTENCE` starting a metaphor because the trial.85 grammar did not allow constructions with a modifier of the form `PO SENTENCE` (so there is no such text). It is a consequence however that there are no longer **lepo** words. `LE PO SENTENCE GUO` and `PO SENTENCE GUO` are two separate constructions, and even an explicit pause does not break the first one. **le, po sucmi ditca** and **lepo sucmi ditca**

both mean “the swimming lesson”, while **le ge po sucmi guo ditca** means “the swimming teacher” (teacher of events of swimming) and this can be said more conveniently as **le poi sucmi ditca**, where the new word **poi** implements the old short-scope **po**.

I have implemented in my current parser a major new proposal allowing clearer handling of LEPO clauses (and incidentally of PO predicates) by supplying several new closure operators for such clauses. I have given explicit examples of the use of these closures in rephrasing some especially nasty examples of nested GUO closures in the Visit.

I made some changes in the less used of the basic sentence structures. In the gasent construction **PA predicate termset ga terms**, I made the **ga terms** optional (so a sentence like **Na crina** is read as **Na crina ga ba**, an observative, “It is raining”, not as an imperative “Be rained on!”: imperatives are restricted to being untensed subject free sentences) and further required that the final **ga terms** contain either exactly one argument or all the arguments in the sentence. The motive behind both of these rules is that the appearance of the **ga terms** should not cause a radical re-reading of the sentence on the fly. Semantically, it is also a positive good to recover the observative sentences. I also cause the grammar to recognize that a sentence in which all arguments before the predicate are modifiers is an imperative. I am very critical of the prescription in our sources that the interpretation of sentence forms in which final arguments are fronted should depend on knowing what the last argument of a predicate (which might have many little-used arguments) might be: I suggest alternatives below.

My intentions in all of this were conservative. My intention was to give an adequate language definition, supported by current and readily maintained software, supporting a language which would be intelligible to a speaker of 1989 Loglan if such a being existed. I can present actual evidence that I have done this: the parse of the Visit to Loglandia required relatively few changes to the text, mostly of highly stereotyped kinds (some were frequent but routine, such as doubling continuants in names and inserting pauses after APA words).

My provisional parser and the dictionaries I am maintaining are freely accessible and I hope reasonably easy to use. The parser is designed so that the user can examine the structure of the parse with more hope of seeing that they have parsed the sentence correctly. With LIP, the output format for a complex sentence is unintelligible [people can only count so many parentheses]: I suspect it was more often used as an oracle (does this parse?)

without proper attention to whether the parse was as intended, since it was very hard to tell.

At this point, I am done with the language definition! I am sure that minor bugs will pop up and I'll deal with them as necessary, but I believe that I have presented a workable grammar. I briefly enumerate things I am intending to work on further.

I should think about doing more translations.

On the note of things parsing as intended, I have a project of going through the Visit and checking whether **lepo** clauses are closed where intended. This is hard work because it requires that one actually read the text and determine what it means! Control of closing **lepo** clauses in the right places is a remaining major issue in the grammar.

I would like to survey the dictionary looking for semantic issues. Some words may have odd argument orders or missing arguments which could reasonably be revised. I have parsed all the words in the dictionary – they are all well-formed under the current phonology, at least!

A huge further project would be to implement logical transformations of Loglan sentences in software. I write theorem provers: I have relevant technical expertise to do this. Loglan incorporates features of natural languages which have traditionally been avoided in formalized logic, such as logically connected arguments and anaphora. Implementing permitted logical transformations of Loglan sentences might lead to issues of interest in formal language and/or natural language processing generally, quite independently of this specific language. Of course this project, if I undertake it, will lead to further discussion of the semantic aspects of the definition of the language.

I should learn to use the optional case tags. Case tags, optional or numerical, are another feature which may lead to considerable difficulties in a formal implementation of reasoning in the language!

4 A catalogue of issues

In this section, I summarize major issues which I perceived when I set out to overhaul the language in 2013, and issues which arose in the course of carrying out the overhaul. I describe each issue and give a high-level account of my solution(s), official and/or provisional.

This section is intended to be read by someone with prior familiarity with the language.

As we reiterate in the last point, this is not necessarily an exhaustive catalogue of Issues. Others will appear with less fanfare in the reference grammar and the appendices.

general intentions: My general intentions are conservative. I aimed to create a precise language definition for a language which would be intelligible to a speaker of 1989 Loglan (if such a being existed) apart possibly from some necessary local changes to less-used features. I did not want to engage in a fundamental philosophical overhaul of the language or add major improvements at this time.

I now have extensive concrete evidence that this is what I have actually achieved. I have been parsing Alex Leith's novel "A First Visit to Loglandia", and I find that with attention to a few stereotyped issues, the text parses, apparently much as intended. The same is true of the other snippets of text on our web page.

institutional: The Loglan Academy had been moribund since JCB's death, and I had not been in communication with the president and board of trustees. I have revived the Academy, and I have been in contact with the board, which approves of my activities.

Related to this is the intellectual property policy of the Loglan Institute and its relations with the sister language. My view is that we should retain the claim of copyright over our major documents, but allow free use for non-commercial purposes by anyone who is interested. The reason for us to maintain at least theoretical ownership of our intellectual property is that we do not want independent workers to claim that things are TLI Loglan which are not.

I am perfectly happy to refer to our language as TLI Loglan and acknowledge that Lojban and some other related languages are "Loglans"

and are related to our project. After all, they are related. I try to maintain good relations with the other language(s); after all, some of our active Loglanists have come over from Lojban. I am for example friendly to adopting linguistic devices for incorporating Lojban text into Loglan utterances. Of course, usually when I say Loglan I mean *our* Loglan.

legacy software, documents, and language definition: In general terms, I have felt for a long time that the status of the Loglan language definition and basic claims which we make about the language definition was unsatisfactory.

The grammar was defined and publicly available (in trial.85, the BNF grammar which appears as the last appendix) and this grammar was in a sense formally verified as unambiguous. This was less impressive than it appeared. The problem is that the orthography and lexicography were not formally defined, and in fact their status was unsatisfactory and demonstrably created residual ambiguities.

We had a fairly good description of the orthography and phonetics in the previous documents. In making this fully precise, I found few occasions where I needed to change anything, though many occasions when I needed to make them more definite.

The lexicography (in particular definitions of large word classes such as A and PA and LE) was in a quite unsatisfactory state. There was no formal definition of the word classes except implicit in a non-human-readable lookup table in the LIP software which is demonstrably buggy. I made complete formal definitions of the word classes which do not agree precisely with the word classes as defined in the software and earlier documents, but do support the words frequently used. In some cases, changes in the large word classes had to be made to avert problems; in other cases, I gave a general definition of the class which worked for all practical purposes which demonstrably did not agree with LIP on the extent of the word class in question: generally, my definitions tend to allow more words.

The grammar proper as expressed in trial.85 was the best documented and implemented part of the language, though various improvements were needed which will be discussed under separate headings in the catalogue of issues. My first pass at implementing it was indeed to directly

translate the BNF grammar in trial.85 into PEG notation (reordering alternatives as necessary to avoid incorrect preemption of intended alternatives by earlier ones), and this is still often visible in the current format (the reader is invited to compare the PEG appendix with the trial.85 appendix).

The legacy documents, Loglan 1 of 1989 and NB3, both superseded in details by decisions recorded in Appendix H, still both needed to be consulted for motivation of features of the language and in order to support decisions when I had to make precise something that was unclear. I note that the corpus in NB3 has been enormously valuable to me for testing purposes (though of course I have had to revise the corpus to reflect changes made later by me and by others). The dictionaries were mostly satisfactory; I am very pleased with Peter Hill's software which allows me (and indeed would allow any interested individual worker) to easily maintain and generate new HTML dictionaries. I add here that I am enormously impressed with the work and thought of JCB and others which went into Loglan 1 of 1975 and 1989, Notebook 3, and the dictionaries. These books are essential to understanding Loglan; at any rate I have created nothing that would replace them. Ultimately, it might nice to have revised versions.

The old parser LIP was not available to me in a form which I could update. It has other weaknesses: it does not present parses in an easy to read format, so historically it seems to have been used as a yes/no oracle (can this be parsed or not?) rather than to check whether something parseable was parsed *in the right way*. My new parser presents parses in a more readable fashion.

the decision to use PEG to parse Loglan: I followed the example of a Lojban worker in deciding to produce a new parser using PEG (Parsing Expression Grammars) a formal method of generating parsers due to Bryan Ford. PEG grammars are fairly easy to write (at least for me) and more powerful computationally than the BNF grammars used by the previous generation of Loglanists. On the other hand, there is no clear analogue to the automated disambiguation checks which exist for BNF grammars of specific restricted forms. A PEG grammar is in principle always unambiguous, because it uses a priority scheme to determine which of a list of alternative local parses to attempt first;

the analogue to failures of ambiguity is the choice of the wrong alternative in crucial points in a parse, which is more difficult to check for automatically. I have had to reorder alternatives in many rules in the trial.85 grammar for this reason.

On the other hand the greater logical power of the PEG primitives was essential to my basic goal, which was to have a single grammar of Loglan from the level of letters upward, with no preprocessing at all. Presenting the rather baroque phonetic rules of Loglan predicates (in particular) as a BNF grammar of the sort which can be automatically disambiguated would have been difficult or impossible.

I did in any event make the decision to write a PEG to implement my overhaul of Loglan. This does mean that there is no analogue to the automated disambiguation that the previous Loglanists used in checking their grammars. What one does want to check (and I have manually checked this from time to time) is roughly that in each list of alternatives there is no possibility of an earlier alternative in a list of alternative forms of applying to a proper initial segment of an instance of a later alternative: this is the commonest way that an unintended parse happens, and it is what I mean by “preemption” above. I wrote my own PEG engine to implement my grammar; I have contemplated writing automated tools which would warn the user when there is danger of an unintended parse, but have not yet done this. I did include a termination checker in my PEG implementation⁴; if it raises no warning, the parser is guaranteed never to go into an infinite loop (which can happen otherwise).

orthography and phonetics: An early decision was to eliminate the letters **q,w,x** from the language, outside of embedded alien text. Progress in this direction was already being made in the 1990’s, when predicates containing these letters were eliminated.

To implement the baroque definition of predicates, it was essential to formalize the definition of the Loglan syllable. No precise definition of this notion is given in NB3, in spite of the important role that the notion of syllable already played in the language; our specification can be supported at every step by remarks in NB3, and words from the dictionary do parse sensibly.

⁴In the ML version only, so far

Having defined the syllable, we made the further decision to require that names be resolvable into syllables as well⁵. This in itself did not lead to any need to change the orthography of any names in the corpus. But we did require that all syllabic (“vocalic”) consonants be written as double consonants, which did require changes in spelling of many names in the corpus. It should be noted that this spelling rule is actually suggested in a note in Loglan 1 (1989). It can further be noted that in parsing Leith’s Visit I ran up against the fact that the Loglan syllable cannot end in more than two consonants (often one of these is a continuant, and we can fix by doubling the continuant, as in **la Marrks** or **la Hollmz**) and doubled consonants other than continuants are not permitted, thus **la Betis** (already the attested spelling), **la Oto**.

An analysis of stressed syllables was also required by the definition of predicates. Consideration of stress caused us to make the official change to the language forbidding the stress inducing vowel groups **aa**, **ee**, **oo** in borrowings. It turned out that this required us to change just one borrowed predicate, **alkooli**, and we changed it to **alkoholi**, which is an improvement!

We adopted a different rule for grouping long strings of vowels in borrowings or names than any which appears in Loglan 1 or NB3, based on the allowed and optional monosyllables and working from the left.

There is one new phonetic rule, forbidding a syllable from ending in two consonants the first of which is not one of **mnlr** and the second of which is one of these: such an appendix to a syllable would have to be pronounced as a separate syllable. This rule only affects names and borrowings, and seems to be phonetic common sense.

In summary, I believe the implementation of the phonetics is almost exactly as in the original definition. I based my work on borrowed predicates and names on a precise definition of the Loglan syllable. I do require that names resolve into Loglan syllables (which does not affect any names appearing in the corpus). Because of this, I require that names with **la** be written as pronounced, while names with foreign spelling may be written with the **lao** form for foreign names. Early in this work I proposed to the Academy that doubled vowels which force stress not be allowed in borrowings, and this was approved. I

⁵Does Lojban do this as well?

require that syllabic consonants be doubled, which does affect spellings of several names in the corpus, but which is also explicitly suggested in Loglan 1. I adopted a different rule for grouping long strings of vowels in names or borrowings than is given in the sources. I proposed and the Academy accepted a clarification of the phonetic maneuver that abolished the **slinkui** test, but it appears that the modification may have been a restatement of the original Academy's actual intentions, as the change did not materially affect the dictionary.

I have recently parsed all the words in the dictionary, and fixed the very few words whose form was incorrect. I also finished the elimination of the letter X.

the decision to produce a phonetic parser: I had thought from the outset that a phonetic parser for Loglan would be useful. Since I had to define an exact notion of syllable in order to even define the penultimate stress criterion for predicates formally, and engage in often rather indirect deductions about stresses to determine whether strings met the criteria, it occurred to me that if I added explicit notation for syllable breaks and for stress on syllables, I could develop the phonetic parser as an operating mode of the parser I already had.

The idea is that one and the same parser can parse sentences in traditional Loglan orthography or “phonetic transcripts” of sentences, in which no whitespace appears unless it is an explicit, comma marked pause (and mixed forms as well). Some design decisions were needed to make this work. It was necessary to ensure that a comma marked pause was legal wherever a phonetic pause was actually required. It is necessary to mark stress explicitly if the whitespace at the end of a predicate is not expressed. On the other hand, the parser does need to be able to deduce the stressed syllable in a predicate whose end is indicated by a space, and check that it is legal to stress this syllable. A text is a phonetic transcript if all whitespace is comma marked and the essential stresses are marked; of course an exhausting phonetic transcript may include all explicit syllable breaks and stresses.

I chose to use the hyphen - as the syllable break, which precludes the use of the hyphen to abbreviate the spoken hyphen y which is attested in our founding documents. The close comma used for explicit syllable breaks between vowels in our founding documents is replaced by the

hyphen. In general, **punctuation is not to be pronounced** (except insofar as it indicates a pause or silence). I use ' and * for ordinary and emphatic syllable stress. These are used instead of syllable breaks after initial or medial stressed syllables and may appear after final syllables as well. I note that emphatic stress can be added to otherwise orthographic text to indicate rhetorical emphasis.⁶

The development of the ability to parse phonetic transcripts means that it is actually possible to express the stress rule that finally stressed cmapua before predicates must be followed by explicit pauses⁷. It has also made it possible to effectively test solutions to the false name marker problem.

In any event, there is a working phonetic parser for my provisional Loglan grammar, which is the same as the usual parser, but applied to different strings. I do not believe that the sister language has a phonetic parser at all (other than toy partial implementations).⁸

I have introduced JCB's marker # for end of utterance or change of voice. This is not a piece of punctuation in the language (it cannot appear in a quoted or parenthesized Loglan utterance). It can appear quoted or in alien text without risk, it appears. What it allows me to do is mark changes of voice in texts I am processing (including the same speaker stopping and then starting again) without a line break. Extensive use of this is not encouraged: what the parser is doing with it is terrifyingly recursive.

definitions of specific word classes: Certain word classes are not completely defined in our founding documents.

The truly baroque classes are PA (tense/location/relation operators) and NI (numerals/quantifiers).

The classes of logical connectives (A and kin) acquire complexity because they can be suffixed with PA words.

⁶Recording a question which Cyril asked me, there is no grammatical or phonological difference between stress and emphatic stress. JCB says in L1 and/or NB3 that the distinction between the two forms of stress is phonemic, so I provide both.

⁷I cannot see any way that an utterance could fail to parse under LIP due to this major rule of Loglan phonology; it can happen under my parser

⁸I may of course be wrong; I have no intention to run down **la Sorme Lengü**.

Complete definitions have been given, which do not coincide with the definitions implicit in LIP (they are often more liberal) but which meet the requirement that the language is intelligible; there are very few cases where changes are made which forbid words appearing in previous texts. There is one such case to be specifically noted; the structure of logically connected tenses attested by examples in NB3 demonstrably led to ambiguity, and I had to make a change in it to avert this.

In both PA and NI words (unlike any other multisyllable words⁹) we support the ability to pause in the middle of words (with some restrictions). Words of these classes can potentially be very long, and pausing to articulate them is quite natural.

the APA issue: The words like **apa** create no end of trouble. The problem is that there are situations where **apa** and **a** followed by **pa** both make sense and do not have the same logical effect. The solution adopted is to require PA suffixes in APA, CAPA, IPA, ICAPA words, to be closed either with an explicit pause or with the syllable -FI.¹⁰

We have eliminated the ability supported by LIP to suffix PA words to KI and KA words.

There is also some semantic funny business about these words. The meanings assigned to **apa** and kin and grammatically similar words **erau** and kin appear to be reversed in terms of expansions with explicit sentence modifiers. We prefer to leave the meanings as they are. We could eliminate these words entirely: we do not because they are extensively used in the NB3 corpus, and because the words like **irau** are indispensable (changing them would affect lots of existing text) and present the same closure problems.

The problem of pauses to clarify A PA situations is ubiquitous in the Visit to Loglandia, our longest text. It is also clear that Leith was aware of the issue, and often inserts these pauses where needed.

structure word breaks: I am told on good authority that in Lojban there are in effect no cmapua words of more than one syllable: one is completely free to add pauses in the middle of a stream of cmapua syllables as one pleases without changing meanings. The defining characteristic

⁹except predicates containing borrowing affixes

¹⁰Originally I used -GU, but this conflicted with other functions of GU.

of a multisyllable *cmapua* word is that one cannot pause in the middle of it without changing the meaning of what one says or making it ungrammatical. Loglan has multisyllable *cmapua* words. In the case of PA and NI words, we do allow internal pauses under certain conditions (and such pauses are also allowed in complex predicates with borrowing affix components).

A phenomenon regarded as malignant by Lojbanists is pauses required as word breaks to terminate a *cmapua* (structure word breaks). The original solution to the LEPO problem had this flavor. We definitely have multisyllable words, but we have striven to minimize situations in which *cmapua* need to be terminated with pauses; we have arranged for PA and NI words not to be terminated by whitespace or even comma marked pauses (under suitable conditions). We certainly *allow* such breaks (a pause will definitively end a word except in the exceptional cases noted); the point is not to require them.

One thing we have done is removed the possibility of inserting whitespace in the middle of what is in fact a word.

An interesting point about the grammar which I had not fully realized until I was editing this document (though I must have realized it when I made the change in question) is that the provisional grammar now does not actually have any provision for structure word breaks proper at all. It used to be that a comma pause between structure words was automatically parsed as part of the preceding *cmapua*, terminating it. This is no longer the case; such word-breaking comma pauses are now parsed as free modifiers on the grammar level. Since the present grammar does appear to work, this suggests that the structure word break problem as such was solved.

acronyms and proper use of letterals: It was already evident to the previous generation of Academists that acronyms were a problem. The difficulty is that an acronym, as a string of letters and numerals, may grab a following letter (which may be a pronoun, and so grammatically crucial) or numeral. The solution adopted was to require explicit pauses between successive letterals appearing as arguments, which to my mind is absurd. A minor feature of the grammar should not affect pronounceability of examples of a major feature. Pronouns trump acronyms.

Acronymic predicates have been eliminated. Acronyms are regarded as *names*, which makes much more sense semantically. As names, they are front marked with articles or pauses, and must be followed by explicit pauses, so they cannot eat following literal pronouns, which can safely be pronounced one after the other without pauses with no danger of confusion. The other use of acronyms is as dimensions attached to quantity words, which are supplied with a new initial marker **mue**, always required (**mue** is actually an optional initial component of any acronym, mandatory for numeral-initial or one-symbol acronyms and for acronyms used as dimensions), and also must end with explicit pauses. Acronyms are defused as a problem. Multiletteral pronouns are also eliminated.

I have proposed to eliminate the vowel letterals of the form **afi**, **ama** in favor of **zia**, **ziama**. The old forms are still supported along with the new ones. I favor eliminating them in principle because they are phonetically very eccentric, but in fact they are ubiquitous in large texts and I have continued to maintain and indeed expand the ability of the parser to manage them. The new ones must be used if you want to use them as djifoa: **ziaytrena**, “A-train”, not ***afiytrena**.

strong quotation: The original strong quotation proposal was not BNF or PEG parsable. My new proposal in its simplest form is isomorphic to the previous Academy’s final arrangement for **lao**: **lie** may be followed by a sequence of arbitrary blocks of text separated by **y**. These arbitrary blocks of text must be set off by pauses from **lie**, **y**, and what follows, which may but need not be comma marked. Commas or terminal punctuation can occur in the blocks of text only if not followed by spaces. The more complex original version of this proposal has been simplified. The simple version of this proposal works well in the inherited texts.

serial names: The previous Academy decided to create a separate pause phoneme so that **La Djan Blanu** (the serial name “John the Blue”) would not be confused with **La Djan, blanu** (“John is blue”). This to our mind is absurd: having more than one pause phoneme is a major change which should not be introduced to fix a minor feature. We require instead that predunit components of serial names be introduced with the little word **ci** (**la Djan ci Blanu**) whereupon pauses cannot

be confused. Other refinements in the structure of serial names were required, notably in connection with the false name marker problem (the use of **ci** before a name with a false name marker appearing in a serial name had already been introduced), but this was the serial name Issue. Once again, all pauses are equal.

the false name marker issue: There are a small collection of words (the name markers) such that a name preceded by one of these markers does not have to be preceded by an explicit pause. There has been a struggle with the problem of names in which these markers (which include **la**) appear. JCB tried to rule out false name markers entirely, but it is inconvenient. We have solved this problem (the availability of the phonetic transcript mode of orthography has made it possible to test this). The key is to strictly limit the contexts in which unmarked names can appear. Unmarked vocatives were disastrous and have officially been eliminated from the language; vocative uses of names must be marked with **hoi**¹¹. Otherwise, the only unmarked occurrences of names are in serial names (where they are preceded by a name) and in certain descriptions, of the form **le blanu, Djan**. In the latter context, we require the explicit comma pause (the pause was always required, but did not have to be written). In both of these contexts, we require that the marker **ci** be inserted if the name contains a false name marker (which we define more precisely: an occurrence of a name marker word phonetically in a name is false only if the remainder of the name after the false name marker is a well-formed name). We require that **ci** always occur in a serial name before a name word which follows a pre-unit component (**la Djan ci Blanu ci Djonz**). We further provide that a name appearing after a name marker extends to the next comma pause (or whitespace): this creates an actual obligation to pause in certain contexts. The only caution in speech is that after a serial name ends whose final name word component is unmarked, it is probably advisable to pause soon after a vowel, at the latest after the next name marker (which is always permitted): and my latest work on this makes such pauses mandatory. An orthography which makes it possible to go from a name marker to the end of a later name without pause in an unintended way will be detected and the parse will fail. My confidence that this parser rule works is supported by experience in the parsing

¹¹They can now also be marked with **loi, loa, sia, sie, siu**

of the Visit, which contained a number of horrible examples. This rule is **not** one that a speaker should internalize: a speaker should follow some style rules which avoid creating such horrible situations.

the pause/GU/GUV issue (and free modifiers): JCB introduced the idea that many occurrences of **gu** and related words could be replaced with pauses. His application of this idea is clearly flawed; the parses of complex examples in the NB3 corpus are only saved by adopting the later-introduced device of words like **guu**. I refer to pauses of this kind as “grammatically significant”. I observed some obvious strictures when I was experimenting with this. A pause required by phonetic considerations (as before a vowel-initial word or after a name word) cannot be understood as **gu**. JCB knew this but has offended in this way. Further, it is absurd for such a pause to be understood as existing next to a **gu** word (or a relative like **guu**) LIP clearly does this in reading some of the horrible NB3 corpus examples, and it is absurd. It is clear from the function of such comma words that one would naturally pause next to them.

Free modifiers are not handled by preprocessing in my grammar. Instead, free modifiers are inserted as an option in most medial positions in grammar rules (not final positions). Pause was regarded as a free modifier where it could not be interpreted as GU. In final positions in grammar rules either free modifiers were experimentally not provided as an option, or only non-pause free modifiers (class **freemod**); this is how pause/GU equivalence was supported. **freemod** suffixes were allowed on instances of grammar classes which were in some sense “atomic”, so that a non-pause freemod attached to the end of a structure would in fact be attached to as small a final segment of it as possible.

At this time, I have completely disabled pause/GU equivalence. It seems to be just too easy for a listener to make a pause supposing that it closes one structure when it actually closes another one, or even performs a phonetic function or terminates a word. It also turned out that my implementation of pause/GU equivalence was so different from the one in LIP that parse failures caused by commas when parsing the Visit to Loglandia were ubiquitous. After removing pause/GU equivalence, parsing became easier, though commas were still sometimes an issue for other reasons.

PO sentence forms and the LEPO problem: The distinction between **le, po sucmi ditca** and **lepo sucmi ditca** cited in Loglan 1 (1989) is a scandal. I resisted eliminating it for some time because I have acknowledged that Loglan as it stands unavoidably has multisyllable *cmapua* words and the occasional need to pause to force a word break. I did in the end solve this problem for a totally different reason. Predicates of the form **po mi blanu**, PO words followed by a sentence, were in a ridiculous position in the grammar: they basically could not enter into any nontrivial predicate construction (they could not participate in metaphors). I fixed this by making such predicates *predunit1* phrases. This then created the menace of a need for double closure of **lepo X** clauses, closing first the constituent PO X predicate then the description. This is the actual situation in the sister language! I averted this by denying that LE PO X (GUO) contained any predicate PO X GUO; I made these two different constructions, both closed by GUO. The price of this, very seldom to be paid, is that in LE X, if X happens to begin with a PO Y (as in a rather unusual metaphor), this must be guarded by an initial GE (LE GE PO Y Z rather than LE PO Y Z).

At the same time, the short scope PO was replaced with different words. So the old **lepo sucmi ditca** stays the same, while the old **le, po sucmi ditca** becomes **le poi sucmi ditca** or even **le ge po sucmi guo ditca**. One can say **le, po sucmi ditca** and it means the same thing as **lepo sucmi ditca**, because there are no longer any LEPO words.

the ACI connectives and the shared termset problem: The *aci* series of logical connectives, as used between predicates, have really strange behavior in *trial.85*. I made them fully privileged logical connectives binding more tightly than the usual series.

If you want to say that “I love and like you” you do not say **mi cluva, e fundi tu**, because the **tu** is seized by **fundi**. **Mi cluva, e fundi guu tu**, where **guu** closes the argument list of **fundi**, allows attachment of **tu** as a shared argument of both predicates. Another example is **la Meris, cluva, e dons ta guu la Djan**, “Mary loves John and gave that to him”. The *trial.85* grammar has a lovely solution to free attachment of shared final termsets which is horribly left recursive and

cannot be implemented in a PEG. The solution which actually appears in the PEG has theoretical limitations on such constructions which will probably never appear in practice, because one can only attach further arguments to a predicate so many times.

recognizing imperative sentences: It is an error in the existing grammars that a sentence like **Na la Ven, donsü ta mi** (“at nine give that to me”) is parsed as if it were an SVO declarative sentence, which really should mean “at nine this gives me to something”. We fix this by causing the parser to recognize an imperative sentence as consisting of no terms or a series of sentence modifiers, followed by an unmarked predicate. For reasons to do with interaction with gasents, and following a style warning already given in L3, we parse a sentence consisting of no terms or a string of modifiers followed by a tensed predicate as a declarative sentence with an unexpressed subject **ba** or a subject yet to appear at the end as **ga X**. This modification makes very few sentences ungrammatical: it does rule out sentences of the form **terms gasent** in which one of the terms is an argument (such as **Ta ga donsü mi ga tu**); JCB says in NB3 that such sentences do not make sense (this form was provided to allow fronting of modifiers to gasents) and we explicitly force these terms to be modifiers.

We think that this is a nice solution to the felt lack of sentences with indefinite subjects. **Na crina** now means “It is raining” rather than “Be rained on now!”.

other issues: Of course there are other issues which will be commented on in the reference grammar and appendices.

5 Introduction to the Reference Grammar Sections

The purpose of this document is to give an independent description of the TLI Loglan language, in the provisional version embodied in my PEG parser, without PEG notations which are difficult for a nontechnical reader to follow. This does not mean that technicalities do not arise. One purpose of this is to give a clearly independent description of my intentions which can be used to double check the PEG parser. Another purpose is to give a venue for presenting material which is not in the purview of the parser, such as stating the semantics of grammatical words which are just items in lists for the parser.

An important point is that a lot of jargon (names of grammatical classes) is needed, paralleling structures in the PEG grammar and indeed in trial.85. I would like to create Loglan predicates for many or even all of these terms, first reducing their use as much as possible. There is a side project of creating an adequate native grammatical vocabulary. I have recently introduced English terminology for several important classes in the grammar section, replacing the use of trial.85 or PEG grammar class names, and cross-referenced these with the PEG appendix.

In intention, this document is to be a complete description of the language. It does not represent a power grab on my part: this is in the nature of an extensive proposal to *la Keugru* (and the membership). As always, I am well aware that my parser is not yet official. But I am not modest about the fact that I have definite ideas about how things will go, and I feel free to put them in here – but also obligated to point out proposals implicit in this text. I need to be sure to point out all places where 1989 Loglan has been modified, with or without an official academy decision supporting it.

My intentions are conservative. I do not feel committed to not making changes as I work on this but I am committed to the language described being intelligible to a speaker of 1989 Loglan (if such a being existed). My recent experience in parsing the Visit to Loglandia convinces me that I have been largely successful in this.

The descriptions given in the phonology section are often far simpler than the PEG code makes them look. Some of the specifications are quite awkward to achieve with a PEG (and would be even more awkward with a BNF grammar).

6 The Phonetics Proposal 2017

I wrote this document¹² recently as part of a larger document with a different purpose. It struck me that it could be used as a framework for presenting all of my current phonetic proposals to the Academy in the context of a complete description of the proposed phonetics of Loglan. An interesting point is that there are not actually very many new proposals! This goes along with my basically conservative approach to what we have inherited. I'm hoping that perhaps this can be Approved. Comment by all Loglanists is encouraged!

I think that all of this works. Some of it is quite baroque and could be improved in various ways (some of it perhaps *should* be improved), but my present mission is to get an official view of what the current state of affairs is. Essentially everything here is about phonetics and word forms: there are allusions to other perhaps controversial features of my provisional grammar (such as converting acronyms to names) but these have no bearing on the phonetics.

There is concrete evidence for my claim that this is conservative. I parsed *every word in the dictionary*. Only a handful needed to be changed, and most of those were wrong in 1989 Loglan terms.¹³

6.1 Sounds

6.1.1 Vowels

The Loglan vowels are the *regular vowels* **a,e,i,o,u** and the *irregular vowel* **y**. The pronunciations of the regular vowels are typical Continental European (not English!) pronunciations.

Vowels appearing singly (not adjacent to another vowel) are pronounced as follows:

- a** is pronounced as in *father*
- e** is pronounced as in *bet*
- i** is pronounced as in *machine*
- o** is pronounced as in *lost*

¹²This paragraph is talking about section 6 as a free-standing document, the Phonetics Proposal.

¹³Parsing every word in the dictionary has now been carried out with the PEG implementation of the new proposal. There were no new changes needed that had not already been seen to be needed in earlier parser versions.

u is pronounced as **oo** in *poor*

All of these are pure sounds. They can generally pronounced as in most languages spoken in continental Europe (English is severely aberrant in its spelling).

The sound of **y** is officially the schwa (the unstressed vowel in English *data* or the stressed vowel in English *hunt*), but we think there is something to be said for it being another sound easily distinguished from the regular vowels, such as the *oo* in English *look*.¹⁴ We have also considered *ö* and the Cyrillic letter that looks like *bl*¹⁵ as implementations of Loglan **y**. English and Russian speakers must be cautioned against pronouncing unstressed vowels such as the **a** in **matma**, *mother*, as a schwa: these vowels should have the identifiable value of **a**. In Loglan **matmymatma**, *maternal grandmother*, the two vowels occurring after *tm* should sound distinctly different.

The vowels have two series of names, the legacy VCV forms with the shape **V-fi** (**V-ma** for capitals) and the newer series with the shape **zi-V**. Another series of legacy vowel names with the shape **V-zi** is associated with the Greek alphabet. The legacy vowel names are phonetically weird among Loglan words, but they seem to be reasonably well accommodated. It should be noted that the principal function of all “letteral” words in Loglan is not as names of letters, but as *pronouns*, as will be discussed subsequently.

6.1.2 Consonants

The Loglan consonants are **b,d,f,g,h,j,k,l,m,n,p,r,s,t,v,z**. The pronunciations of these are standard European pronunciations, except that **c** is English *sh* and **j** is the corresponding voiced sound found in English *azure*. **g** is always “hard”. **h** has an alternative pronunciation as *ch* in Scottish English *loch* when final in a syllable¹⁶ [this is NEW: **h** does not occur in syllable final position in 1989 Loglan]. **n** is pronounced as *ng* in English *sing* when it appears before **k,g** or syllable final **h**. Note that Loglan **ng** is always as in English *finger*. NEW: **q,w,x** with odd pronunciations are eliminated: this was already mostly carried out in the 1990’s. It is worth noting specifically that **tc** is the sound in English **church** and **dj** is the sound in English **judge** (and that Loglan **j** is *not* the latter sound).

¹⁴A suggestion of John Cowan

¹⁵It is *not* easy to insert this in LaTeX!

¹⁶I accept an amendment from John Cowan allowing the alternative pronunciation of **h** in all contexts.

The names of the consonants are of the forms C-**ai** (capitals) and C-**ei** (lowercase), with a third series C-**eo** associated with the Greek alphabet. It should be noted that the principal function of all “letteral” words in Loglan is not as names of letters, but as *pronouns*, as will be discussed subsequently.

The Loglan alphabet is **abcdefghijklmnopstuvyz**, the Latin alphabet without **qwx**. The foreign letters **qwx** have proposed names **Kaiu**, **Keiu**, **Vaiu**, **Veiu**, **Haiu**, **Heiu** as part of a general proposal reserving C-**aiu**, C-**eiu** as additional series of letterals. Names for these letters are important, as all have common use as mathematical variables, the prototype for the Loglan use of letter names as pronouns.

6.1.3 Vowels pretending to be consonants

The vowels **i** and **u** are sometimes pronounced as the English consonants *y* and *w*.

6.1.4 Consonants pretending to be vowels

The continuants **m,n,l,r** can appear as syllabic consonants (functioning as the vowel in a syllable). In this role, these consonants are doubled, **mm**, **nn**, **ll**, **rr**. The requirement that syllabic consonants be doubled is NEW, but it is actually suggested by JCB in Loglan 1.

A doubled continuant may not appear adjacent to another occurrence of the same consonant in the same word (i.e., without an intervening pause in speech).

6.2 Diphthongs and vowel grouping

There are some mandatory and some optional pairs of vowels which form diphthongs¹⁷, which can serve as the vowel component of a single syllable.

6.2.1 Mandatory diphthongs

The mandatory diphthongs are **ai**, **ao**, **ei**, **oi**. The pronunciations are as one would expect from the values of the vowels, except that **ao** is as in English *cow*.

¹⁷I looked this word up in the dictionary, and it does seem to refer to vowel combinations in a single syllable, which is what is desired.

A new proposal is the addition of two irregular mandatory diphthongs **iy** and **uy**, allowed only in *cmapua*: the practical use of this for the moment is that it makes the new-style name **ziy** of the letter **y** legal. These are pronounced *yuh* and *wuh*, not what the English-reading eye expects.

6.2.2 Optional diphthongs

The optional diphthongs are the pairs of vowels beginning with **i** or **u**. The pronunciation of these diphthongs is as if the initial **i** were the English consonant *y* or the initial **u** were the English consonant *w*.

When a pair of adjacent vowels is not pronounced as a diphthong (i.e., when there is a syllable break between them), one may flow into the other without pause, or a glottal stop (not expressed in writing except possibly indirectly by a hyphen or stress mark) may be inserted (NEW: we do not allow the glottal stop to be an allophone of the pause, as earlier versions of Loglan did; in Lojban they require insertion of what we would write **h** in non-diphthong vowel pairs¹⁸). It is noted in our sources that such a pair is easier to pronounce if one of the syllables is stressed and the other is not, and when the two vowels are the same this is required.

6.2.3 Expanded notes on diphthong pronunciation

Two-letter diphthongs pronounced monosyllabically are as follows:

ai is English long **i** as in *pine*

ei is English long **a** as in *pane*

oi is as in English *boil*

ao is as **ow** in English *cow* (this is an irregularity, but we are stuck with it).

These four are the mandatory monosyllables: where these letters are grouped together, they must be pronounced monosyllabically.

¹⁸We note that the orthography **la'e** for a typical disyllabic CVV *cmapua* in Lojban, which they read **lahe**, is also valid orthography for this word in TLI Loglan, with the apostrophe signalling stress on the first syllable. We note without recommending it that a dialect in which non-diphthongs are broken with an invisible **h** is perfectly possible, as long as the additional rule is made that the **hV** *cmapua* must be pronounced with the alternative pronunciation of **h** (using the alternative pronunciation with **hVV** *cmapua* would be optional). Of course the invisible uses of **h** would have the usual soft pronunciation of **h**.

The pairs **ia**, **ie**, **ii**, **io**, **iu** are optional monosyllables. They may be pronounced as two syllables (smoothly moving from one vowel to the other without pause) or monosyllabically by pronouncing the initial **i** with the usual consonantal value of English **y**.

The pairs **ua**, **ue**, **ui**, **uo**, **uu** are optional monosyllables. They may be pronounced as two syllables (smoothly moving from one vowel to the other without pause) or monosyllabically by pronouncing the initial **u** with the usual consonantal value of English **w**.

These two classes are all the optional monosyllables. The disyllable pronunciation may be forced by an explicit syllable break (one of **-’***); some contexts without an explicit marker force the monosyllabic pronunciation, but I believe that no context forces the disyllable pronunciation in the absence of an explicit syllable break.¹⁹

The other disyllables are obligatory disyllables: they should be pronounced with a smooth movement from one vowel to the other without pause. Pronunciation is assisted if one is stressed and one is not. In the repeated vowel disyllables **aa**, **ee**, **oo**, one of the syllables must be stressed and the other must be unstressed. An explicit stress marker is permitted to indicate which one is to be stressed, but is certainly not required. The same stress rule applies to **ii** and **uu** where these are pronounced disyllabically. I am open to the idea of a pronunciation of disyllables using a glottal stop, as I do not regard a glottal stop as an adequate implementation of Loglan pauses.

We summarize some refinements. The **i**-final mandatory diphthongs are not read as such if immediately followed by another **i**: e.g., **aii** groups as **a-ii**. The triples **iuu** and **uii** are grouped **i-uu** and **u-ii**. All the preceding rules in this paragraph are designed to avoid formation of **i-i** or **u-u**, which would trigger the doubled vowel stress rule. The triples **iii** and **uuu** are recognized as triggering the double vowel rule (though the orthography does not tell us how). The odd triple **aoa** does group as **ao-o** and also triggers the double vowel rule. The triple **aoi** is grouped **ao-i** and is not regarded as containing a broken mandatory monosyllable.

¹⁹There *were* such contexts in the past: a CCVV or CCCVV predicate with the VV an optional disyllable had of course to be two syllables. But both these shapes for predicates are now banned.

6.2.4 Grouping long streams of vowels in predicates and names

A string of three or more vowels in a name or predicate word which is not marked with explicit syllable breaks (a hyphen or a stress marker is used for this purpose, as we will discuss below) is resolved into syllables (possibly including consonants adjacent to the string of vowels) following a priority order reading left to right

1. group the first two vowels and continue if they make up a mandatory diphthong (or **ii**, **uu**). Do not apply this rule to a mandatory diphthong ending in **i** immediately followed by another **i**.
2. pronounce the first vowel as a single syllable and continue if the second two make up a mandatory diphthong (or **ii**, **uu**) and the first two do not (or in the cases **aii**, **eii**, **oii**).
3. optionally, take the first vowel as a syllable and continue or group the first two vowels and continue if they make up an optional diphthong and the previous conditions do not hold; the parser will always take the second alternative.
4. pronounce the first vowel as a single syllable and continue, if none of the previous conditions hold.
5. by “continue”, we mean “apply the same set of rules to the remainder of the stream of vowels”.

This is NEW, not the same as the rule in 1989 Loglan (given there only for names) but it appears to have similar effects in practice.²⁰

6.2.5 Grouping long streams of vowels in structure words

A stream of vowels of even length parsed as a *cmapua* (structure word) will first be divided into VV units, each of which will then be read as one or two syllables depending on whether it is or can be a diphthong. A stream of vowels of odd length so parsed will not occur, as it would be read as a V word followed by a stream of VV units, and a pause is required before the

²⁰The rule in 1989 Loglan calls for right-grouping, which is psychologically incredible. I think the reasons why JCB wanted to right-group are captured by the second clause of my algorithm.

first of the VV units (see below) (this would equally be the case if it were read as a stream of VV units followed by a V word: it would be necessary to pause before the V word).

6.2.6 Vowel pairs with optional grouping revisited

An optional diphthong not appearing in a stream of three or more vowels may be pronounced either as a single syllable or two syllables: sometimes other factors will force the monosyllabic pronunciation. We believe that it is not possible to force the disyllabic pronunciation of an optional diphthong without explicit indication of a syllable break.

6.2.7 Doubled vowels and stress

Where doubled vowels are not separated by a pause and not pronounced as a diphthong, one of them must be stressed: this always applies to **aa**, **ee**, **oo**, and applies to **ii**, **uu** unless they are pronounced *yee*, *woo*.

6.3 The Loglan syllable

6.3.1 Discussion of stress and notation for syllable breaks and degrees of stress

We now discuss the Loglan syllable. Each Loglan syllable is either unstressed, stressed, or emphatically stressed. Syllables may be separated by a hyphen. A marker ' of stress or * of emphatic stress may terminate a syllable²¹: a stress marker may not be followed by a hyphen, as the stress marker itself serves the purpose of separating the syllable from a following syllable (though a stress marker can also be final in a word; actually, so can a hyphen, in a phonetic transcript where no explicit pause occurs between a word and the following word). Syllable breaks and stress markers do not have to be written explicitly, though it can be useful to do this. The precise definition of the syllable that we give does not appear in the sources, but every component of it is found there, and all words in the dictionary are successfully resolved using this definition; we do not regard this definition as new except in detail. The use of the hyphen and the explicit stress markers is NEW. The hyphens and stress

²¹It is important to note that it is the end of the syllable, not the vocalic unit of the syllable, which is marked with the stress.

markers make possible a style of writing Loglan with no whitespace except where explicit comma-marked pauses occur, which we refer to as “phonetic transcript”. The availability of phonetic transcript means that we have our own “native” phonetic notation and do not need to appeal to IPA notation or to such odd expedients as JCB uses.

Note that a syllable without a stress marker is not necessarily unstressed: we simply have not committed ourselves.²²

We briefly remark on the capitalization conventions of Loglan. A stream of letters and junctures is governed by the capitalization convention. The basic rule is that a lower-case letter will not be followed directly by an upper-case letter (except that **z** may be followed by a capitalized vowel (useful in acronyms, as **DaiNaizA**), and that embedded copies of letter names may freely be capitalized (as in **leSai** and as also seen in **DaiNaizA**). Note that capitalization may resume at a juncture, as in **Beibi-Djein**. Note also that this rule allows all-caps.

We briefly note that stress in Loglan names is completely free, as is stress in Loglan structure words (cmapua): in words of these classes one may have no stressed syllables, one stressed syllable in any desired position, or no stressed syllables. In Loglan predicate words, there is a single primary stress on the penultimate syllable (usually; sometimes a special additional unstressed syllable intervenes, governed by the rule that if the actual penultimate syllable has a syllabic consonant or **y** as its vocalic unit, it cannot be stressed, and in this case the preceding syllable is stressed); there may be additional stresses on the penultimate syllables in *borrowing djifoa* components of predicates, as described below. There is a special rule that a pause must intervene between a finally stressed structure word and a following predicate word. Explicit stress markers may be used to signal an unexpected stress in a foreign name, or may be used by the Loglan writer as a mechanism for rhetorical emphasis in any word or to signal actual stress in reported speech.

²²Many syllables in actual Loglan text can be recognized as definitely unstressed from context, e.g., most syllables in predicate words, but there is no marker for this so far.

6.3.2 Every syllable has a vocalic unit

Each Loglan syllable contains a vocalic unit, which is either a single vowel²³, a diphthong pronounced as such, or a doubled continuant.

6.3.3 Like Gaul, every syllable has (up to) three parts

Each Loglan syllable consists of up to three parts.

1. The first part (which is optional) is a consonant cluster called the initial consonant group. There is a list of pairs of consonants called permissible initial pairs (or just initial pairs). An initial consonant group will be either a single consonant, an initial pair, or a group of three consonants in which each adjacent pair of consonants is a permissible initial pair.²⁴
2. The second part is the mandatory vocalic unit.
3. The third part, which is optional, consists of one or two final consonants. There is a list of impermissible medial pairs and a list of impermissible medial triples.²⁵ A final consonant may not be followed by a vowel and may not stand at the beginning of an impermissible medial pair or triple (regardless of whether the other consonants are in the same syllable, and ignoring stress marks and hyphens). A syllabic pair is an impermissible medial pair: a final consonant may not participate in such a pair, even with a syllable break intervening.

A pair of final consonants may not consist of a non-continuant followed by a continuant [the last sentence is NEW, but seems self-evident: such

²³the irregular vowel **y** can be a vocalic unit, though its distribution is limited; it can occur freely in names and as a “phonetic hyphen” in complex predicates, but not at all in *cmapua* or borrowed predicates: some changes allowing its use in *cmapua* in limited contexts are implemented in the new PEG.

²⁴The initial pairs are **bl br ck cl cm cn cp cr ct dj dr dz fl fr gl gr jm kl kr mr pl pr sk sl sm sn sp sr st sv tc tr ts vl vr zb zl zv**

²⁵The impermissible medial pairs consist of all doubled consonants, any pair beginning with **h**, any pair both of which are taken from **cjsz, fv, kg, pb, td**, any of **fkpt** followed by either of **jz, bj**, and **sb**.

There is a list of impermissible medial triples as well, consisting of **cdz, cvl, ndj, ndz, dcm, dct, dts, pdz, gts, gzb, svl, jdj, jtc, jts, jvr, tvl, kdz, vts**, and **mzb**. All of these consist of a consonant followed by an initial pair, but they are not permitted to occur with the juncture between syllables in either of the two positions.

a pair would basically have to be pronounced as a separate syllable, and no violations occur in the dictionary].

In placing syllable breaks in the absence of an explicit hyphen or stress mark, the rule is followed that a final consonant shall not stand at the beginning of a legal syllable, except in the case of a first final consonant at the end of a syllable whose initial consonant group consists of a single consonant and whose vocalic group is a single regular vowel (the general idea is that the syllable break is usually placed as early as possible, except that a CVC syllable is read in preference to a CV syllable where possible).²⁶

6.3.4 Words must resolve into syllables

Every Loglan word must resolve into syllables. Some classes of words must also resolve into other small units which are not exactly syllables, though syllables do not as a rule cross their boundaries.

Syllable boundaries may be phonemic (in the sense that changing their placement can change a word into a different word) only in the case of syllable breaks between vowels, and only in names. Potential words which can be written only with the use of explicit syllable breaks may be legal names (as the classic example **La Lo-is**) but not legal words of any other class.

6.3.5 Notation for consonant and vowel patterns

C represents a consonant; cc (lowercase pair) represents an initial pair, where CC represents any pair (initial pairs included). V represents a regular vowel (not **y**); vv (lower case) represents a diphthong pronounced as such, where VV represents any pair of regular vowels, whether pronounced as a diphthong

²⁶The previous parser behaved differently in this respect in names, allowing no final consonant to stand at the beginning of a legal syllable in the absence of an explicit syllable break. This isn't a difficulty; there are reasons for the more nuanced rule to be applied in predicates. In complex predicates, any explicit syllable break placement should agree with djifoa structure. In borrowed predicates, the placement of explicit syllable was governed by some rather mysterious rules which had the effect of preventing one from obtaining a legal borrowing predicate by moving syllable breaks in illegal complex predicates in a way which violates djifoa structure: below we give a different description of which borrowing predicates should be excluded for this reason, and the new parser should allow freer placement of explicit syllable breaks.

or not. In such pattern notations, a hyphen stands for an explicitly marked syllable break, whether marked with a hyphen or a stress marker.

6.4 Word Forms

6.4.1 Word forms enumerated

There are four phonetic classes of Loglan words: these are (1) structure words (**cmapura**), (2) name words, (3) borrowed predicates and (4) complex predicates.

6.4.2 Pauses and word boundaries

Words (in the phonetic sense) end at whitespace, at a comma or mark of terminal punctuation (periods and some other punctuation marks), or sometimes without any explicit indication at all (where phonetics are sufficient to recognize where one word ends and another begins).²⁷

A comma in Loglan marks an explicit pause (and is followed by whitespace; the close comma used to indicate unusual syllable breaks between vowels in earlier versions of Loglan is replaced by the hyphen which we use to represent syllable breaks in general).

Whitespace is sometimes an explicit pause and sometimes a word boundary which is not marked by any actual phonetic feature. Where whitespace does not appear, one should not pause. Where a pause is allowed at whitespace, a comma should always be permitted (The old parser LIP does not always support this, but we regard this as debugging, not a novelty). There are situations where whitespace is allowed due to a word break but an actual comma pause would change the parse (and so in speech such a whitespace is not expressible as a pause).²⁸

²⁷The marks of terminal punctuation are **::?!.** The silence or change of voice marker **#** used by JCB is supported. This may not appear in quoted or parenthesized Loglan text; it is not really fully privileged punctuation. It does allow multiple utterances in different voices (including the same voice stopping and starting again) on the same line of parsed text. Words can end at double quotes or close parentheses, which appear in certain special contexts. The dash **–** and ellipses **...** are supported as “free modifiers” and may be used fairly freely.

²⁸There are such instances of whitespace which are permitted to be written but cannot represent a pause, in connection with the handling of the legacy forms of the APA connectives, another pain in the language, and quite another story.

Vowel initial words are always preceded by a pause if they are not at the start of a text or utterance. Consonant final words are always followed by a pause if they are not at the end of a text or utterance. Thus, whitespace preceded by a consonant or whitespace followed by a vowel must represent an actual pause. So we also regard whitespace preceded by consonants or followed by vowels as an explicit pause. It appears to be NEW that we must pause before the first in a stream of VV words, but it is also clearly necessary, as experiments with phonetic transcripts have revealed.²⁹

We note the subtle point that the end of a predicate word may have to be indicated by whitespace if the stressed syllable is not explicitly marked. So in this case whitespace may have no local phonetic meaning but will have the definite phonetic effect of signalling the presence of an earlier stressed syllable. In phonetic transcripts, where whitespace not representing pauses is suppressed, the stresses in predicates must usually be marked explicitly.

It is possible to resolve a stream of Loglan phonemes into words unambiguously, with the qualification that the resolution of streams of Loglan grammatical particles is actually done by the grammar proper. We indicate how to do this.

6.4.3 Structure words (**cmapua**)

The structure words or “little words” (Loglan **cmapua**) are the grammatical vocabulary of Loglan, for the most part.

A structure word (in the phonetic sense: some phonetic structure words are actually semantically names or predicates) is a word which resolves into a stream of V, VV, CV, CVV, and Cvv-V units (where vv denotes a diphthong). These words are called **cmapua** in Loglan. The CVV units do not have to be syllables (there is no requirement that the VV be a diphthong, or that it be pronounced as such if it is an optional diphthong). A V unit can only appear initially; if any unit in a **cmapua** is of the shape VV, all units are of the shape VV. We recall the rule that one must pause before a word which begins with a vowel, so one must pause before an initial V- unit; it is not necessary to pause between VV units in a phonetic word made up of such units, but it is necessary to pause before a lone VV unit or the first in

²⁹The new PEG implementation supports the traditional requirements that pauses at the end of a serial name and pauses before a logical connective must be actual comma pauses. Some logical connectives are consonant-initial: there is a purely phonetic description of the front of a logical connective in the PEG.

a stream of such units (the necessity of pausing before the first in a stream of VV units seems not to have been recognized in NB3). Except in the case of the end of a string of VV units, it is impossible to recognize the ends of individual words in a stream of cmapua phonetic units on phonetic grounds alone: the grammar proper allows us to resolve streams of unit cmapua into words, but for phonetic purposes we may regard streams of VV units and streams of non-VV units as “words”.^{30 31}

To support names for the letter **y** (and perhaps add a little cmapua space), we propose to allow the irregular monosyllabic diphthongs **iy** and **uy** in CVV (but not Cvv-V) cmapua units. We further propose to allow **y** to be a V djifoa unit (in compounds). This makes the name **yfi** of **y** legal.

It is part of the definition of a cmapua unit that it cannot be an initial segment of a predicate (either complex or borrowed): a more precise statement of this is that a cmapua unit may not be followed without an intervening explicit pause by **y** or by **CyC** (nor may a CV cmapua unit be followed by **CCy**), and may only be followed without an intervening explicit pause by **CC** if the **CC** itself is initial in a predicate word (this last condition requiring lookahead!) This indicates the (sometimes rather subtle) way that the end of a phonetic structure word at the beginning of a predicate word is recognized; how to recognize the end of a structure word at the beginning of a name word is covered in the next section (and may require considerable lookahead).

There is a precise set of conditions under which an apparent cmapua unit cannot be part of a cmapua word because it is the start of a predicate word (if it is anything legal). It is further the case that the start of a borrowing or complex not beginning with an initial pair or triple of consonants must pass one of these tests:

1. If the unit is followed without pause by two or more consonants, where the consonant group is not an initial group of consonants or is a consonant or initial group of consonants followed by a syllabic pair.
2. If the unit stands at the beginning of a copy of a CVVy, CVCy, or

³⁰The parser will end a stream of cmapua units before a cmapua unit which is either a name marker, a marker of an alien text construction, or an initial marker of a quotation construction; a stream of cmapua units can begin with a name marker word if it does not start a name word.

³¹There is a further subtlety that certain streams of unit cmapua which are broken by explicit pauses may be recognized semantically as words of class PA or NI: this is a semantic rather than a phonetic proposal, not discussed here

CVCCy djifoa (see below for discussion of djifoa).

3. If the unit is finally stressed and immediately followed without pause by more than one consonant, except in the case where the unit is a V or Cvv-V cmapua unit or the head of a stream of VV cmapua units followed by ccV: this last cluster of cases is simply illegal, as no predicate can start in such a way for technical reasons.
4. If the unit is followed by an initial group of consonants which is then followed by V or VV followed by whitespace or non-syllable-break punctuation or by a stressed V followed by a single V not in a diphthong.

There is an exception: a V or Cvv-V cmapua unit may be followed by ccV (the cc being an initial pair) if not finally stressed. We note that the parser applies these exact tests rather than looking ahead to see if a predicate word actually starts at the beginning of the apparent unit under consideration.

Where the final unit of a phonetic cmapua immediately precedes a predicate word and is stressed, it must be followed by a pause: if the stress is marked in writing and the following predicate is consonant initial, the pause must be explicitly written as a comma. We are pleased with the fact that this rule, going back to the beginnings of Loglan, can be expressed in our orthography and is enforced by our parser; it is quite invisible to LIP.³²

6.4.4 Name words and the false name marker problem

Semantically, name words are *proper names*, as their name suggests. Name words are usually capitalized, but this is not obligatory. Names are required to end with consonants. A Loglan name borrowed from a foreign source which happens to end in a vowel is conventionally terminated with an **s**, as in **Selis**, *Sally*. Names borrowed from foreign sources will be spelled in such a way that the Loglan speaker's pronunciation of them will approximate

³²We note a subtle point about the articulation of acronyms in Loglan: these are semantically names but phonetically cmapua. The legacy vowel names are of the weird shape VCV; they can occur without initial pause following a CVV unit because the (CVV)(VCV) shape (when articulated as letters) can be rearticulated as (Cvv-V)(CV) for purposes of articulation as cmapua units. We have abandoned an irregularity found in the previous provisional parser: the CVV letters of the common sorts have the VV actually a diphthong, but there are **Ceo** letters, and these formerly worked in acronyms preceding a legacy vowel, but no longer do.

the original (**Ainctain**, not **Einstein**). There is a direct foreign name construction which allows preservation of foreign spelling, to be discussed later. Names derived from Loglan words which are not names (and so vowel-final) are conventionally constructed by adding an **n** to the Loglan word.

A name word is a stream of Loglan syllables ending in a consonant followed by whitespace, a comma marked pause, or terminal punctuation. Name words are the only Loglan words which end in consonants. The parser currently requires an explicit comma pause in place of whitespace after a name word in most contexts, though the fact that a consonant followed by whitespace is recognized as an explicit pause suggests that this can be relaxed. The boundary of a name word on the right is readily recognized (the consonant followed by a explicit pause or terminal punctuation). Left boundaries of name words always fall just after either a *cmapua* belonging to a class of “name markers”³³, or an explicit pause. A candidate left boundary for a name word falls just after an explicit pause or just after a phonetic copy of a name marker such that the text between its end and the recognizable right boundary of the name word resolves into syllables (and of course contains no pauses or whitespace). A candidate left boundary is said to be marked if it is just after a name marker or just after an explicit pause immediately preceded by a name marker. The actual left boundary of a name word is the leftmost marked candidate left boundary, if there is one, and otherwise the rightmost candidate left boundary.

We refer to name markers which are candidate left boundaries of name words but are not the actual left boundaries as “false name markers”. Earlier versions of Loglan forbade these, with the odd effect that (for example) **la** could not occur in a name. Lojban still forbids this (and has very few name marker words). Our current grammar of Loglan strongly restricts the situations in which any name word can occur with an unmarked left boundary (for example, the current Academy has banned unmarked vocatives), and in any such context there is provision for the boundary to be marked with a suitable name marker if the name word happens to contain a false name marker. This is a generalization of rules for handling serial names adopted

³³The name markers are **la**, **hoi**, **hue**, **ci**, **liu**, which must be name markers for one reason or another, **gao** if a proposal to allow this *cmapua* to form letter names from name words is accepted, and in a NEW proposal, the words of social lubrication **loi**, **loa**, **sia**, **sie**, **siu**. If adding the words of social lubrication is thought excessive, we can continue to say **Loi hoi Djan** instead of **Loi Djan**: in fact, the new PEG does not have these words as name markers.

in the 1990's.

Where a false name marker occurs and it was the actual intention of the speaker to make it the left boundary of the name word, the intention can be realized by pausing explicitly after the name marker, making it a true name marker. The intention would more usually be realized by pausing somewhere earlier, due to the actual class of errors which leads to this situation, discussed in the next paragraph.

The parser will raise an error if it finds a name marker word followed possibly by an explicit pause, followed by text including whitespace but no explicit pause, followed by a name word (recognizable by its right boundary at a consonant). The problem with this is that because of the intervening whitespace, it cannot be the intention of the writer that the name marker sets the left boundary of the name word, but if the utterance were read without pause, the name marker would indeed set the left boundary of the name word.³⁴ There are uses of name markers in which they are not followed immediately by name words: it is the obligation of the speaker to explicitly pause at some point after such an occurrence of a name marker and before the next occurrence of an actual name word, and Loglan orthography requires this to be indicated explicitly. Note that whitespace before a vowel or after a consonant does suffice, but where this doesn't happen, an explicit comma pause (of the form V, C) may be required. Actually complying with this rule is best implemented by style directives such as "always pause after a predicate name", rather than by attention to this esoteric rule as such.³⁵³⁶

³⁴A reader requested an example. **La Farfu ga cluva la Djan** would trigger this problem, and needs to be corrected to **La Farfu, ga cluva la Djan**. Without the explicit pause, it could be read phonetically as **lafar'fuclu'valadjan**, which parses as **La Farfugacluvaladjan**, a single name.

³⁵A directive which would always work is, "always pause after the first word after a name marker, whether it is a name word or not (and whether you paused after the name marker or not)". When what follows the name marker is not a name word, one need not pause after exactly one word, but one should pause at some natural point before the next name word. An example: in **hoi le farfu je la Rabrrt**, which is bad, it is legal but odd to amend it to **hoi le, farfu je la Rabrrt**, most natural to amend it to **hoi le farfu, je la Rabrrt**, and a sign of last minute panic to amend it to **hoi le farfu je la, Rabrrt**.

³⁶It is further important to note that my previous parser did something different in the situation where a name marker is immediately followed by a pause, which seemed incredibly clever in the context of the kind of parsing I am doing but which I have reverted to the exact behavior described here. A first approximation to what my parser did is that it chose the leftmost candidate left boundary in which the name marker was not followed by a pause, and otherwise the rightmost candidate left boundary: but further, this was

The rule that false name markers cannot occur in names has already been abandoned (in the 1990s) by TLI Loglan; it still obtains in Lojban, which has very few name markers. Precise definition of what you do with false name markers was hard to think about before phonetic parsing was available. The requirement that names resolve into syllables is NEW (I seem to recall that Lojban does something like this?), and interacts with the definition of a false name marker as indicated above. The addition of the words of social lubrication to the list of name markers is NEW.³⁷ The automatic detection of dangerous situations where a non-explicit pause should be made explicit is NEW, and we can report from extensive experience in parsing Alex Leith's Visit to Loglandia that there are straightforward ways to correct problems it detects (and that it really does detect things which are potential problems). We can also report, based on extensive parsing of phonetic transcripts of Loglan utterances, that the rules stated above appear to work: the situations in which a name is unintentionally started too early can be controlled.

It might be thought that imposing additional restrictions on the formation of Loglan names would damage the corpus. In fact, there are only two difficulties which arise. Continuants must be doubled (as in **Rrl**, "Earl") and this happens not unseldom; but JCB *did* actually suggest that this might be a good idea in Loglan 1. The other problem is that our definition of the syllable does not allow final consonant clusters of more than two consonants. Usually one of the consonants in such a cluster is a continuant, so this can be fixed, as in **Hollmz**, **Marrks**.

6.4.5 Borrowed predicates

Semantically, borrowed predicates are common nouns, verbs, adjectives or adverbs (not names) borrowed from foreign sources. Their phonetics are dictated by the idea that they should not take any of the phonetic shapes of "native" Loglan predicates (complexes) other than the primitive five-letter shapes CCVCV or CVCCV (in which case they are phonetically treated as

handled dynamically by aggressively reading name words after name markers which are not followed by pauses, but attempting to read something else after a name marker which was followed by a pause, and then reading a name word if this attempt at parsing failed: the effect of this was that the resolution into words depended strongly on non-phonetic features of the grammar, which was a bad thing: the new PEG fixes this, behaving exactly as described here.

³⁷The new PEG implementation does not add the words of social lubrication as name markers.

(very simple) complexes). There will be a discussion later of how to modify the foreign originals to get legal Loglan predicates which are not phonetically of the shape of a complex.

Phonetically, all predicates (borrowings and complexes) have certain common traits. They are *penultimately stressed* (with the exception that a special unstressed syllable with a continuant or **y** as their vocalic unit may intervene between the stressed syllable and the final syllable). They are *vowel-final*. They contain a *consonant pair*, either CC or CyC (the latter in some complexes).

A borrowed predicate is a stream of Loglan syllables which contains just one stressed syllable, whose vocalic unit is not a continuant pair, the stressed syllable being followed by at least one and no more than two syllables, the first of these, if there are two, having a continuant vocalic unit, and the final one not having a continuant vocalic unit and not ending in a consonant: in other words, it is penultimately stressed, ignoring one possible intervening unstressed continuant syllable, and ends in a vowel. A borrowed predicate contains at least one pair of distinct adjacent consonants; it is permissible for one of these to be in a continuant pair. A borrowed predicate may not contain two successive syllables with continuant pairs nor may it start with such a syllable (or, as noted, end with such a syllable), nor may a continuant pair immediately follow a vowel in a borrowed predicate. A borrowed predicate may not contain a doubled vowel unless the doubled vowel is pronounced as a diphthong. A borrowed predicate may not contain **y**. The part of the borrowed predicate before the first pair of distinct adjacent consonants must have the property that omitting it (and dropping any explicit syllable break between the pair of consonants) will not leave a legal borrowed predicate. A borrowing cannot begin VccV with the cc permissible initial, and there can be no ccVV or cccVV borrowed predicates³⁸. A Cvv-V cmapua unit or a stream of VV cmapua units followed by ccV cannot be the shape of a borrowing: this restriction is caused by a technical problem with borrowing djifoa (see below). The last restriction is NEW, motivated by the same problem averted by forbidding VccV-initial borrowings; only one word in the dictionary was affected.³⁹

³⁸The ccVV predicates are forbidden so that CVCccV complex predicates do not have to be **y**-hyphenated; the cccVV predicates would, if allowed, greatly complicate our parsing algorithm for a weird technical reason.

³⁹What needs to be averted is the possibility of reading a borrowing djifoa ending with ccVy as a stream of phonetic cmapua units followed by a predicate beginning with the

The beginning of a borrowed predicate must either be at an initial pair or triple of consonants or a CV^n passing one of the tests listed above in the *cmapua* section (and thereby failing to be a *cmapua* unit). The end of a borrowed predicate is recognized either by seeing an explicitly stressed syllable and counting the one or two allowed following unstressed syllables, or by whitespace, a comma, end of text, or terminal punctuation: in the latter case, the fact that one is in a borrowed predicate is recognized by the occurrence of two adjacent distinct consonants.

A borrowing must not resolve into *djifoa* (see the next section), even ones with badly placed internal syllable boundaries or lacking required phonetic hyphens (noting that the five-letter *djifoa* and the borrowing *djifoa*, when not final, *include* their **y** hyphens and so will not be involved in such resolutions, since a borrowing candidate cannot include **y**, and a final borrowing *djifoa* is always preceded by **y**). This both ensures that actual complex predicates are read as complex predicates rather than borrowings, and ensures that certain illegal complexes are not read as legal borrowed predicates.

There is almost nothing actually new in the description of borrowed predicates (only the one point labelled with NEW above): some features are points worked out in the 1990's (all details of borrowing *djifoa* are late and not in 1989 Loglan 1: these details motivated the elimination of **y** from borrowings and the more baroque excluded forms above). Forbidding doubled vowels in borrowings is the most recent change, made by the current Academy in the last few years. Everything else is explicit in the sources somewhere (there may be some guesswork about the exact rules for use of continuants for gluing, but they fit actual practice). The precise definition of the syllable was made in order to make it possible to implement the description of borrowings in NB3 and L1.

cc**Vy** *djifoa* instead of the intended borrowing *djifoa*. Both the original $VccV$ rule and my new rule forbid some predicate shapes which are actually not problematic (but also not needed). I have modified this proposal recently to allow more of the improbable space of possible borrowings with many vowels before the CC pair to be preserved; these very likely should be banned, but this rule is not the context in which the decision should be made. I note the possible reasonable restriction that no predicate should begin CV^4 or V^4 .

6.4.6 Complex predicates

Semantically, the complex predicates are the native common nouns, verbs, adjectives and adverbs of Loglan (which really does not have these parts of speech). The root vocabulary of Loglan, apart from the *cmapua* which make up its grammatical vocabulary, is made up of primitive predicates of the five-letter forms CCVCV and CVCCV and complex predicates built from these five letter roots and shorter combining forms (the deprecated original terminology for these is “affixes”; we now call them *djifoa*, a Loglan term) of shapes listed below: each five letter root has a form in which its final vowel is replaced with **y** which may appear in non-initial position in a complex, and may have one or two three-letter forms of the shapes CCV, CVV, CVC associated with it. Here we concern ourselves with the phonetic rules by which such complex predicates are to be constructed. Recall that Loglan predicates have the common characteristics of *penultimate stress*, being *vowel-final*, and having a *consonant pair*.

A complex predicate is a stream of units distinct from syllables, called *djifoa*, with the additional property that any syllable breaks respect the *djifoa* boundaries (by which we mean that no syllable contains parts of two neighboring *djifoa*; testing for resolution into *djifoa* requires that *djifoa* with badly placed internal syllable boundaries be recognized; predicates which resolve into *djifoa* with badly placed boundaries are to be rejected as borrowings as well).

The *djifoa* take the forms

1. CVV (legal syllable forms Cvv or CVV or CV-V)
2. CVC
3. ccV (the cc must be permissible initial).
4. ccVCV (when this is not final, the final V is replaced with **y**). The CC must be permissible initial. The only legal syllable break is ccV-CV.
5. CVCCV (when this is not final, the final V is replaced with **y**). The CC must not be impermissible medial. CV-ccV (if the CC is permissible initial) and CVC-CV are legal syllable breaks. Either break is permitted if the CC is permissible initial.

6. a borrowing predicate (modified to have final rather than penultimate stress if not in final position) with appended **y** (in an unstressed syllable by itself) if not in final position.
7. When attempting to resolve a predicate into djifoa to establish that it cannot be a borrowed predicate, the forms to consider are CVV, CVV**r**, CVV**n** (only when followed by **r**), CVC, ccV, and the five-letter forms in final position only, plus the illegal CV-C, c-cV, c-cVCV (the last in final position, the hyphen indicating an explicit syllable break or stress marker; c-c denotes an initial pair broken by an explicit syllable break). A mechanical resolution into these forms, without any side conditions, shows that a string cannot be a borrowed predicate. We do not need to list forms with syllable breaks C-V because these are explicitly forbidden by our phonetics already.

Please note that we are not saying anything about where djifoa come from, not because this is unimportant, but because it does not bear on the strictly phonetic business at hand.

Note that a complex predicate may not contain any continuant pair, except in the context of a borrowing djifoa.

Any complex includes at least two djifoa, unless it consists of a single five-letter djifoa (the latter could also be viewed as a separate species of primitive predicate).⁴⁰

Djifoa appearing in non-final position may have phonetic hyphens appended, which may take the shapes **n**, **r**, or **y**. Only one phonetic hyphen can be appended to any djifoa. The consonant hyphens can only be appended to CVV cmapua. A phonetic hyphen **n** appears only when followed by **r** in the next djifoa. A phonetic hyphen is never final in a complex predicate nor will it follow a five-letter djifoa. A phonetic hyphen **y** is always unstressed, and appears by itself in a syllable or in a **Cy** syllable in the context CV-**Cy** of a CVC djifoa (CVC-**y** also being allowed; the **y** in a five-letter djifoa in non-final position may participate in a **Cy** or cc**y** syllable, which is also always unstressed). A borrowing djifoa is always preceded by **y** if not initial (the **y** will be a hyphen or a constituent of a five-letter or borrowing djifoa) and includes an appended **y** if not final.

⁴⁰The PEG grammar also thinks that a borrowing is a one-complex djifoa, so it classes all predicates as complexes!

An initial CVV followed by a djifoa beginning CV must be hyphenated with a consonant. An initial CVC followed by C in a way which would make a permissible initial pair must be hyphenated with **y** if the entire word is not of length 6. These are rules to allow recognition of the start of a predicate word, preventing cmapua units from falling off the front.⁴¹

There must be a CC pair (or a CyC pair) in a complex predicate. This is enforced by two provisions: the start of a complex must be at a CC pair or at a CVⁿ passing one of the tests listed in the cmapua section (and thereby failing to be a cmapua unit); further, any CVV**y** djifoa must be followed by a complete complex or borrowing (a complex made entirely of CVV's using **y**-hyphens would not contain a CC or CyC pair, and is excluded by this rule). This last condition is NEW: LIP accepts **ceaydea** as a predicate, for example).

A complex predicate must end with a regular vowel (so the last djifoa will not be CVC or phonetically hyphenated). Any adjacent pair or triple of consonants in a complex predicate may not be impermissible medial.

A complex predicate may contain stress only in the penultimate position (where it must be stressed; in determining the penultimate stress, a syllable with **y** intervening between the stressed syllable and the final syllable may be ignored) or in the final position of a non-final borrowing djifoa (just before the **y**; the final, not the penultimate syllable, of the borrowing; here stress is optional, unless such a syllable is also penultimate in the predicate). It is also permitted to contain a pause (in explicit comma form) after the **y** at the end of a stressed non-final borrowing djifoa (violating our expectations about word boundaries, but it is there in the sources, and it may be practical, as borrowing djifoa are large). We think that the stress shift in non-final borrowing djifoa may serve as a useful marker that something odd is going on when this happens.⁴²

⁴¹No additional enforcement of this is needed in the parser: the algorithm described in an earlier footnote for determining whether an apparent cmapua unit before an initial pair of consonants is actually a cmapua unit enforces this already by causing the CV- in a CVC djifoa to fall off if it is initial and makes an initial pair with a following consonant.

⁴²We do not allow secondary stress on the penultimate syllable of a borrowing djifoa (the final syllable of the parent borrowing) or a pause after it if what follows the borrowing djifoa in the complex is not itself a legal complex. To quote an example where we actually ran afoul of this, it is not legal to articulate **aurmoykoo** as either **aurmo'yko'o** or **aurmo'y, ko'o**, because **ko'o** itself is not a legal complex. In **igllu'ymao**, the final syllable of the borrowing djifoa is stressed, but it must be because it is penultimate in the predicate (ignoring the **y** syllable on which stress cannot fall); this is primary, not

Note that stress strongly constrains where a CV-V djifoa may appear if the VV is a doubled vowel.

Nothing in the description of complex predicates is new, though all language about borrowing djifoa comes from decisions taken in the 1990s after the 1989 edition of Loglan 1. Considerations about explicit stress markers and syllable breaks are new but consistent with the logic of complex predicates as already defined: we do not want an illegal complex to become a legal borrowing by moving a syllable break so that it doesn't conform with a djifoa boundary. We expect that directly implementing the resolution procedure described above will allow us to relax some rather odd rules about placement of explicit syllable breaks which are currently enforced in borrowings under our parser.

6.5 Appendix: Alien Text and Quotations

The latest version of the parser does a complete pass with a checker for the phonetics, then does a pass with the lexicography and grammar checker. Thus, the phonetics checker needs to be able to recognize constructions with alien text (strong quotation with **lie**, foreign names with **lao**, foreign predicates with **sao**, onomatopoeia with **sue**, and vocatives and inverse vocatives with **hue**, **hue**).

The parser recognizes the first four markers as inevitably followed by alien text. Alien text comes in two forms: it can begin with a double quote, end with a double quote, and contain any character but double quote, or it can consist of blocks of text containing any characters but whitespace, commas, and terminal punctuation and separated by the special word **y**. Blocks of alien text are set off with pauses initially and finally, if they are to be pronounced (and Loglan provides no advice on how to pronounce them). These pauses can be expressed by commas but do not have to be. When **hoi** and **hue** are followed directly by alien text (when addressing someone whose name is illegal in Loglan) the alien text must be enclosed in quotes: this is practical, as typos or grammatical errors might go unnoticed due to the parser accepting bad Loglan text as alien text.

The format for alien text is essentially that allowed to follow **lao** in 1990's Loglan (originally the constructor for Linnaean names: Steve Rice observed correctly that it was better thought of and used as a general foreign name

secondary, stress. **igllu'y**, **mao** is not permitted.

constructor), though the `, y`, to be inserted at whitespace was left unexpressed in writing in the original proposal. We require it to be expressed in the absence of quotes, and note that of course whitespace is to be read as `, y`, in quoted alien text. If we want to refer to Einstein with his native spelling, we can style him **lao Einstein** and pronounce this as in German or English, where we must write **la Ainctain** if we want to import his name properly into Loglan. If we style him more elaborately as **lao Albert Einstein** we must remember (though this is optional in writing) that this is read **lao Albert y Einstein**. This proposal is not new for Linnaeans, in effect, but it is NEW as an implementation of strong quotation (the 1989 Loglan version is quite unparseable in BNF or PEG formats) and allows NEW constructions in the cases of **sue** and **sao** (**sao "ice cream"**, to be read **sao ice y cream** (also transcribable in the latter form).

The phonetic rules also support use of double quotes in **li...lu** quotations by recognizing **li** followed by a double quote followed by a phonetically well formed utterance followed by a double quote followed by **lu** (with possible intervening whitespace and comma pauses) as phonetically valid. Use of double quotes is optional (and the parser will keep track of correct nesting, which is not the case for the alien text quotes). The grammar will further enforce that what is enclosed in these quotes is a grammatical Loglan utterance. Similarly, the use of parentheses in the format **kie (...)** **kiu** in a parenthetical free modifier is supported, and can be nested.

7 Lexicography (a formal Proposal as of 2/3/2018)

Having written a complete phonetics proposal, I'm writing a complete lexicography proposal. The subject of this essay is the Loglan "word". As we will see, this does not precisely accord with the definition given in NB3, though that is a place to start. In NB3, JCB said that the fundamental hallmark of a word is that one cannot pause in the middle of it (without changing its meaning). In Loglan 1 (1989), however, JCB allows pauses after borrowing *djifoa* in predicate words, which nonetheless clearly remain words.

It should also in general always be possible to pause before and after a word. Further, our intention (perhaps not perfectly realized) is that when whitespace represents a pause, not forced by phonetics to represent a pause, and omission of the whitespace would form a word, mere whitespace is not permitted: a comma-marked pause will be as a rule required in such a situ-

ation.

In earlier material of ours, we have maintained that classes PA and PANOPAUSES (in which we do allow pauses) are word classes. We now think the analysis suggests that the word units in these classes are smaller (though not always monosyllables) so we expect not to list these as exceptions to JCB's criterion in this document. We still view pauses in class NI between digits as not being word breaks, and so constituting an exception. We introduced the possibility of pausing and resuming in acronyms, which we will present here as an exception; we also, however, now deprecate the use of Loglan 1989 style acronyms in favor of using names.

In the sister language Lojban, it is said that it is possible to pause anywhere in a stream of unit *cmapua* without affecting meaning, and so that there are no multisyllable *cmapua* words. We believe that this is an elegant situation, but also that TLI Loglan definitely has not achieved it. We do have multisyllable *cmapua* words, and we will give an analysis of these in this document. There are some multisyllable words which *must* be multisyllable words on pain of ambiguities, and there are some which we view as multisyllable words as a matter of style: some unit *cmapua* seem to us clearly to be affixes in the proper linguistic sense rather than words, and we do not see value added in being allowed to pause before them.

7.1 Name Words (and Alien Text)

The description of name words is simple. These are the name words and acronym words of the phonetics section.

Djan is a word of this class. **DaiNaizA** is a word of this class. Words of these classes are followed by pauses.

The use of the 1989 (or earlier) acronym words is now in my mind deprecated (though still supported): I prefer the use of name words proper obtained by suffixing **-n** to the original acronym words in place of these (as for example **DaiNaizAn**), and I have made grammatical arrangements to support this in the case of dimensions.

I regard the name marker words and alien text marker words as words, separate from the names or alien text which follow them. This analysis might be open to debate. A chunk of alien text is not by its nature a Loglan word! The **y** which separates chunks of alien text is I suppose a Loglan word.

I have been more conservative in this implementation of Loglan grammar (built on the phonetics proposal) about how many name markers there are.

- la:** The article which constructs names. A name marker. It can also appear as an article in descriptions.
- hoi:** The principal vocative constructor. A name marker. It can also be followed by other grammatical forms. The words of social lubrication are also vocative markers, but are not name markers.
- hue:** The inverse vocative constructure. A name marker. It can also be followed by other grammatical forms.
- ci:** A general “verbal hyphen”. Because it can be used between items in serial names, it is a name marker.
- liu:** The word quotation operator. A name marker. The non-Loglan word quotation marker **niu** is not a name marker, and is not likely to be used before name words.
- gao:** A proposed operator which converts words to letterals. Because it can be used with names, as in **gao Alef**, it is a name marker.
- mue:** A new name marker: this is used before a name word used as a dimension. **mue** appears as a unit in acronyms of the 1989 form, but in those contexts it is not a word.

Here are the alien text marker words and **y**.

- lao:** The foreign name constructor.
- lie:** The strong quotation operator. It should be noted that our grammar for strong quotation is totally different from the 1989 grammar of strong quotation: it is modelled instead on the 1989 grammar of **lao**.
- sao:** The foreign predicate constructor.
- sue:** The onomatopoeic predicate constructor: **sue miao** to meow. **sue sss** to hiss, **sue ccc** to shush.
- hoi, hue:** The vocative and inverse vocative markers can be used as alien text markers. When they are so used, the following alien text must be enclosed in quotes in writing. This avoids the problem of text with errors in it which should fail to parse parsing unintentionally as alien text.
- y:** A word used to separate blocks in a single item of alien text.

7.2 Predicate Words

The description of predicate words is straightforward, though it is not quite as simple as “the phonetic predicate words are the predicate words”.

To begin with, the phonetic predicate words (including those with internal pauses after stressed borrowing djifoa) are words. To give an account in which units ending with stressed borrowing djifoa were themselves words would be a complication, not a simplification. Such an account would be made easier by our requirement (not stated, or not made clear in the original proposal) that what follows such a pause must itself be a well-formed complex, but this sort of account would add no value to the grammar.

Numerical predicates are pause-free NI cmapua phrases (described below) followed by one of **ra**, **ri**, **ro**. **N-ra** is the predicate of N element sets. **N-ri** represents the predicate “X is the Nth item in series Y”. **N-ro** (a 1990’s innovation) is used to qualify other predicates: **N-ro preda** means Nth most **preda** (**nero gudbi** is “best”, **toro gudbi** is “second best”)⁴³. These are genuinely words because pausing in the middle would produce a description of a certain quantity of items described by a different predicate word. **tetora** is the predicate “is a set with thirty two elements”; **te tora** is an indefinite description, “three pairs”. In 1989 Loglan, numerical predicates had to be penultimately stressed; we have partially implemented this requirement in our parser (there is some freedom of stress placement in some cases). In any case this is not required to recognize where these words start or end. More information about the structure of these words will be found in the discussion of NI phrases later in this document.

The words

bia: (is part of),

bie: (is a member of (a set)),

cie: (is less than (math)),

cio: (is greater than (math)),

bi: (is defined as)

⁴³Some discussion of the place structure of these predicates is wanted.

are all predicates semantically, though they are structure words phonetically. They form a grammatical class BI of “identity predicates” (not a terribly accurate description).

I propose adding to the class BI all the forms obtained by prefixing **nu**, giving converse operators (my parser allows this). These converse forms are words (the grammar does not allow **nu** to be applied to identity predicates as to normal predicates).

The words

he: (interrogative predicate; a sentence with a **he** in it is a question with a predicate answer),

dua: (first free predicate variable),

dui: (second free predicate variable),

bua: (first bound predicate variable),

bui: (second bound predicate variable)

are grammatically ordinary predicates, though phonetically structure words. None of them are really very ordinary predicates!

The acronyms, which were predicates in 1989 Loglan, are treated as names under our proposals (and in fact we suggest phasing them out and using names for their purposes).

7.3 Structure Word Classes Introduced

I cannot make this a section because LaTeX doesn’t allow sections to be nested beyond a certain depth, so this is a short section introducing the structure word sections which follow.

Most but not all cmapua words are single syllables (one unit cmapua). Many compound forms often written without spaces, such as **lemi**, “my”, actually fall apart into two words. But some do not. Of these, some simply **cannot** be so viewed, and some we think do not make sense as compound words. We will discuss all these cases.

7.4 Tight logical connectives: CA roots

There is a series of logical connectives which must be presented first, as words (or affixes) of this class appear as components of elements of many other classes (including some complex logical connectives!)

The root words of class CA are

ca: and/or

ce: and

co: if and only if

cu: whether or not

nucu: converse to **cu**

ciha: interrogative quantifier

ze: mixture

CA roots may be prefixed with **no**, indicating negation of the first connected item, or suffixed with **noi**, indicating negation of the second connected item. Such a structure is called a CA core (a CA root optionally decorated with initial and/or following negations).

7.5 Letters, acronyms, and pronouns

A Loglan upper case consonant letter is **Cai**. A Loglan lower case consonant letter is **Cei**. A third series **Ceo** is provided for lower case Greek letters. Further series **Caiu** and **Ceiu** are provided: **QqWwXx** are **Kaiu**, **keiu**, **Vaiu**, **veiu**, **Haiu**, **heiu**. What the other new letters are, who knows?

A Loglan lower case vowel has the form **ziV**, and the upper case form is **ziVma**. The old style forms **Vfi** and **Vma** are fully supported in the parser, though we are not fond of them. These include **yma**, **yfi**. The **Vzi** form for lower case Greek letters is supported. The VCV letterals are multisyllable *cmapua* words.

The primary use of the letters in Loglan is *not* as names of phonemes but as **pronouns**. As a pronoun, a letter refers back to a recent argument with the same initial letter. There is a convention favoring using capital letters to refer back to proper names and lower case letters for general descriptions.

There is a further class of atomic pronoun words

tao: (that [of situations]),
tio: (this [of situations]),
tua: (???tu ze da. this may be obsolete),
mio: (we (first + third), independently),
miu: (we (first + third) mass),
muo: (we (first + second+third) independently),
muu: (we (first + second + third) mass),
toa: (this [of text]),
toi: (that [of text]),
too: (you, plural, independently),
tou: (you, plural, jointly),
tuo: (you and others independently (2+3)),
tuu: (you and others (2+3) mass),
suo: (self),
hu: (interrogative pronoun),
(ba, be, bo , bu): series of indefinite [quantified] pronouns,
(da, de, di do du): the series of old-style definite pronouns,
mi: (I),
tu: (you),
mu: (we (1+2) mass),
ti: (this),
ta: (that),
mo: (we (1+2) independently)

The anaphora convention for the series **da, de, di, do, du** can be read about in L1. The idea is that these words live on a stack in alphabetical order (those that are not already in use) and the *n*th description back in the text not already bound to a pronoun will be bound to the *n*th letter on this stack when needed. It seems rather baroque but very simple cases can surely be used correctly. We note also that the existence of the digit-suffixed forms should make it easier to use this system.

The general class of pronoun words consists of letters or other pronouns, optionally suffixed with **ci** followed by a NIO unit (usually a digit; see the section below on numerals and quantifiers). It is advisable to pause after a digit suffixed pronoun like **dacine** (because **ci** is a name marker; with a little work in the parser I might be able to ensure that such occurrences of **ci** are not tagged as name markers). The numerically indexed pronouns are multisyllable *cmapua* words. It is very important to notice that for us a pronoun is a **single letter**, possibly suffixed with a single digit. Multiletter variables lead to horrible ambiguities which do serious grammatical damage. Multiletter pronouns are in fact supported by LIP but there is language in NB3 which suggests that JCB did not intend to have them, and we *strictly forbid multiletter pronouns (repetition deliberate)*.

The reason that it is vitally important **not** to allow multiletter pronouns is that the use of a sequence of individual letters as a sequence of pronoun arguments without the inconvenience of having to pause after each one is grammatically far more important than any use of sequences of letters as single pronouns or acronyms.

Further letter words, which may be used as pronouns, but to which we may not attach numerical suffixes, are generated by **gao** followed by a single well-formed word, either a name, a predicate, or a consonant initial unit *cmapua* (CVV or CV). This is a proposal of John Cowan, intended to provide names for letters in alien alphabets.

7.5.1 Remarks on acronyms

An acronym is a sequence of letter names (possibly abbreviated in the case of vowels to *zV* – not to just *V* as in older versions of the language – which eliminates distinctions of case of course; corrections of *V* to *zV* in acronyms may be required in old texts), and number names (atomic quantifier words or numeral units), beginning either with the acronym marker **mue** [a proposed feature] or a letter (possibly abbreviated) and having more than one com-

ponent (the dummy **mue** allows the formation of one letter acronyms and also of numeral initial acronyms without confusion with numerals or letterals). Acronyms are used to form dimensioned numbers (as discussed below) and to form acronymic names (no longer acronymic predicates – a proposal of course). The initial marker **mue** ensures that dimensioned number acronyms are not confused with sequences of pronouns, and the fact that acronymic names are **names** ensures that they are head marked in a way which ensures that they cannot be confused with sequences of letter pronouns. Acronyms must always be marked with **ci** when used as components of serial names or name-final descriptions. A pause, terminal punctuation, or end of text is required after an acronym (so it can never attempt to consume a following letteral pronoun). One can pause inside an acronym and resume if the pause is immediately followed by **mue**; this corrects for problems of resolution of sequences of letterals, especially where the VCV forms are involved.

We add as a footnote a remark on why we do not like the VCV letterals. When VCV letterals are used in acronyms, as in **la daiafi**, the analysis of this into phonetic cmapua units has to be **daia-fi**, not coordinated with the semantic analysis into **dai-afi**. I did take the trouble to make sure that though one must pause before VCV letterals where they appear as words rather than acronym components, one does not need to explicitly comma pause; they are treated in the same way as vowel-initial predicates.

We currently propose that the use of acronyms be replaced by the use of actual name words, formed by appending **-n** to the legacy acronyms, and we have made grammatical arrangements to support this usage

7.6 Numerals and quantifiers

The numerals in Loglan are

ni: (0),

ne: (1),

to: (2),

te: (3),

fo: (4),

fe: (5),

so: (6),

se: (7),

vo: (8),

ve: (9).

Other words of the atomic quantifier word class NIO are

kua: (division)

gie: (left bracket),

giu: (right bracket),

hie: (left parenthesis),

hiu: (right parenthesis),

kue: (inverse division),

nea: (unary minus sign) ,

nio: (subtraction),

pea: (unary plus sign),

pio: (addition),

suu: (root),

sua: (exponent),

tia: (times),

zoo: (double prime),

zoa: (prime),

pi: (decimal point),

re: (more than half of (quantifier)),

ru: (enough of (quantifier)),

hi: (close comma),

ho: (interrogative quantifier)

The closely related RA class contains

ra: (all),

ri: (few),

ro: (many);

these words are distinct because they have a different meaning when they appear as a suffix to a quantifier word (a quantifier word with a suffix with the phonetic shape of a RA word is a numerical predicate, for which see below).⁴⁴

The SA class of quantifier prefixes consists of

sa: (about/approximately (prefix to a quantifier, by itself **sara**),

si: (at most, prefix to a quantifier, by itself **sine**),

su: (some/any/at least (quantifier prefix) by itself **sune**),

sinoi: (more than; a prefix to a quantifier, by itself **sinoine**???) new proposal),

sunoi: (less than; a prefix to a quantifier, by itself **sunoira**???) new proposal)

sanoi: supported by the grammar, and its meaning is deducible, but seems not likely to be used.

The SA-**noi** forms are multisyllable words or units in multisyllable words: all uses of **-noi** are as affixes.

We moved **ie** (who/what/which?) to class SA and eliminated all special references to it as a class. Note that it could attach to somewhat higher level argument classes in the old grammar, but it can still attach to them in the

⁴⁴This dual use of the RA words has been corrected in Lojban, but we believe we are stuck with it: it is just one of the peculiar charms of the original Loglan. It seems possible to us that it might be wise to put **re** and **ru** in this class as well.

form **ie me** under the new arrangements. In fact, any word in class SA other than **ie** itself can be prefixed with **ie** to give a new element of class SA (this was needed to support **iesu**, which appears in Notebook 3). Further, **ie** may be succeeded by a pause in all cases; phonetics officially forbids a “word” in the proper sense which contains VV units and other sorts of unit cmapua.

We give semantics for these words briefly, but we do not envisage incorporating any official grammar of mathematical expressions into TLI Loglan; such a grammar might be desired by a group of users of the language, and they can develop their own for local use.

We handle the items **ma** and **moa** (00 and 000) differently than in earlier descriptions of the language. We define a class NI1 of numeral units consisting of a numeral (any word of class NI0 but this really makes sense only for the digits⁴⁵ followed optionally by **ma** then optionally by **moa**, and a digit may optionally follow **moa**. D **ma** means D followed by two zeroes; D **moa** means D followed by three zeroes. D (**ma**) **moa** n means D followed by (2+) 3n zeroes. Originally, **ma** and **mo** were words of class NI0 meaning 00 and 000. **mo** is overused for other purposes, so we changed it to **moa**, and the use of an exponent seems better than repeating it. Replacing **mo** with **moa** is occasionally necessary in old texts.

A quantifier core (class NI2) is a sequence linked by CA cores of items of the following kinds (the items linked may further optionally be suffixed with **noi**):

SA: A SA word.

numeral block: A sequence of one or more NI1 words, with internal whitespace or explicit pauses permitted. It may optionally be preceded by a SA word.

RA: A RA1 word, which may optionally be prefixed by a SA word (this last option is a change from 1989 Loglan). A RA1 unit is a RA word suffixed with **mo** and/or **moa** optionally followed by a numeral, to give forms with meanings like “several hundred”. Question: how do we say “several dozen”? Or do we? It is important to note here that **sara**, for example, is not a numerical predicate, but a quantifier; the 1989 Loglan predicate **sara** becomes **sarara**. Replacements of things like **sara**, **sira**

⁴⁵You live and learn: in the Visit I found a need for forms like **rimoa**, a few thousand, so there is also a class RA1.

with (resp.) **sarara**, **sinera** is an occasional correction needed in old texts.

A general quantifier word has a quite complex definition. It begins with a quantifier core as described above. This may optionally be followed by an acronym which must start with the marker **mue** [or by the word **mue** followed by a consonant final name word]; if this is present it is the last element in the word and is followed by end of text, terminal punctuation or an explicit pause. There is a final option of appending **cu**. Old Loglan texts will not have the marker **mue** before dimensions; this may need to be inserted.

General quantifier words are regarded as multisyllable *cmapua* words, even when they contain pauses between NI1 units.

The suffix **cu** (a late proposal of the last Keugru) generates indefinite mass or set descriptors from quantifiers (which are themselves grammatically a species of quantifier). I have to think carefully about whether this construction really describes a set as JCB says or a mass object; JCB, especially in later periods, tended to confuse the two.

The acronym suffixes create dimensioned numbers. The initial marker **mue** is a proposal of ours.

Quantifiers have important grammatical uses in the language, to be revealed below. This is quite a separate issue from having a complex internal grammar of quantifiers/numerals, which we avoid⁴⁶. The word “mex” (abbreviating “mathematical expression”) is used in the grammar section for quantifier words.

7.7 Tense/location/relation operators

The root words of this class (which we call PA words (or phrases) for short) are

gia: (time free continuous tense, -ing),

gua: (timeless habitual tense),

pia: (past continuous tense, until [before terms]),

⁴⁶2/3/2018 I am considering some simple grammar of quantifier cores. Not implemented, I am just thinking about it.

- puu:** (was habitually -ing, continuous past tense),
- nia:** (continuous present tense, during [before terms]),
- nua:** (am now habitually -ing, continuous present tense),
- biu:** (possibly, under conditions X [before terms]),
- fea :** ...happens in the same possible world(s) as...(actuality, in the sense of Kripke models of possible worlds). Not necessarily an official part of Loglan.
- fia:** (will be -ing future continuous tense, since X [before terms]),
- fua:** (will habitually be -ing, future continuous tense),
- via:** (throughout a place of medium size),
- vii:** (throughout a small place),
- viu:** (throughout a large place),
- ciu:** (X ga Y ciu Z means Z ga Y as much as X ga Y) [left here for the moment but actually moved to class KOU in 3/9 fix],
- coi:** (according to rule X),
- dau:** (probably, likely under conditions X),
- dii:** (for, on behalf of X),
- duo:** (by method X),
- foi:** (X foi Y, X must Y, X ga Y foi Z, X must Y under conditions Z – Y a predicate),
- fui :** (should, same structure as foi),
- gau:** (can (same structure as foi?)),
- hea:** (by, with the help of, X),
- kau:** (can, is able to (structure of foi)),

- kii:** (with/accompanied by X),
- kui:** ...is accessible from...(in the abstract sense of Kripke models, possible worlds). Not necessarily an accepted part of Loglan. I am now quite in favor of using this to build null prepositional phrases in order to force an indefinite variable such as **ba** to have a larger scope, as in **Mi djano lezo ba mormao la Djan, guo kui raba**, “I know who killed John”, which literally means “I know for each x to what extent x killed John”; without the **kui ba** it would just mean “I know the extent to which it is true that someone killed John”.
- lia:** (like, in the way that – I suggest that X ga Y lia Z means that X ga Z as Y ga Z, but X ga Y lia lepo Z ga W means X ga Y as Z ga W),
- lui :** (for, in order to please X),
- mia:** (subjective subjunctive, mia lepo X = were X the case),
- mou:** (more than, structure of ciu) [left here for the moment but actually moved to class KOU in 3/9 fix],
- nui:** (may/is permitted to, structure of foi),
- peu:** (as for/concerning X), roi (X roi Y = X intends to Y; X ga Y roi Z = X intends to Y under conditions Z),
- ru:** ...obligates/makes it necessary that... from a counterfactual proposal. Not in the dictionary; not necessarily an accepted part of Loglan.
- sea :** (instead of X),
- sio:** (certainly, certain under conditions X [before terms]),
- tie:** (with/through/by means of instrument X),
- va:** (in the middle distance, near X),
- vi:** (here, at X),
- vu:** (far away, far from X),
- na:** (now, present tense, at the same time as X),

pa : (past tense, before X),

fa: (future tense, after X)

pau: (ago): added 11/14/2015 to support its use in A First Visit to Loglandia. I am not convinced that we need this *cmapua*.

and the related small class of KOU roots

kou: (because (cause) of X),

moi: (because/in order to (motive) of X),

rau: (because (reason) of X),

soa: (because(logical premise) of X)

ciu: (X ga Y *ciu* Z means Z ga Y as much as/to the same degree as X ga Y)

mou: (more than, structure of *ciu*)

which can be prefixed with **nu**, **no**, or **nuno** to give additional forms which we call KOU cores (a root is also a core). A KOU core is a multisyllable *cmapua* word.

It is important to notice that **nokou lepo X** does not deny X; in fact, it asserts X and says that the main event happened in spite of X. Forms like **nukou** are converses: they are versions of “therefore X”. Forms like **nunokou** are versions of “nevertheless X”; X happens, but not because of the main event, rather in spite of it.

The forms with initial **no** are obligatory words: pausing between the syllables of the word **nokou**, for example can radically change the meaning of a Loglan utterance. **Mi cluva la Meris, nokou Mai bilti** means “I love Mary, but not because she is beautiful”. **Mi cluva la Meris, no, kou lepo Mai bilti** has the same meaning as **No, mi cluva la Meris, kou lepo Mai bilti**, “It is not the case that I love Mary because she is beautiful”: it is possible that I am not saying that I love Mary at all. In the second sentence, use of an explicit comma to indicate this unexpected usage is required. **Mi cluva la Meris, no kou lepo Mai bilti** will not parse.

The words **ciu** and **mou** were moved into class KOU, to support formation of negative and/or converse forms of these words which are described

in Paradigm K on our web site, though they never seem to have been implemented in LIP. The new “causal connectives” **mouki** and **ciuki** (and relatives) created by this move may have uses (I like them very much!).

We propose (with implementation) that PA roots other than KOU roots may be converted with initial **nu-** and/or negated with final **-noi**: these forms enter into all subsequent constructions as PA units (these may be called PA cores). These forms may further be prefixed with non-logically connected NI words (also producing PA cores). The new forms are multisyllable cmapua words (except that pauses are permitted between digits in the optional initial NI segment): this is a case where pauses would not be harmful but it does not seem to me that the **nu** or **noi** are functioning as freestanding words: they are affixes in the proper sense of that word. The conversion and negation forms for KOU roots remain as before (and KOU cores may be further prefixed with non-logically connected NI to obtain more KOU cores). Replacing **nokou** with **kounoi** is probably a good idea, but this would involve extensive changes in existing text. A PA or KOU core may be further decorated with a qualifier of class ZI (**za**, **zi**, **zu**), still obtaining a PA or KOU core (to see the effects of these qualifiers on tense and location operators, see the dictionary).

The class PA of phrases used as tenses or standalone modifiers consists of strings of PA or KOU cores (which may or may not be separated by pauses), such strings possibly linked with CA cores to further such strings.

The class PANOPAUSES used in modifiers with an attached argument consists of PA or KOU **words** (strings of PA/KOU cores not separated by pauses) possibly linked with CA cores to further such words.

Strings of PA/KOU cores not separated by pauses are viewed as multisyllable cmapua words.

When a PA class phrase is followed by a PANOPAUSES class phrase, an explicitly marked comma pause must intervene.

We think that the intent of **Mi smarue pa, vi le kruma** and **Mi smarue pavi le kruma** is different. The pause in the first sentence must be explicitly marked.

If it is not desired to draw the distinction between PA and PANOPAUSES, the grammatical solution would be to require that a PA phrase used as a modifier must be closed with **gu** when followed by another modifier. In this case, the form of PA and PANOPAUSES phrases would be that given for PA phrases above, and each PA/KOU core would be a (possibly multisyllable) cmapua word.

The semantics of complex PA words will require a considerable essay, to be inserted here in due course. In particular, a summary of the location and tense words and their interaction with **-zV** suffixes is needed, since these have some ad hoc features. **pazu** a long time ago versus **panazu** in the past for a long time interval is an example I insert to remind myself.

These phrases can be used as prepositions (followed by an argument) or as tenses in the broadest sense (followed by a predicate): note the difference in phonetic form between these two uses, indicated above. The word **ga** is a content free tense word not usable as a preposition. **ga** has other uses as well. Details of this will be seen in the grammar.

Where a PA word occurs as a suffix to another word form (with attached explicit pause), it is generally illegal for it to be replaced by whitespace followed by a PA word in turn followed by an explicit pause: where a PA suffix is legal, it cannot be replaced by a following PA word without an explicit pause being indicated. **Da na clivi, o na brute** (an example in L1) does not actually parse correctly with LIP because of lexer problems with APA words; an unintended **ona** is read. It parses correctly as written under the current parser. **Da na clivi, o na, brute** fails to parse under the current parser, because the given pause pattern is in danger of creating an **ona**. **Da na clivi, o, na, brute** does parse as intended.

7.7.1 The system of tense and location words

Here we will lay out the system of compound tense and location words, indicating difficulties and possibly some suggestions for improvement.

The basic series of tense words is **pa, na, fa**, which mark present, past, future tense when they mark a predicate; **pa X, na X, fa X** mean before X, at the same time as X, after X, respectively.

A second series of tense words **pia, nia, fia** express continuous tenses. **pia preda** means “was preda-ing”. **pia X** means “until X”. **fia preda** means “will be preda-ing”. **fia X** means “since X”. **pia preda** means “was preda-ing”. **pia X** means “until X”. **nia preda** means “is preda-ing”. **nia X** means “during X (throughout)”.

A third series of tense words **pua, nua, fua** express habitual tenses. Their meanings are similar to those of the previous series, but they refer to events which often or usually happen during an indicated period rather than events which happen continuously during an indicated period.

These words can be compounded. Here are the dictionary meanings of compound tenses.

papa: had (been)... ed, sign of the past perfect tense.

pana: was/were then... ing, sign of the past coincident tense.

pafa: was/were going to..., sign of past progressive tense, english inexact

napa: has/have (been).../a..., sign of the present perfect tense; already

nana: am/are/is now... ing, sign of the present coincident tense.

nafa: is/are going to..., sign of present progressive tense, English inexact.

fapa: will have... (been) ed, sign of the future perfect tense.

fana: shall/will be then... ing, sign of the future coincident tense.

fafa: will-be going to..., describes an action which takes place after the (future) time being recounted.

These words can be qualified with the suffixes **zV**. Here are the dictionary entries.

pazi: just... ed/was just (now a), a modified tense operator; just before..., before event terms.

nazi: at/coincident with..., an instant in time; at the time when, momentary event clauses.

fazi: will immediately (be a)..., modified tense operator; just after, before event terms.

paza: lately/newly/recently... ed, not too long ago, a modified tense operator; shortly before..., before event terms.

naza: during/in..., in some short interval, with terms.

faza: will soon (be)/be about to/just going to..; shortly after, with clauses.

pazu: long before, some event, before clauses.

nazu: during, in some long interval, with terms; while, during some long event.

fazu: will eventually (be a), a modified tense oper.; long after, some event, before terms.

The dictionary definitions are not fully systematic. Notice that **nia** and **nazu** express different meanings of “while, during”. I think in spite of some ambiguity about **nazV** forms, that the **zV** operators do something uniform, qualifying the distance of the event from the argument (or the present in the case of tenses). **nazu** doesn’t say that the event actually is far from the present, but since it says the event is in a long interval around the present it permits a long distance from the present.

Continuous examples are also listed

piazu: for all that time until now, adverb and before preds; long-before then and until, with clauses.

niaza: while/throughout the short time, clauses.

niazu: while/throughout the long time, clauses.

fiazu: since, for a long time after, with clauses.

The basic series of location operators is **vi**, **va**, **vu**, at/near/far from.

The second series of location operators is **vii**, **via**, **viu**, throughout a small/medium/large sized place.

Here are the compounds listed in the dictionary.

vivi: around, in the place where, before terms.

viva: out of where, a short way, with clauses.

vivu: out of, for a long way, before terms.

vavi: into where, from nearby, before clauses.

vava: past where, nearby, before clauses.

vavu: away from, from near to far, before terms.

vuvi: into where, from far away, before clauses.

vuva: toward the place where, before clauses.

vuvu: past where, at a distance, before clauses.

Modifications with **zV** affixes:

vizi: right here/at this spot, before preds; at the spot where, with point like events.

vazi: near this spot/the spot where, of point like events, before predicates.

vuzi: far from this spot, before predicates; far from where, spatially limited events.

viza: in this place/small region, before preds; where, before spatially limited events.

vaza: near this place, before predicates; near the place where, of limited events.

vuzu: far from this place, before predicates; far from where, of medium sized events.

vizu: in this place/big region, before preds; where, before spatially extensive events.

vazu: near this region, of extensive events, before predicates; near the place where, of extensive events.

vuzu: far from this region, before predicates; far from where, of extensive events.

The difficulty here is that there really isn't a system as such – at least, if there is, it is only implicitly given. It is possible to extrapolate from this, and it is also possible to compare with the sister language Lojban, in which an effort has been made to systematize these issues.

Another point is the status of the qualifiers **zV**. These are affixes, and one of these terminates a PA core, for us. In LIP, these affixes seemed to terminate PA words. Thus we allow **pazicevuzu** and LIP does not.

It is clear that a lot more words are formally possible, both for my grammar and for LIP.

7.8 Connectives

There are numerous parallel classes of logical and causal connective words in Loglan. Here we are only talking about binary logical connectives like English “and”; the word **no** for the unary negation connective is the sole inhabitant of a separate word class of its own.

7.8.1 Logical connectives for sentence components

The basic series of connective roots is **a, e, o, u, nuu, ha**. These are words by themselves, but certain affixes can be attached to them to build a large class of words. One can add the prefix **no** and/or the suffix **noi** to an A root to obtain an A core.

We describe the class A of basic logical connectives. The root is taken from **a, e, o, u, nuu, ha** (possibly with prefixed **no** and/or affixed **noi**, i.e., an A core). A complete PA word (a tense in the broadest sense) with no internal pauses or spaces may follow as a suffix; finally, if and only if a PA component is present, **fi** or a full comma pause must close the word. An A word may not be followed without intervening space by a PA word (with no internal pauses) then whitespace: this is purely a technical device to detect unclosed APA words in legacy text. It is worth noting that in the NB3 corpus, JCB appeared to be following a rule of closing IKOU words with commas as one would expect here (though not APA words).

All A words are preceded by explicit comma-marked pauses. The phonetic reason for this exists only when the words are vowel-initial, but the rule is enforced for all words of this class.

It should be noted that our treatment of APA words is a new proposal. These words present considerable difficulties in LIP, and have been abandoned entirely in Lojban. We have preserved them so far because they are common in the NB3 corpus and in the Visit to Loglandia, and because the related IKOU words, which present much the same difficulties of termination, are clearly not dispensable without doing some violence to the corpus. I have tried a couple of different solutions: my aims here are to produce a solution which will allow parsing of legacy text with minimum violence (some pauses) and which will impose no unexpected obligations to pause on a speaker who always closes APA words and their relatives with **fi**.

a means “or” (the inclusive and/or). **e** means “and”. **o** means “if and only if”. **u** means “whether or not”. **nuu** is the converse of **u** in the obvious

sense. **ha** is the interrogative quantifier; an utterance with **ha** in it is a question which calls for an A word as an answer. Compounds built with **ha** are not excluded by the grammar but certainly would be odd.

Prefixing **no** has the effect of negating the part of the logically connected utterance before the A word. Suffixing **noi** has the effect of negating the part of the logically connected utterance after the A word.

Suffixing a PA word has different semantics depending on whether or not the PA word is a KOU word. X, **efa** Y means X and then Y while X **erau** Y means X because Y, and careful analysis reveals that the first is **fa** X, Y while the second is X, **rau** Y. This is a slip, but we suggest following Lojban and keeping it this way. The alternative would be to have **epa** mean “and then”.

We now describe other series of connectives. The ACI and AGE connectives consist of an A connective, with any pause or **fi** after a PA word omitted, followed by **ci**, **ge** respectively. These connectives differ from A in precedence; their uses will be discussed in the grammar proper. They must be preceded by a pause, just as in the case of A connectives.

The CA connectives are another related class (already briefly introduced above). They are not preceded by pauses. The CA root forms are **ca**, **ce**, **co**, **cu**, **nucu**, **ciha**, **ze**. A CA root or a CA root with a prefix **no** and/or a suffix **noi** is a CA core. The semantics of **ca**, **ce**, **co**, **cu**, **nucu**, **ciha** are analogous to those of the A forms (and adding the **no** and/or **noi** has the same effect). **ze** builds composite objects or mixed predicates; its semantics are entirely different.

A CA connective word may take all the forms of an A connective with the A root component replaced by the corresponding CA component. A preceding pause is not required. The word **ze** has uses which a general CA word does not have (it can connect arguments). I am contemplating the formal possibility of **zenoi** and wondering if it might be useful.

The precise extent of the system of logical connective words here is not the same as that supported by LIP, but it is close. The scheme here allows more CA words than LIP does; we will see if they are useful.

7.8.2 Sentence connectives and new utterance markers

The connectives given so far connect arguments and predicates. We now consider connectives which connect sentences.

The word **i** (always preceded by a pause) begins a new utterance, but can

often be treated as if it were a high level logical connective meaning roughly *e.* Further words of the same class I can be constructed by appending a PA word as a suffix, which must be closed with **fi** or a comma pause. The same issue exists for semantics of IPA words that is discussed above for APA words. All words of this class are preceded by a phonetically mandated comma-marked pause.

A word of the class ICA consists of I followed by a CA connective word. This is a logical connective acting between sentences. Because it is vowel-initial, it must be preceded by a comma marked pause.

An I or ICA word cannot be followed by whitespace then a PA word (an explicit pause is needed to separate a sentence initial PA word from the I or ICA word).

There are further forms ICI and IGE constructed from words of class I or ICA by appending **ci** or **ge** (after removing closures on component PA words).

The closure of logical and sentence connectives with **fi** is a new proposal here (I used **gu** earlier, but it creates conflicts, and I have experimented with different pause conventions).

7.8.3 Forethought logical and causal connectives

The root forethought logical connective forms are **ka**, **ke**, **ko**, **ku**, **nuku**, **kiha**, each possibly followed by **noi**. The root KOU words are **kou**, **moi**, **rau**, **soa** [**under a proposal also ciu**, **mou**] (optionally prefixed with **nu**, **no** or **nuno** to give forms which we call KOU cores (roots are cores too)), of which we will have more to say later. The forethought logical connective words of class KA are either one of these root words, or a KOU core, followed by **ki** then possibly **noi**. These forms appear before the first of the two items connected, with **ki** or **kinoi** appearing between the two items. Forethought connectives can connect almost any grammatical structure that can be linked by logical connectives. Note that forethought analogues of APA words are not provided; they did exist in LIP and could easily be restored if wanted.

The force of the causal connectives such as **kouki X ki Y** is (for example) X and Y (because of X). **nokouki X ki Y** is (for example) X and Y (not because of (in spite of) X). Note that the initial **no** is not negating X or Y, they are both asserted!

The new connectives **mouki** and **ciuki** have fairly clear meanings: **mouki X ki y**, “X more than Y”. **Mi cluva mouki la Meris, ki la Selis**, “I love

Mary more than Sally”. **Mouki mi cluva tu, ki tu cluva mi**, “It is more the case that I love you than that you love me”.

How these words are *used* will be discussed below in the grammar.

7.9 Articles

The basic articles (constructors of definite arguments) are

lea: article for sets: the set of all things with property ...

leu: The particular set I have in mind of things with property...

loe: The typical...

lee: The one or more things I mean which actually are...

laa: The unique object which actually is... (the logical definite description).

le: The default article. The objects(s) understood from context which the hearer will be expected to think have property X...

lo: The mass article (describes composite objects made of all the objects designated).

la: The article for proper names.

These are now all the words of this class. The former construction of complex words of this class by following the root with an optional pronoun followed by an optional PA suffix has been superseded by a modification to the grammar class **descriptn**.

The name constructor **la** appears in the list above but appears in special constructions as well. The precise ways in which names are handled in this grammar involve new proposals.

There is a special class LEFORPO consisting of **le**, **lo**, and the quantifier cores (NI2) which may appear followed by PO in the formation of abstract descriptions. Notice that no new words are involved. It is worth noting that **lepo** and related forms are not single words, though they are often written without a space, and so can be written **le po** or even **le, po**.

Details of the use of these classes belong in the grammar below.

lau, **lua** and **lou**, **lou** are paired forms beginning and ending unordered and ordered lists, respectively.

7.9.1 Constructions involving alien text and related articles (see the appendix to the Phonetics Proposal for some modifications)

In this subsection we introduce the articles which handle quotations and imported foreign text, and we also give the full constructions of arguments (and predicates) of this kind. The strong quotation construction that we give is a completely new proposal.

Any well-formed Loglan utterance *X* can be quoted **li X lu**. *X* may be preceded and followed by explicit pauses (commas) if desired (this is not required). Under the Phonetics Proposal, we have not yet restored quotation of serial names (which are not utterances by themselves, though they are when marked) using **li/lu**, though we may do so. **li** is not a name marker word. I am contemplating allowing **li** to quote a descpred followed optionally by a name (this construction may now be the basis of a vocative or inverse vocative) but this seems less likely to be needed. Quotation marks may be inserted after **li** and before **lu** (and must match: if in one place, then also in the other).

A single Loglan word *X* may be quoted **liu X**. This is the only context in the grammar where the phonetic class of structure words plays any role. In LIP it plays no role even here, as LIP apparently only allows **liu** for actual *cmapua*. Lojban I believe only allows unit *cmapua* to be quoted; we admit that there are compound words, so we allow them to be quoted. A **liu** construction must always end with an explicit pause (a new proposal, concurrent with the master Phonetics Proposal). **niu** may be used instead of **liu** to explicitly signal that a quoted word, though phonetically acceptable, is not a Loglan word. The Phonetics Proposal allows **liu** to quote marked names (as **liu la, Djan** or even **liu la, Djan Braon**) and alien text constructions (as **liu sao word**).

One may refer to a letter (rather than use it as a pronoun) using the form **lii X**.

The further forms discussed here operate on alien text. Alien text will be a block of text beginning with whitespace or an explicit pause and ending with whitespace, an explicit pause (comma), or before terminal punctuation or end of text, and containing no commas or terminal punctuation otherwise. It may contain other symbols or non-Loglan letters. Initial and final whitespace must be expressed phonetically as a pause.

The article **lao** followed by one or more blocks of alien text, with blocks

being separated by **y** set off with spaces (which must be pronounced as explicit pauses) if there is more than one block, forms a foreign name. Wherever names are to be written by “look” rather than as they are to be read phonetically in Loglan, **lao** should probably be used. This construction was originally presented as a construction for the Linnaean names of biology; it is a valuable observation due to Steve Rice that it has a far more general usefulness. We abandon all other aspects of JCB’s discussion of Linnaean names as such: the details of scientific terminology are not part of the purview of the Loglan grammarian.

sao followed by alien text forms a predicate. This is a way to import a foreign word directly. **sue** followed by foreign text intended to transcribe or suggest a sound forms a predicate meaning “makes that sound”. **sue miao** is to meow.

Now we present our strong quotation proposal. The basic idea is that a series of blocks of alien text separated by whitespace is quoted by placing **lie** before the first block and **y** before each subsequent block. This is an entirely new proposal, though it turned out to be accidentally similar to the last proposal for the **lao** construction. The original strong quotation method is not PEG parsable (it is not even BNF parsable) and I think has other weaknesses. I have removed complexities of my original strong quotation proposal and made it parallel to **lao**.

The bit in Alice with the multifariously nested quotation marks must be translated into Loglan using this quotation style!

In the Phonetics Proposal, we have omitted the qualifiers **za** and **zi** for quotation of text versus speech.

We further note that the Phonetics Proposal allows alien text to be enclosed in double quotes, with whitespace allowed to be quoted (but pronounced , **y**, of course). The Phonetics Proposal *requires* that alien text following **hoi** or **hue** be quoted, to avert the possibility of non-name Loglan text with typos or grammatical errors being read as legal alien text inadvertently. The Phonetics Proposal allows multiple blocks of alien text to be used after **sao** or **sue**, with or without quotes, as in **sao “ice cream”**, pronounced (and also permitted to be written) as **sao ice y cream**, a predicate meaning (of course) “ice cream”.

7.10 Assorted grammatical particles, somewhat classified

Here is a list of terminators and boundary markers: **ci**, **cui**, **ga**, **ge**, **geu** (**cue**), **gi**, **go**, **gu**, **gui**, **guo**, **guu**, **gue**, and also the new **guoa**, **guoe**, **guoi**, **guoo**, **guou** (or alternatively **guoza**, **guozi**, **guozu**). There is a proposal of a new particle **gio**. Variants **guiza**, **guizi**, **guizu** are provided for the alternative parser.

New right closers **gueda guua**, **giuo**, **meu** have been added.

The particles **je** and **jue** mark tightly bound arguments (or modifiers, according to a proposal).

The JI words

jie: (restrictive set membership),

jae: (nonrestrictive set membership),

pe: (general possessive),

ji: (which/that (is) (identifying),

ja: (which/that (is) nonidentifying

nuji: (new 1/10/2016) converse of **ji**: can be used to set values of pronouns.

La Djan, **nuji Daicine** sets reference of the pronoun **Daicine** to John.

construct subordinate clauses from arguments, modifiers or predicates.

The JIO words **jio**, **jao** construct subordinate clauses from sentences (resp. identifying, nonidentifying) Variants of the JI and JIO words suffixed with **za**, **zi**, or **zu** are provided in the alternative parser, matched with alternative closers **guiza**, **guizi**, **guizu**. This allows efficient closure (with forethought) of nested subordinate clauses. This feature I will almost certainly add to the official parser.

The case tags, including the positional ones are listed:

beu: (patients/parts),

cau: (quantities/amounts/values),

dio: (destinations/receivers),

foa: (wholes/sets/collectives),
kao: (actors/agents/doers),
jui: (lessers),
neu: (conditions/circumstances/fields),
pou: (products/purposes),
goa: (greaterers),
sau: (sources/reasons/causes),
veu: (effects/states/effects/deeds/means/routes),
zua: (first argument),
zue: (second argument),
zui: (third argument),
zuo: (fourth argument),
zuu: (fifth argument),
lae: (lae X = what is referred to by X),
lue: (lue X = something which refers to X)

The operators of indirect reference **lae** and **lue** are a different sort of creature, which originally had the same grammar as case tags, but now have somewhat different behavior. The latter two operators can be iterated (and so can case tags, probably indicating that more than one applies to the same argument).

My opinion of the optional case tag system is that I would never have installed it myself, and it represents an extra layer of work for dictionary maintenance, but it is potentially usable and represents a large amount of work by our predecessors, so my intention is to leave it in place (and try to be good about assigning tags when I define predicates) and maybe maybe some day actually learn the case tags! The whole scheme is quite optional for speakers, though pressure to learn them would be imposed on a hypothetical Loglan community if many speakers actually used them.

The particle **me** constructs predicates from arguments. I believe the addition of **mea** was a mistake, as **me**, properly understood, already served its exact function. I'll write an essay on this eventually. A new closer **meu** has been provided to close **me** predicates (**gu** will still work).

The particles **nu**, **fu**, **ju** interchange the 2nd, 3rd, 4th argument of a predicate respectively with the first. These are called conversion operators.

The particles **nuo**, **fuo**, **juo** eliminate the 2nd, 3rd, 4th argument place of a predicate respectively, stipulating that it is occupied by the same object that occupies the first argument place (these are reflexives).

More conversion and reflexive words are formed by suffixing a quantifier. The only meaningful ones as far as I can see would be numerals larger than 4 and **ra**, which would choose the last argument place.

Yet more words of this class can be formed by concatenating conversion operators and reflexives; they simply compose, allowing complex reordering and identification of arguments.

Words which form abstraction predicates are the short-scope **poi**, **pui**, **zoi** and the long-scope **po**, **pu**, **zo**. In each set, the words form predicates for events, properties, and quantities respectively. Additional words **poia**, **poie**, **poii**, **poio**, **poiu**, **puia**, **puie**, **puii**, **puio**, **puiu**, **zoia**, **zoie**, **zoii**, **zoio**, **zoiu** are also long scope abstraction operators but with different closure words, **guoa**, **guoe**, **guoi**, **guoo**, **guou**, the final vowel indicating which closure word is to be used. There is an alternative version of this proposal adding abstraction words **poza**, **pozi**, **pozu**, **puza**, **puzi**, **puzu**, **zoza**, **zozu**, with closure words **guoza**, **guozi**, **guozu**; it is thought that **poia** in particular might be confused with **po ia** (though I disagree, insisting that a considerable pause is required in **po ia**) and certainly three additional sets are sufficient.

The uses of all these words will be revealed by the grammar.

7.11 Words which form free modifiers

The register markers indicate attitude toward the person addressed:

die: (dear),

fi: (comrade/brother/sister),

kae: (gentle as in gentle reader to an equal at a certain distance),

nue: (Mr Ms Mrs neutral and at a distance),

rie: (Sir, Madam, Sire, Honorable – to a superior)

They can be negated: there is no reason that we cannot address people nastily in a logical language.

The vocative marker is **hoi**. The inverse vocative marker (indicating the speaker or author) is **hue**.

The “right scare quote” is **jo**, which may be prefixed with a numeral. It indicates that previous text is not to be taken quite literally; the numeral would indicate how many words are in the scope of the **jo**. I notice that if a scare quote were to be applied to a quantity, it would have to be **nejo**. *soi crano*.

The paired words **kie** and **kiu** serve as spoken parentheses: include a well-formed Loglan utterance between them to form a free modifier. Actual parentheses can now be inserted after **kie** and before **kiu**.

Smilies can be spoken in Loglan: **soi X**, where X is a predicate, forms a free modifier inviting the auditor to imagine the speaker doing X. **soi crano** is literally :-) Loglan smilies are almost as old as the historical origin of smilies, I believe.

The freestanding attitudinal words of the original VV flavor, generally expressing emotions or attitudes, are

ua: (there! thats it! done! satisfaction),

ue: (indeed! oh! surprise),

ui : (fine! good! (pleasure)),

uo: (come now! look here! (annoyance)),

uu: (Alas! Sorry! sadness/sympathy/regret/not apology, that is sie),

oa: (moral obligation – it must be),

oe: (preferably),

oi : (permissibly, you may),

oo: (disapproving hmmm)[to be added!],

ou: (no matter (ethical indifference)),
ia : (yes), agreement),
ii : (maybe (tentative belief)),
io: (I expect that, apparently, moderate belief),
iu: (I have no idea!, ignorance, lack of belief or knowledge),
ea: (let's, I suggest...),
ee: (caution! careful! take care! [to be added]),
ei: (is it true that? forms yes/no questions),
eo: (please? will you? asks permission),
eu: (let us suppose that...(subjunctive)),
aa: (I see (what you mean)),
ae: (yes, I wish to (hope or weak intention)),
ai : (I intend to...Definitely...(strong intention)),
ao: (Yes, I want to, Ill try to...(moderate intention)),
au: (I dont care...indifference, absence of intention)

ie is not really an attitudinal, but an interrogative meaning “which”. (the words **aa**, **ee**, **oo** are not in the trial.85 list of UI words, though likely the preparser handles them fine in LIP; I have added them).

Additional words with the same grammar are

bea: (for example),
buo: (however, on the contrary, but),
cea: (in other words, namely),
cia : (similarly), coa (in short, briefly),
dou: (given, by hypothesis),

fae : (and vice versa),
fao : (finally, in conclusion),
feu : (in fact, actually),
gea: (again, I repeat),
kuo: (usually, customarily),
kuu: (generally),
rea : (clearly, obviously, of course),
nao: (now, next, new paragraph),
nie : (in detail, looking closely),
pae: (etc., and so forth) ,
piu : (in particular),
saa: (roughly, simplifying),
sui : (also, as well, furthermore),
taa : (in turn, sequence),
toe : (respectively),
voi : (skipping details),
zou: (by the way, incidentally),
ceu: (anyhow),
sii : (evidently)

These words are discourse operators, comments on the way we are speaking.

The word **cao** emphasizes the next word. The grammar will not show this, as it associates attitudinals with the previous word or construction!

Notice that one can use the phonetic stress markers to indicate stress in writing.⁴⁷

The word **seu** (a proposal) has a semantic effect, though it is grammatically an attitudinal. It marks an *answer*. This is **significantly useful**⁴⁸ for indicating that a predicate word given as an answer to a question is not intended as an imperative; it may have other uses.

Finally, we have words of social lubrication,

loi: (hello),

loa: (goodbye),

sia: (thank you),

siu: (you're welcome, dont mention it),

sie: (sorry (apology))

The word **sie** (to be distinguished from **uu**, sorry in the sense of regret but not apology) is new. Cyril and I believe it reasonable that **siu** be a polite answer to **sie** as well as **sia**.

There is a proposal, currently implemented, that these words also be vocative markers but not name markers, so that one can say **Loa Djan** as well as **Loa hoi Djan**.

The attitudinal, discourse and social words (class UI) can be negated by preceding them with **no** or following them with **noi** (the use of **noi** is a tiny proposal).⁴⁹

In addition, there are discursive operators firstly, secondly, lastly formed by suffixing quantity words with **fi**.

⁴⁷The word **kia** is listed as having the effect of cancelling the previous word. I do not at the moment intend to implement this: a grammatical implementation would involve recognizing certain **kia**-final constructions as freemods, and there would be decisions to make about what the units cancelled were to be (it appears to me for example that entire quoted constructions would be cancelled, and **liu kia** would be a quoted word, but there would be other restrictions, basically to do with the fact that a cancelled unit could occur only where a freemod could be expected).

⁴⁸serving to compensate for the fact that Loglan, unlike Lojban, does not have an explicit marker for the imperative; we further compensate for this by insisting that tense-marked gasents are observatives, not imperatives.

⁴⁹The ability to write "words" like **noia** (explicitly articulated as **no-ia**, and without a pause before the vowel initial **ia**) requires explicit overrides of the usual phonetic rules; I doubt that **liu noia** will parse, but this can be pronounced without pause.

7.12 Negation

The word **no** is the logical negation operator. Initial **no** in attitudinal forms, KOU words, and subordinate clauses (as well as occurrences internal to some compound structure words) must be excluded from this grammatical class. Pauses after **no** may be semantically significant, because they cause word breaks, and also because of the possible use of **no** to negate an entire utterance rather than its first argument (which usually does not affect meaning, though it affects the parse of a sentence).

7.13 Essays on word-making, and on what a word is exactly

7.13.1 Borrowing predicates

The responsibilities of a Loglan user borrowing a predicate from another language for use in Loglan are outlined.

One first roughly transcribes the word into Loglan phonetics. One replaces foreign sounds with Loglan sounds. It needs to be free of bad consonant combinations which Loglan doesn't support; this could be fixed by inserting vowels or sometimes by doubling continuants. Doubled non-continuants need to be undoubled.

It needs to have a left boundary of the right form. If it begins with a permissible initial consonant cluster, this is handled. Otherwise, we need to look after its initial $(C)V^n$ and see if a consonant cluster can be introduced. Appending **h** after a second single consonant as in **athomi** has been a frequent maneuver.

It needs to have a right boundary of the right form, which really amounts to being vowel-final: a vowel is added if necessary.

It needs to not be a complex. A vowel initial borrowing is of course never a complex. Doubling a continuant as in **hidrroterapi** can prevent a borrowed predicate from being a complex (and in this case also prevented the initial **hi** from falling off as it otherwise would, **dr** being an initial pair of consonants: this kind of gluing is another reason to introduce a syllabic consonant in a borrowing). Ensuring the presence of a sequence of three vowels would do this cheaply. A final sequence of three vowels will always work to prevent resolution into a complex, if the resulting stress is bearable to the hearer.

The non-Loglan speaker may need to adapt to the stress being in an

unexpected place. Part of the art of the borrower into Loglan is to try to make the word sound reasonably like the original while meeting the requirements for a Loglan borrowing.

It is also permissible for a borrowed word to take one of the shapes of five letter Loglan primitive predicates, CCVCV or CVCCV; it is not permissible for it to resolve into multiple djifoa. We do require that there are no Loglan predicates of the primitive shapes which differ only in their final vowel, unless they are actually variations of the same word, as in the animal or cultural “declensions”. This is vital because the identity of the final vowel is suppressed in forming the five letter djifoa. Such a borrowing becomes in effect a primitive and can form djifoa like any other primitive.

It is worth noting strategies used in salvaging VCCV initial borrowings: we have used doubling continuants, and also used initial **h**.

There are semantic requirements to making a predicate of either sort: one has to decide on an argument structure and, if one is really kind, decide on assignments of case tags to the arguments.

7.13.2 Making complex predicates

The responsibilities of the Loglan user in making complex predicates are outlined.

No new five letter “composite” atomic predicates are expected to be made: the esoteric process by which they were made does not need to be discussed. One might in theory make a five letter predicate as a borrowing as noted in the previous section. This should not happen often.

The maker of a complex should have a metaphor in mind. The components of the metaphor are then arranged in a suitable order (there might be some freedom in the order as well). One then chooses the right djifoa associated with the components. A borrowing has only one djifoa form, of course. Every primitive predicate (and any five-letter borrowing) has its five letter final form and its five letter medial forms with final **y**. Most of the primitive predicates have one or more three letter forms available, and the Loglan learner (and certainly the imaginary Loglan native speaker) should *know* the djifoa as part of the root vocabulary.

There are then certain restraints on the use of the three letter forms. One has to make sure that there is a CC junction. In fact, the only situation where there is a CC junction problem is if the first djifoa one has chosen is CVV, and followed by a CVx or CVCCx, and the problem is fixed by hyphenating

the first djifoa. An **r** or **n** hyphen is used by preference. A CVV with a **y** hyphen should be used only before a borrowing djifoa (where this is mandatory) or if the intention is that the CVV djifoa represent the cmapua of the same shape. The presence of a borrowing djifoa of course ensures the presence of a CC junction. We note with horror the possibility of complexes beginning **CVVy(C)VⁿCC**, which can happen if a CVV djifoa is followed by a borrowing djifoa. **CVCy(C)VⁿCC** is not much more appetizing.

This is a good moment to note that some **CVr** and (under a proposal of mine) all **CVh** djifoa are reserved to represent CV cmapua. The legacy vowel letterals may not be used as djifoa, but the new ones are eligible: **ziaytrena**, “A-train”.

A CVC djifoa in initial position will have to be followed by a **y** hyphen if an initial pair of consonants would otherwise be formed (or if it is followed by a borrowing djifoa). A **CyC** sequence does count as a CC pair, as in **mekykiu**.

The CVV djifoa with repeated vowels that force a stress cannot occur except in final position or in penultimate position, followed by a monosyllable.

Where a CVV which is an optional monosyllable ends a complex, it may be the case that two possible patterns of syllabification and stress are possible for the complex.

The remaining obligations are aesthetic: make a reasonably short, pronounceable and even pretty word. Aesthetics may vary: this writer *likes* the word **likcke**.

There are semantic requirements to making a predicate of either sort: one has to decide on an argument structure and, if one is really kind, decide on assignments of case tags to the arguments.

7.13.3 Name words

The name words consist of the name words in the phonetic sense of the first section and the acronyms. One is required to pause after an acronym used as a name, and one is permitted to omit the explicit comma in writing under exactly the same conditions as after an ordinary name word. It is worth noting that a pause is also required after an acronym when it is used as a dimension in a quantity.

Contrary to statements in L1, we maintain that a Loglan name word should always be written as it is to be pronounced. Names written to look visually like their forms in other languages should be treated as alien text

and turned into grammatical proper names with **lao**. Thus, **la Ainctain** is the native version of Einstein's name, but we can of course also write **lao Einstein**. The first must usually be followed by an explicit pause, while the latter may be followed by an innocent space – which will also be a pause, as stated in the rules for alien text. **la Einstein** is a legal Loglan name, but would be pronounced quite oddly.

Creating Loglan proper names is generally a process of transcription of a name from some other language. Transcribed names must resolve into Loglan syllables. One should notice that we do not allow double consonants except for syllabic consonants, and that syllabic consonants *must* be doubled. Further, a name may not contain more than two successive non-syllabic consonants at the end, though this may be fixed by doubling a continuant, as in **la Marrks**.

It is conventional but not required to convert a vowel-final name from foreign sources to a Loglan name by adding **s**; there is nothing wrong per se with using another consonant, particularly if there is an etymological reason to do so. Loglan names can be made from predicates by omitting final vowels or (conventionally) by adding **n**. Another idea which I have encountered recently is to make an illegal complex ending with a CVC djifoa and use this as a name, which strikes me as a lovely idea. sub

7.13.4 Essay: what is a word?

Cyril Slobin asks me, what is a Loglan word? How does the hearer resolve a stream of Loglan sounds or letters into words?

JCB's answer in NB3 was that a word is a sequence of phonemes in the midst of which one cannot pause.

This is not perfect, but it is a good approximation. JCB himself defined an exception: one can pause in the middle of a predicate word after a borrowing affix! Cyril himself proposed an exception for long NI words (numerals): pauses, even comma marked ones, between NI1 units do not affect semantics.

Name words are reasonably easy to recognize phonetically (pause free sequences of phonemes, usually marked by an initial name marker word, ending unmistakably with a pause after a consonant). They certainly meet JCB's criterion; pausing in the middle of a name breaks it. We also view the name marker word as a word.

Predicate words are fairly easily recognized phonetically, starting with a characteristic CVⁿCC phonetic configuration and ending with a penulti-

mate stress. They do not break into separate breathgroups except for JCB's exception of allowing pauses after borrowing affixes. Of course, one might heretically view a predicate with a borrowing affix as a kind of phrase, but I think it is really still a word. Similar remarks apply to John Cowan's **zao** construction, another way to build a complex predicate which actually allows internal pauses.

More headaches about what a word is arise with *cmapua*. The Lojbanists have apparently arranged things so that one can pause anywhere in a stream of *cmapua* syllables without affecting meaning, so that the unit "words" are just unit *cmapua*. This is not true in TLI Loglan. JCB certainly thought that compound *cmapua* words existed in the language. I regard the members of certain large *cmapua* classes as words, and in most cases I have enforced the rule that one cannot pause inside them.

I make a list of classes that are inhabited by multisyllable *cmapua* words.

TAIO: this class includes multisyllable names of letters that do not fall apart.

A: This class includes quite complex logical connectives. One cannot pause inside such a word. **noapacenoia** is a long example.⁵⁰

ACI, AGE, CA: relatives of A, similarly large classes of words in which breaks are not permitted (except as in the footnote under the previous entry).

I, ICA, ICI, IGE: again phonetically and to some extent semantically similar to A.

KA, KI: These classes include compound words, all fairly short, since we exclude PA-suffixing of such words.

NI: This is a large class of quantifier words, and I really do think that they are words, except that I allow pauses between NI1 numeral units. This does not mean that one can freely pause anywhere in a NI word; at many junctures one cannot, and certain constructions unequivocally close such a word. The related class of numerical predicates does not allow internal pauses.

⁵⁰One now **can** pause inside such a word, next to a CA0 connective, but it is still clearly a word.

Acronym: Acronyms are words (or in the case of dimensions, parts of NI words). One cannot pause in the middle of an acronym, and its boundaries are clearly marked (by **mue** or a name marker on the left and a pause on the right).⁵¹

DA: Suffixed pronouns are multisyllable cmapua.

PA: Pause free strings of PA cores are words. Series of PA cores linked with CA cores are now viewed as phrases.

LE: Compound articles such as **lemi**, **levi** were words under LIP (LIP allowed spaces in them but not commas) and under previous versions of my parser, but I have (at least experimentally) modified class **descriptn** so that things like **le mi hasfa**, **le va hasfa**, **le mi na hasfa** are actually read word by word. The sentence **le mi hasfa** is now an instance of the same grammatical construction as **le la Djan, hasfa**, which was not true in trial.85, though every learner may have thought so.

JI: I allow **nuji**.

NU: Suffixed conversion operators such as **nufe**.

UI: NI F i discursives are words. Negative attitudinals such as **noia** might be viewed as words: **liu noia** (with following pause) should parse under the Phonetics Proposal.

BI: I allow forms like **nubi**, which are treated as words (**La Djan, nubi da** is parseable, but **La Djan, nu bi da** is not: **nubi** is semantically but not grammatically parallel to **nu blanu**.)

Other cmapua classes define words inhabited by one-unit cmapua (not necessarily one syllable, as some unit cmapua are disyllables).

This is actually not a terribly long list. Familiarity with the phonetics of names and predicates (admittedly quite nasty in its finer technical details, but usually quite manageable in normal situations) and the grammar of a few word classes will allow you to recognize the Loglan word.

⁵¹added the ability to insert **, mue** into an acronym, so yes, one can pause, but it still looks like a word class.

It is important to notice, though, that while the recognition of name and predicate words is a matter of phonetics, recognition of the *cmapua* words is a matter of understanding the grammar. They do have a common phonetic property (most of them), in not admitting internal pauses, but they are not resolved using phonetic criteria.

8 Grammatical Constructions

This part of the document is fairly closely based on the last official Loglan BNF grammar which underlies LIP. There are changes, major and minor, which I will mention as we encounter them. I started trying to write it in the order presented in the grammar, and this is simply wrong. I have taken an alternative approach working through *trial.85* backwards, more or less, hoping that this will give a more top down view.

The original *trial.85* grammar appears as an appendix. More useful is the other appendix containing the complete PEG grammar with extensive comments.

Some grammar classes are given English names: some classes are referred to by their names in the PEG formal grammar, which are usually derived from names in the *trial.85* grammar.

8.0.5 Note on Right Closers

The original class *gap* (manifested as **gu** possibly flanked on one or both sides by commas) has been subdivided; I refer to things of the other classes *guua*, *guea*, *giuo*, *meu* as “gaps”, but in fact they are now separate classes in the grammar. Each of them can manifest itself as the word naming it or as **gu**. This is all perfectly analogous to the earlier diversification of **gu** into **gue**, **gui**, **guo**, **guu**, but these sorts of closures are less common.

8.1 Sentences and Utterances

This corresponds to the last part of the *trial.85* or PEG appendix document, which discusses sentences and utterances.

8.1.1 The most basic sentences

The most typical Loglan sentence consists of terms (a list of arguments and/or modifiers including at least one argument and no more than one un-case-tagged argument, of class **subject**), followed optionally by **gio** followed by a sequence of terms, followed by a predicate: it is important to note that the predicate may include a final list of arguments and modifiers, so this is the form of an SVO sentence (an SOV(O) sentence if the **gio** clause is present). The set of terms is usually a single argument (the subject of the sentence) but it may be accompanied by modifiers, and by other arguments if they are case-tagged, or if they are marked with **gio**. The initial list of terms must include at least one argument, or the sentence will be understood as an imperative. Giving an example with more than one argument before the predicate, **Da gio de blanu** or **Da zue de blanu** has the same meaning as **da blanu de**, “X is bluer than Y” (an SOV sentence can thus also be constructed using this rule).

Another alternative is the subject-deferred sentence (class **gasent**), a VO(S) construction, which consists of an optional initial **no** of negation (which should not be followed by a pause if intended to be part of the subject-deferred sentence, though it may be followed by other free modifiers) followed by **ga** or a tense marker, followed by a bare predicate without tense marker, followed optionally by a suffix consisting of **ga** followed by more terms, which we require to be of class **subject** (contain either at least one argument but at most one un-case-tagged argument) or to contain all the terms in the sentence; in the all terms case the first non-case-tagged argument may optionally be separated by **gio** from all subsequent non-case-tagged arguments (the subject-deferred sentence may also fail to have a subject at all). The bare predicate may include final arguments: the argument(s) after the **ga** is initial. **Na blanu de ga da** means the same as **Da na blanu de**, “X is now bluer than Y”. An essay on the modifications we have made in the **gasent** class would be useful.

It is possible for the final component **ga** + terms to be omitted, giving a subject-free sentence like **Ga blanu** or **Na blanu**: (It is) blue. “It’s raining” can be said in this way: **Na crina!** where just **Crina!** would be an imperative, meaning something like “be rained on!”. When the final **ga**+ terms is omitted, a missing **ga ba** is understood.

As explained below (under logically connected sentences (class **sentence**)), an unmarked predicate (which may include following terms) possibly pre-

ceded by one or more terms not including any arguments is an imperative sentence: **Donsu ta mi**; “give that to me”; **Na la Ven, dons u ta mi**: “At nine, give that to me”. Marking the predicate with a tense makes it a declarative sentence with an indefinite subject: **Fazi dons u ta mi**, “Someone will shortly give that to me”. This is to be understood as **Fazi dons u ta mi ga ba**. The sentence **Fazi dons u ta mi ga la Djan**: “John will shortly give this to me”.

It is also possible for one or more modifiers to appear before a subject-deferred sentence.⁵² In the alternative parser, modifiers before the “main verb” in “verb-initial” sentences (imperatives and gasents) are not permitted: the reason for this is that sentences which parse in unintended ways due to failure to properly close such a modifier, allowing it to eat the subject, give such forms under the “official” parser. In particular, this happens in the Leith novella (where pause/**gu** equivalence originally managed closure of such initial modifiers); I only managed to efficiently detect these misparses by excluding these forms in a test parser. Such modifiers can be added as object of class **headterms** using **gi**: see below; no means of expression are lost.

All three of these forms are options in the grammar rule “statement”: it looks for a subject-deferred sentence first, then a subject-deferred sentence with initial modifiers, and only then for a sensible SVO sentence (or S(O)VO sentence); under the alternative parser, only the last two forms are considered.

Underlying this is the form Px_1, x_2, \dots, x_n of an atomic sentence in logic, with a predicate (verb) followed by a list of objects. To accommodate the most typical word order in natural languages, this was changed to x_1Px_2, \dots, x_n . In what I regard as a much more dubious decision, this was grouped $x_1[Px_2, \dots, x_n]$, with following arguments incorporated into the predicate. Further modifications are that the x_1 may be replaced by a series of arguments and that the list of arguments may be padded with modifiers (tense/location/relative clauses) which may appear in any position, before, between or after the arguments.

The maneuvers to move an initial argument or segment of arguments to the end are part of a scheme for achieving all the possible orders of subject

⁵²Notice the reordering of what is going on by the hearer when **di fa dons u de...** (which sounds as if Z will give Y to ...) is completed **di fa dons u de ga da**: this is why we forbid such sentences [LIP allows them, but JCB clearly states in NB3 that the initial terms in the form **terms gasent** were intended to be modifiers].

verb and object(s). This scheme is not completed yet: there is a further device for fronting a final sequence of arguments (allowing arguments which are final to a predicate to appear first) which does not appear quite yet because it distributes over logically connected sentences, as the subject or initial arguments moved to the end by **ga** do not.

8.1.2 Logically connected basic sentences (and final arguments moved to the front)

The next group of sentence forms to be introduced are logically connected forms. A forethought connected sentence (class **keksent**) – I will defer describing for a moment.

A logical unit sentence (class **sen1**) is either (1) a statement, (2) a solitary predicate without a tense marker possibly preceded by modifiers (an imperative sentence, as noted above; leading modifiers are not permitted under the alternative parser), or (3) a forethought connected sentence. A **sen1** may further be negated using class **neghead** (negation with sentence-long scope, possibly repeatedly): a **neghead** is either **no** followed by optional freemods followed by **gu** or **no** not initial in a **predunit2** followed directly by an explicit comma pause.⁵³

A sentence is a logical unit sentence followed by zero or more logical connectives of class ICA each followed by a logical unit sentence: **Da blanu, ica de blanu, ice kukra!** “X is blue, or Y is blue, and run!” We must note semantically that these group to the left: **(Da blanu, ica de blanu), ice kukra.** This is important when different logical connectives are used together. ***Da blanu, de blanu, ice kukra!** is *not* correct; in a sentence in Loglan there will be connectives between each pair of adjacent logical unit sentences (the same holds for any chain of items linked by afterthought logical connectives). A sentence which is not a logical unit sentence may be called a logically connected sentence.

A sentence with fronted arguments (class **uttAx**) is a sequence of terms followed by **gi** followed by a sentence, possibly closed with a **gu**. The terms are final to the predicates involved and distribute over all the component logical unit sentences (if indeed the sentence is logically connected (which is why an ability to close the sequence is needed⁵⁴)). Fronted terms can also

⁵³**No, kukra prano** means “Run slowly”, not “Don’t run fast!”, which is expressed by **No gu kukra prano.**

⁵⁴but closing such a sentence is quite hard: one probably needs to close a final sentence

be connected with **goi**: these are quantifier prefixes and require separate extended semantic discussion!

De gi da blanu Simple OSV order without logical connection issues, “X is bluer than Y”.

Di gi mi cluva, e tu donsu de means “I love Z and you gave Y to Z”. If I want to follow this with an afterthought connective and a sentence without Z as a final argument, I need the pause.

Ra ba goi, ba cluva mi “Everyone loves me” is an example of quantifier prefixes. Of course one can say **Raba cluva mi** in this very simple case.

A semantic point (and a proposal): our Sources dictate that the last argument before **gi** must be the actual last argument of the predicate, so that we can skip middle arguments. I regard this as a bad idea; one reason for this is that there are predicates which have many arguments, the last of which may be very obscure to the speaker and/or listener. Instead, I propose that the default position for the final arguments be that the first one in the block immediately follows the last argument appearing in the sentences following (as in my example); if the first argument in the **gi** block (after any arguments with semantic case tags) is marked with a positional case tag, it will take that position in all following sentences and following arguments will be in the positions following the explicitly marked one. This will allow the desired argument-skipping effect. There is a more thorough proposal along these lines in the list of proposals below in the Report.

A forethought connected sentence (class **keksent**) is an optional negative **no**, followed by a word like **ke**, followed by a sentence or sentence with fronted arguments, followed by **ki** followed, surprisingly, by an instance of the very general “sentence fragment” (**uttA1**) class of utterances described below, which does include the various sorts of sentences given so far. **Ke mi vizka tu ki mi cluva tu** “I see you and I love you”. These are forethought logical connectives – one needs to plan these in advance!

8.1.3 Free Modifiers and Utterances

I will now move up a level to general forms of utterances.

The first topic is *free modifiers* (freemods). These are a rather miscellaneous collection of constructions which have the feature that they can be

with **guu** before closing the sentence with fronted arguments with **gu**. This kind of caution applies to all closures which are still effected with **gu**: it can be hard to tell what the **gu** actually closes.

inserted into a Loglan utterance almost anywhere. In almost all locations in between elements of a Loglan rule, a free modifier may appear. A position before one of the closing forms (**gu** and the special terminating forms **gue**, **gui**, **guo**, **guu**, **geu**) is not regarded as a medial position where a freemod can be expected to be allowed. A closing form may itself always be followed by a free modifier (modifying the construction which it closes as a whole). A free modifier is generally attached to what it follows, so free modifiers almost never appear at the beginning of grammatical rules.

The varieties of free modifier follow:

negative attitudinals: phrases like **no ui** and **no sia** fit in here. It is important to notice that this *no* has no logical negative effect. **No ui mi hijra**, “Unhappily, I am here”. A pause here breaks this effect. **No, ui mi hijra** seems to mean “It is not the case that I am happily here”. A phrase of this class like **noiu** is detected as **no iu**, and not read as **noi-u**: this took special effort.

attitudinals: Words like **ui**, or importantly the word **ei** that turns a sentence into a yes/no question.

“**smilies**” **soi** followed by a predicate of the descpred class, suggests an action or attribute of the speaker. **soi crano** is a quite literal translation of :-)

register markers: indications of attitude toward the one addressed such as **die**, dear.

negative register markers: indications of negative attitude toward the one addressed, such as **no die**.

parenthesized utterance: **kie** followed by any complete Loglan utterance followed by **kiu**. A side remark. The parenthesized utterance can optionally be set off from the **kie**, **kiu** by comma pauses. The Phonetics Proposal allows **kie/kiu** to be supplemented (not replaced) with actual parentheses, in the format **kie(..)kiu**, with a legal Loglan utterance between the parentheses.

inverse vocative: **hue** followed by a name (including foreign names), a statement, a descriptive predicate optionally followed by a gap (**giuo**) (with optional following name: **hue bilti** works, and so does **hue bilti**,

Djin) or an argument list, indicating who is speaking. We changed the argument list option from the **terms** class to the class **termset1**, because otherwise it is very hard to prevent an inverse vocative in initial position from consuming an entire following sentence. Reading Leith has convinced me that the statement form is useful; at any rate we have a lot of text in which both **terms(et)** and statement forms are used. In setting up old text to parse, it will very often be necessary to close inverse vocatives with **guu**. It may also be necessary to insert a pause after the **hue** if what follows is not a name. If what follows is alien text, it must be enclosed in double quotes.

vocative: This is a separate grammatical form introduced later. **Hoi Djan** is an example. A reform of the language forbids the simple use of a name by itself as a vocative, even when preceded by a space. For reasons, see discussion of the “false name marker problem”.

pause, ellipsis, hyphen A comma pause, not significant. Ellipsis . . . and hyphen -- can also appear as freemods.

scare quote: an optional numeral followed by **jo** (**jo** being equivalent to **nejo**) signals that the preceding word(s) (the number of words suggested by the numeral) are not to be taken literally. **Ai tu fremi jo mi: Certainly you are my “friend”**. How to ooze insincerity. An explicit pause will allow a numeral to be put in scare quotes. **Ti ne, jo zavlo** “This is “one” bad thing”.

Now we commence the treatment of utterance forms.

An “answer fragment” (class **uttA** is a connective or a number. These can only occur as utterances as answers to questions.

A “sentence fragment” (class **uttA1**), a very general class of utterances already mentioned in the connection with the forethought connected sentence class above, may be a logical unit sentence, an sentence with fronted arguments, **no** by itself, a tightly bound argument list beginning with **je** or **jue**, an argument modifier (subordinate clause, class **argmod**), or a list of modifiers followed by a forethought connected sentence. This may be terminated with a period or other final punctuation.

I think the sentence fragment class, which includes a lot of utterance fragments, often serves to provide a form for answers to questions. But it does include the logical unit sentence and “sentence with fronted arguments”

classes of complete sentences, so it can include quite general utterances. And it enters into the makeup of the forethought connected sentence class. Some of the forms it permits for forethought connected sentences are very weird. **Ibuo nukouki mi no nu fatru ki lo aurmo** “But I don’t care, because gold!”, a translation of part of a gaming joke my son likes.

I have proposed an attitudinal word which marks *answers*. This would remove the potential semantic ambiguity between predicates given as answers to questions with **he** and imperative sentences. I have suggested the new answer attitudinal **seu** for this purpose.

There are now various layers of utterance up to the full Loglan **utterance** class. These classes are named as in the PEG grammar.

An uttC is a sentence fragment (optionally) preceded by one or more **no**’s set off from the following utterance by **gu** or a pause (this is the only surviving bit of pause/GU equivalence in the language, but now further restricted not to begin a **predunit2**, so no pause of this kind will actually have any semantic effect⁵⁵); an initial **no** will otherwise be absorbed into some shorter structure at the beginning of the utterance.

An uttD is either a sentence optionally followed by terminal punctuation and not followed by ICI or ICA, or one or more uttC’s linked by the afterthought ICI connectives.

An uttE is one or more uttD’s linked by the usual ICA connectives. Note that a sentence (sen1’s linked by ICA connectives) will be parsed as a single uttD (and as a sentence) rather than as a string of uttD’s. This resolves a dissatisfaction of ours with the parser without, we think, significantly changing any parses.

An uttF is one of more uttE’s linked by I class connectives. Notice that causal connectives like **ikou** are of the I class not the ICA class and so will bind less tightly than the ICA logical connectives.

The Loglan utterance can be one of the following, with the further restriction that an utterance cannot begin with the little word **ge**:

free modifier initial utterance: A non-pause free modifier followed by another utterance. This is basically the only free-modifier-initial construction in the language.

free modifier alone: A free modifier alone, with the same restrictions. It

⁵⁵The worst the pause can do with this restriction is move a negation from the first argument of a sentence to the entire sentence, which has no effect on meaning.

checks for the following utterance first. A period or other terminal punctuation will close this.

ige construction: An uttF followed by an IGE afterthought connective, which links the first utterance to the entirety of what follows (no left grouping). This rule is the reason that an utterance cannot begin with **ge**, to avert ambiguity.

i (or another I word) followed by a free modifier: just what it says.
A period will close it.

an uttF: Just what it says.

i (or another I word) followed by an uttF: Just what it says.

Further, if any well-formed utterance is followed by a well-formed utterance beginning with a word of class I, it expands to include the following utterance. The same holds for **#** followed by an utterance, so far as the parser is concerned: the reader should regard this as a complete change of voice (even if the same speaker has paused and resumed). The parser treats **#** followed by an utterance as if it were end of text, which may not be quite true of **i** followed by an utterance.

8.2 A semantic note: scopes of quantifiers

These are experimental specifications of mine, intended to be used in software for evaluation of logical arguments in Loglan which does not exist yet.

The scope of the quantifier binding a particular occurrence of a bound variable **ba**, **be**, **bo**, **bu** (or indexed forms of these) is the smallest sentence (class **sentence**) or sentence with fronted arguments (class **uttAx**) containing all occurrences of the given variable which lie in a common sentence or sentence with fronted arguments with it. It might be better to require smallest logical unit sentence or sentence with fronted arguments, but I have specific examples in mind which motivate the choice of class **sentence**. For this it is important to remember that logical operators that build class **sentence** are left-grouping.

An indefinite phrase like **ra mrenu** or **su mrenu** is also a quantifier in the sense of this note, with a restricted domain: **ra mrenu** for example can be thought of as abbreviating **ra ba ji mrenu**.

Where two quantifiers have the same scope, the one which is outermost is the one which occurs first, with the proviso that where a quantified variable has a subordinate clause attached to it, the position of the variable used to determine order is its position in the subordinate clause, not its position at the head of the subordinate clause.

8.3 Predicates

At least initially, we will discuss construction of predicates from the bottom up.

8.3.1 The basic building blocks of predicates: predunit classes

The class of atomic predicate units (`predunit1`) consists of predicates which are in a certain sense atomic (basic building blocks).

1. Predicates of the form **sue eep** (onomatopoeia) or **sao antidisestablishmentarianism** (foreign predicates). Details of these forms are discussed in the lexicography section. They are semantically quite different from each other but share the trait of being formed using a little word followed by alien text (and by a mandatory phonetic pause, though it may be written as whitespace).
2. a conversion (or reflexive) operator followed by **ge** followed by a simple description predicate (a flavor of descriptive predicate described below) closed off optionally by **geu** (or the archaic **cue**). This is a sort of parenthesis operation (with conversion) allowing a more complex predicate to be treated as a basic predicate building block.
3. a conversion (or reflexive) operator (such as **nu**) followed by a predicate word (this is just a basic predicate with arguments reordered).
4. The parenthesis form without a conversion operator: **ge** followed by a simple description predicate (**despredE**) closed off optionally by **geu** or **cue**.
5. An abstraction forming word like **po** followed by a sentence with fronted arguments closed off optionally by **GUO** (**guo** or **gu**).
6. An abstraction forming word like **po** followed by a “sentence” closed off optionally by **GUO** (**guo** or **gu**). This form and the previous one are part of a repair to the language which I made recently: in `trial.85` the uses of these kinds of predicates are incredibly (and unnecessarily) constrained.
7. As in the previous two cases, but using **POA**, **POE**, **POI**, **POO**, **POU** and closing with respectively **GUOA**, **GUOE**, **GUOI**, **GUOO**, **GUOU**.

This is by analogy with the similar construction of abstract descriptions, which I think is **needed**; this option may or may not see use.

8. The predicativizing little word **me** followed by an argument closed off optionally by **gu** (optionally flanked by explicit pauses on one or both sides) [i.e., a gap (**meu**)].
9. a predicate word (see above in the lexicography section).

Any of the above forms of atomic predicate unit may include free modifiers in all medial positions and terminally.

A predunit2 is formed by (optionally) affixing one or more **no**'s (possibly followed by free modifiers) to the front of an atomic predicate unit.

A predunit3 is a predunit2 followed optionally by a list of arguments of class linkargs (tightly bound with **je** or **jue** as we will describe).

A predicate unit (**predunit**) is either a predunit3 or a predunit3 preceded by a short-scope PO operator such as **poi**. The predicate unit is an important level to pause at, as this is exactly the sort of predicate which can appear as a unit in a serial name like **la Djan ci Blanu** (“John the Blue”). More to the point, we can have **la Djan ci Blanu Je Tu**: John the Bluer-than-You, or **la Djan ci ge Cmalu Hasfa** (John the Small House), but not ***la Djan ci Blanu Tu** or ***la Djan ci Cmalu Hasfa**. It is important to notice that modification of one predicate by another can occur in a predicate unit only inside a **ge...(geu)** block.

A forethought connected predicate (**kekpredunit**) is a (possibly multiply negated) forethought connected pair of predicates (in the most general sense to be seen at the end of this section). The form is one or more **no**'s of negation (the negations(s) are optional) followed by a word of class KA followed by a general predicate followed by a word of class KI followed by a general predicate. At every juncture except after the general predicates a free modifier may be inserted. A simple example: **no ke blanu ki cmalo**, “not both blue and small”. A **kekpredunit** can optionally be closed with **guu**.

8.3.2 Description predicates

These are predicates intended to appear in descriptions (as components of “noun phrases”) rather than those which appear as “verbs”. We will see below the contexts in which they are used.

A *despredA* is the most tightly bound metaphor construction: it is a sequence of predicate units and forethought connected predicates separated by **ci**. **cmalo ci hasfa**, “small house”. **cmalo ci nirda ci hasfa**, “(small bird) house”. All metaphor constructions group to the left. To say “small birdhouse”, **cmalo ci ge nirda ci hasfa**. Note of course that all these phrases make perfect sense with all instances of **ci** omitted: these phrases would only normally be used embedded in a more complex construction. Free modifiers can appear before and after **ci**.

A *despredB* is either a *despredA* or the little word **cui** followed by a *despredC* followed by a CA word followed by a *despredB*. An example is **cui cmalo bekti ca groda** “a small thing or a biggie”. Free modifiers are allowed next to the CUI and CA in medial positions.

A *despredC* is a chain of *despredB*'s. This is a special version of the basic metaphor construction, as used between a CUI and a CA in the previous rule. Free modifiers are allowed in medial and final position.

A *despredD* is a chain of *despredB*'s linked by CA words (free modifiers allowed before and after the CA words). This is top level logical connection with CA words. These are grouped to the left.

A simple description predicate (*despredE*) is a chain of *despredD*'s with free modifiers allowed in medial and final position; this is the top level metaphor construction, grouped to the left. **cmalo nirda hasfa** is a house for small birds and **cmalo ge nirda hasfa** is a small birdhouse. The simple description predicate class is of special note as being the sort of predicate which can be enclosed in **ge...(geu)** to form an atomic predicate unit.

A description predicate (*descpred*), the top level class of predicates used in descriptions, is either a simple description predicate or a simple description predicate followed by **go** followed by a description predicate, where the order of modification is reversed: **nirda hasfa go cmalo** is a small birdhouse. Free modifiers are allowed before or after the **go**.

Detailed examples of metaphor constructions using all the indicated features are owed (and can be found in L1 and NB3).

8.3.3 Sentence predicates, first pass

A simple sentence predicate is a *despredE* or a *despredE* followed by **go** followed by a bare predicate (see below). This is more general than a description predicate because a bare predicate may have a list of arguments

attached (loosely rather than with JE/JUE).⁵⁶

8.3.4 Sentence predicates, second pass

Here we introduce a black box: a termset is a (quite complex as we will see) argument list which can be attached to a predicate. In **cluva la Djan**, “love John”, **la Djan** is a termset. These can be much more complicated, but their internal details do not enter into the grammar of predicates (though they complicate the semantics!) A termset is the word **guu**, or an argument, or list of arguments, or a structure built by logical connection of simpler termsets.

A bare predicate (**barepred**) is a simple sentence predicate followed by an optional termset (which may have a free modifier before it) and optionally by **guu** if preceded by a termset or followed by a term. This is the class which can appear after **go** in a simple sentence predicate.

A tensed predicate (**markpred**) is a PA word or **ga** followed by a bare predicate. This is a predicate with a tense marker, with the option of the null tense marker **ga**.

A **backpred1** consists of (optionally) one or more **no**'s of negation (with following optional free modifiers) followed by a (required) bare or marked predicate. The caveat applies to each **no** of negation that it does not start a **predunit2**: in **no blanu hasfa** the initial **no** is captured in the **predunit2** component **no blanu**. **no blanu hasfa** means “is a non-blue house”. **no ga blanu hasfa**, where the negative is not captured, means “is not a blue house”. There are cases where a bare predicate can be negated to form a **backpred1**, as in **no poi blanu**.

A **backpred** is either a **backpred1** or a structure built by linking **backpred1**'s with ACI afterthought logical connectives and optionally adding a termset closable with optional **guu** (shared by all the logically linked **backpred1**'s), or a structure built by linking general **backpreds** with ACI connectives and optionally adding a termset (shared by all the logically linked **backpreds**) optionally closable with **guu**. It is important to notice that before

⁵⁶This is now the only difference between basic description predicates and basic sentence predicates: there used to be a systematic difference due to a requirement that the head of a sentence predicate metaphor could not be forethought connected; this rule was lifted by a reform in the 1990's but with a restriction that preserved the need for four classes of sentence predicates analogous to four of the classes of description predicates given above, which now have been eliminated.

a shared termset can be added, the termset of the last item in the backpred must be closed with **guu**, even if it is null (this is why **backpred** without a termset can be closed with **guu** if there is a following term). The same remark applies to shared termsets in the next class **predicate2**.

A **predicate2** is either a **backpred** or a structure built by linking **backpreds** with A afterthought logical connectives and adding a termset (shared by all the logically linked **backpreds**), optionally closable with **guu**, or a structure built by linking general **predicate2**'s with A afterthought connectives and (optionally) adding a termset (shared by all the logically linked **predicate2**'s), optionally closable with **guu**. A **predicate2** cannot begin with **ge**, nor can a **backpred** following an A connective begin with **ge** (to defend the AGE connectives); nor is there any reason that it should.

Both in **backpred** and in **predicate2**, where termsets optionally closed with **guu** (or **gu**) appear, the option also exists of closing with **guu** with no preceding termset, if it is followed by a term, as was the case in **backpred**.

Both ACI and A logical connectives group to the left.

This approach has the same practical effect as the **trial.85** approach in most cases, but is quite different in detail (and in background theory). First of all, the ACI connectives are fully privileged logical connectives binding more tightly than the A connectives. Secondly (and perhaps most strikingly) no distinctions are drawn between marked and unmarked classes; these distinctions seem to be unnecessary even in **trial.85**. Thirdly, the handling of logically shared final termsets is rather different. The **trial.85** solution is quite lovely, but extremely hard to implement in a PEG. It seems most unlikely that a layering of logically shared final segments of termsets which could not be handled by the rule we give here would ever appear in speech.

Extensive examples will be needed. It should be noted that there are no examples of constructions with complex logically shared final termsets in the NB3 corpus.⁵⁷

A **predicate1** is either a **predicate2** or a **predicate2** followed by an AGE connective followed by a **predicate1**. Notice that there is no provision for adding termsets shared via AGE connectives, and also that these highest level afterthought connectives group to the right.

A **predicate** is a **predicate1** or an **identpred** (one of the identity predicates listed above in the lexicography section; note that we allow these last to be prefixed with **nu**). Note that we **can** logically link **identpreds** to other

⁵⁷There are such examples in L1

predicates using forethought connectives, but we are certainly strongly discouraged from doing so.

8.4 Clauses, arguments and term lists

In this term we do the constructions which culminate in terms (arguments and modifiers) and term lists.

8.4.1 Serial names and the false name marker problem

A name word refers to either a consonant final name word or an acronymic name. We already know that such words must be followed by pauses.

The words **la**, **hoi**, **ci**, **hue**, **liu**, **gao**, plus the social lubricant words **sia**, **sie**, **siu**, **loa**, **loi**, are the “name markers”. A name word must be preceded either by a pause or by a name marker. Under the Phonetics Proposal, the status of the social lubricant words as name markers is under review (not implemented at the moment, but still worth mentioning as possible).

An occurrence of a string identical to a name marker word in a name is called a “false name marker” if what follows the apparent name marker word is itself a well formed name word.

Complex name constructions are supported (serial names). A serial name begins with a name word, followed by a series of items of the following sorts (each of which will begin with at least a space):

1. **ci** (possibly preceded by a free modifier) followed optionally by a pause followed by a name word, as in **Pierr ci Laplas** (it is generally better not to pause after a name marker which is actually followed by a name word).
2. **ci** (possibly preceded by a free modifier) followed optionally by a pause followed by a predicate unit; this may not be followed immediately by an item of the next type (an unmarked name word), as in **Djan ci Blanu**
3. an unmarked non-acronymic name word containing no false name markers, as in **Djan Braon** (never preceded in a serial name by a predicate unit).

It should be recalled that a name word is always followed by an explicit pause, except when it is followed by end of text, terminal punctuation, a space followed by **ci** or a space followed by another name word (commas are permitted but not required in the latter two contexts). The last special cases are motivated as we can now see by the structure of serial names. In the special cases, there is a pause at the end of the name word (and at least whitespace if not at end of text) even though it is not expressed by a comma.

We add some remarks about the general problem of false name markers. The issue is whether we can tell where a name word starts. The end of a name word is always detectable as an explicit pause (or terminal punctuation, or a space before **ci** or another name word). The problem is ensuring that we can recognize the beginning of a name word. The key to our solution is that the parser will only attempt to read a name word starting in very precisely defined positions: immediately after an explicit pause, or a name marker word, or another name word already read. Moreover, name words only appear in quite specific grammatical contexts (this was enforced by eliminating unmarked vocatives (addressing John as just **Djan** rather than **hoi Djan**) which made it possible for name words to be free modifiers capable of appearing almost anywhere), and by making some further technical modifications in how name words can appear in other grammatical constructions.

Where any name marker appears followed (with an optional intervening comma pause) by something which can be read as a serial name, this is the actual parse which will be produced. If this parse is not intended, perhaps the speaker should pause somewhere (an unintendedly false name marker can be made a true one by putting a comma pause after it): in fact, the parser will now report an error if it sees a series of merely possible pauses (spaces followed by vowels) followed by a pause or silence after a consonant, after a name marker.

It is important to notice that a name-final description like **la bilti, Djin** is *not* a serial name.

8.4.2 Arguments (including subordinate clauses)

These are the pronouns and noun phrases of Loglan.

We begin with some preliminaries.

A gap, we remind ourselves, is **gu** with optional comma pauses before and/or after it. [As of 5/9, gaps have been subdivided into flavors which can be expressed with **guua** (closing arguments in various contexts), **guea**

(closing description predicates in various contexts), **giuo** (closing sentences in various contexts) and **meu** (closing **me** predicates).]

A laname is **la** followed by an optional comma pause then a mandatory serial name.

A vocative is either **hoi** followed by an optional comma pause followed by a serial name, or **hoi** followed by an optional free modifier followed by a descriptive predicate, followed optionally by a gap (**guea**), (with optional following name [which would follow the gap if it were present] as in **hoi bilti, Djin**), or **hoi** followed by an optional free modifier followed by an argument followed by an optional gap (**guua**), or **hoi** followed by alien text construed as a foreign name. Notice that all vocatives are marked with a name marker word. A serial name by itself is not even an utterance. Notice that vocatives are themselves free modifiers. I have under consideration adding **loa, loi, sia, sie, siu** as vocative markers with an exception: these words cannot be followed by foreign names. One must say **loa lao Xanqipis** rather than **loa Xanqipis**, though one can say **hoi Xanqipis**. The problem is that the social lubricant words are independent UI words and naturally might be used in contexts where one would be in danger of interpreting following material supposed to be meaningful as alien text.

We now present a series of classes which are sorts of argument.

A basic description (**descriptn**) is one of the following sequence of kinds of descriptive phrase (considered in this order):

1. A LE word followed by an optional freemod followed by a description predicate. This is very basic: **le mrenu, le cmalo hasfa**, etc. This is guarded against being an initial segment of a LANAME (so that something like **la Hasfaran** really is read as a LANAME, not as **la Hasfa**, leaving the ran dangling).
2. A LE word followed by a mex (mathematical expression) followed by a description predicate. Freemods are allowed in both medial positions. **Le to mrenu** is a simple example.
3. A LE word followed by a mex followed by an atomic argument (another subtle flavor of argument; atomic arguments include pronouns as well as descriptions) [with freemods insertable in medial positions]. An example is **Le to le mrenu**
4. **ge** followed by a mex followed by a basic description. I need to firm up my understanding of what this case is for.

5. Explicit set and ordered list forms. We emulate the notation $\{x, y, z\}$ for sets or $[x, y, z]$ for lists. The opening and closing brackets are **lau**, **lua** for sets, **lou**, **luo** for lists. The commas are **zeia** for sets, **zeio** for lists. The two grammatical constructions are disjoint (they have parallel structure but they are different classes). The items are of classes “atomic argument” or indefinite. The lists may be single items as in **lua la Djan**, **lau** or empty as in **lou luo**. These constructions can be nested. In the trial.85 grammar these constructions appeared at the very top level in argument and could not enter into any grammatical constructions, which was not satisfactory (much like the trial.85 treatment of abstract predicates built from sentences).

The class of basic descriptions also includes forms in which a LE word is followed by an optional atomic argument not beginning with a quantifier and an optional tense (PA2), followed by any of the things which can follow LE above (cases 1 to 3). This supports things like **lemi hasfa**, **leva hasfa**, **lemina hasfa** formerly handled by compound words in the LE class, and also supports the possessive construction **le la Djan**, **hasfa** and allows its extension to **le la Djan**, **na hasfa**, “John’s present house”. This is a change: the intention is not really that speakers explore the new space created by this proposal (this reader at least does not really like this possessive construction) but that the parallelism which a learner really is likely to feel between **lemi hasfa** and **le le mrenu gu hasfa** should turn out to be really there.

A description (**arg1**) is one of the following quite long laundry list of noun phrase constructions (tested for in the order given by the parser):

1. A LEFORPO word (this class includes LE and the NI cores) followed by a PO word followed by a sentence with fronted arguments, optionally closed with **guo** or a gap. Freemods may appear in medial positions.
2. A LEFORPO word followed by a PO word followed by a sentence, optionally closed with **guo** or a gap. Freemods may appear in medial positions. Note in both of the first two cases that these do not have a PO-initial predunit as a component, though it looks like it is there. We avoid parsing these constructions to include such a predicate to avoid having to close these constructions twice (once for the predicate and once for the argument).
3. As cases 1 and 2, but replacing the PO word with a POA, POE, POI, POO, POU word, and closing respectively with **GUOA**, **GUOE**, **GUOI**,

GUOO, GUOU. This allows efficient closure of abstract descriptions. All the abstract descriptions are now included in a separate subclass **abstractn**.

4. **lio** followed by either a description predicate or a term or a mex (tested for in that order) closed optionally by a gap. This may also be followed by foreign text (digits come to mind).
5. a foreign name starting with **lao** (details under lexicography).
6. a laname (described in preliminaries above). It is important to note that laname is always preferred to basic description.
7. a basic description, optionally followed by a non-pause freemod, followed optionally by a gap (**guua**), followed optionally by a serial name, with the serial name being marked initially either by **ci** (optionally flanked on either side by explicit comma pauses, or just by an explicit comma pause, in which case the initial name word in the serial name should be non-acronymic and contain no false name markers).

The construction without a name is just the very common **le cmalo hasfa**. With a name, we have such things as **le blanu, Djan** or **le blanu ci Djan**, “Blue John”. A practical example of this is to titles: **Le surpoi, Djonz**, “Lord Jones”. I think Mr, Mrs., Miss should be implemented in this way (I am not saying that this was intended, but it is a clear use of this construction).

The requirement of an explicit comma pause before the optional name here (when **ci** is not used) is I believe new, a feature of the general solution of the false name marker problem.

8. A word quoted with LIU or NIU, or a letter quoted with LII. See above.
9. A LIE strong quotation. See above.
10. A LI quotation. See above.

The slightly richer class of atomic arguments (**arg1a**) consists of the following kinds (tested for in this order):

1. a DA pronoun

2. a TAI letteral pronoun
3. a description
4. **ge** followed by an optional freemod followed by an atomic argument. I need to understand the use of prefixing **ge** here.

An atomic argument of any of these shapes may further include a following free modifier.

Note that this class adds in the pronouns. This was already important in understanding basic descriptions above.

We now introduce argument modifiers (subordinate clauses).

An `argmod1` (atomic subordinate clause) consists of an optional **no** of negation (not currently allowed to be followed by a freemod – should I allow this?), then one of the following:

1. a JI word followed by a predicate
2. a JIO word followed by a sentence or sentence with fronted arguments
3. a JI word followed by a modifier (a relative clause)
4. a JI word followed by an argument

Any of these forms are closed by **gui**, but only when they occur alone or as the last element of an afterthought connected chain of such clauses. See `arg2`. Alternative forms in which both the JI word and the matching closer (if there is one) are suffixed with the same one of **za**, **zi**, **zu**, are provided in the alternative parser.

A subordinate clause or argument modifier (class `argmod`) is a series of `argmod1`'s linked by A logical connectives.

An argument of class `arg2` is an atomic argument or an atomic argument followed by one or more argument modifiers, optionally closed by a gap (**gui**). [NOTE: should there be a medial freemod in this rule?]

An `arg3` is either an `arg2` or an `arg2` preceded by a `mex` (a quantifier) with a medial freemod allowed. e.g., **ra le mrenu**.

An `indef1` is a `mex` followed by an optional freemod followed by a description predicate (e.g., **to mrenu**).

An indefinite is an `indef1` followed by an argument modifier. NOTE: as above, should a freemod be allowed medially? There may be a reason not to allow this.

An *arg4* is a string of (possibly mixed) *arg3*'s and indefinites linked by **ze**: this forms mixed arguments. This is a distinct grammatical usage of **ze** from the one as an instance of CA.

An *arg5* is an *arg4* or an *arg4* forethought connected to an *argx* (this class is described soon below): this form is a KA word followed by an *arg4* followed by a KI word followed by an *argx*, with medial freemods allowed.

An *arg6* is an *arg5* possibly modified by **lae** or **lue** (the operator **ie** (interrogative which) removed 4/28/17). Repeated modifications are supported. Medial freemods are allowed.

An *argx* is a possibly multiply negated *arg6* (negation being achieved as usual by prefixing **no** followed by an optional freemod).

An *arg7* is a chain of *argx*'s linked by ACI logical connectives. These group to the left as always.

An *arg8* is a chain of *arg7*'s linked by A connectives: an *arg8* is further constrained not to begin with the *cmappua* **ge**.

An *argument1* is a chain of *arg8*'s linked by AGE connectives, optionally followed by a GUU followed by an *argmod* (allowing attachment of a subordinate clause at the very top level). I suspect that AGE connectives should group to the right as AGE predicate connectives certainly do.

An *argument* is an *argument1* possibly prefixed with one or more case tags, further possibly (multiply) negated.

NOTE: the ability to attach subordinate clauses only to low complexity arguments or at the very top level may be a limitation.

NOTE: do we want to be able to forethought connect subordinate clauses?

NOTE: I have a general concern about where closures of argument constructions are or are not needed. I note that basic description constructions do not have closures at all (different from the situation in **la Sorme Lengu**), but some complex constructions do close. [I do find that *descriptn* can be closed with a gap because *descriptn gap* (**guua**) is a case of *arg1*]. I have an overall impression that closures of constructions involving *argmods* would benefit from an overhaul.

NOTE: the construction of arguments has been modified so that case tags only occur at the very top level.

8.4.3 A semantic note on multiple reference of arguments

Any argument in Loglan may in fact refer to more than one object. **Le mrenu** for example, refers to each of a set of men I have in mind, and makes

whatever assertion is being made of each of them. **me le mrenu** (gu) forms a predicate which applies to exactly the elements of this set,

What requires special note is the extension to be ascribed to a quantified variable. **me ba** is a predicate applying to the domain the variable **ba** (or indefinite argument) is seen to range over in the context. It should be noted that the use of **me** and the transformations it allows mean the the Loglan quantifier does not act merely on a single object but on sets in certain situations: as in a sentence I recently coined, **Ba goi mi sirfio lepo ra me ba, o ba nu krido la Djan, tio**: this asserts the existence of a set of beliefs **ba** of which I am certain John believes exactly those things about the matter at hand. The point here is that we say something not only about each individual possible referent of **ba** (belief held by John about the matter at hand) but about the entire set of them. Such set quantifications have been studied, and writing the Loglan argument analyzer that I plan will require that I work out correct rules of inference for such a system.

8.4.4 Modifiers = relative clauses, prepositional phrases

A tense/location/relative clause (class mod1) consists of a PA word followed optionally by an argument, optionally closed with a gap (**guua** if the argument is present). The option without the argument gives a relative clause which can be distinguished from a tense (a PA word included in a markpred).

A kekmod is a forethought connected modifier: this consists of zero or more **no**'s, followed by a KA word, followed by a top level modifier, followed by a KI word, followed by a modifier of class mod (defined immediately below). Medial freemods are allowed.

A mod is either a mod1, or a mod1 prefixed with one or more negations, or a kekmod.

A modifier (top level) is a mod or a chain of mods linked (left grouped as usual) by A connectives.

8.4.5 Terms, term lists, and termsets (including link sets)

A term is an argument or modifier.

There are two kinds of argument lists, the loose lists which are conglomerations of terms with no explicit operator optionally closed with GUU and the tightly bound lists built with JE/JUE.

A construction of class **terms** is a sequence of one or more terms with optional medial freemods. The lists of terms in the definitions of the basic sentence classes are of this grammatical class (termsets appear internally to predicates only). No more than four un-case-tagged arguments may occur in a string of this class.

The alternative parser imposes the extra requirement that a second or further un-case-tagged argument will not be read as part of a **terms** construction if it would start a sentence. Thus **Na lepo la Djan, kamla mi blanu** actually parses as “When John comes, I am blue”, which is not how the official parser reads it. This is helpful in reading Leith’s sentences as he intended.

A construction of class **subject** is a sequence of terms at least one of which is an argument and no more than one of which is an un-case-tagged argument.

A **termset1** is a construction of class **terms** (in the alternative parser, restricted so that even the first un-case-tagged argument cannot start a sentence) or a forethought construction: a KA word followed by a **termset2** followed optionally by **guu** (or **gu**) followed by a KI word followed by a **termset1** (medial freemods allowed). NOTE: should I include optional negations here?

A **termset2** is a sequence of **termset1**’s linked by A connectives, with medial freemods allowed, and with **guu** or **gu** allowed before A connectives.

A termset is one of the following:

1. an item of class **terms** optionally followed by **go** followed by a bare predicate. The bare predicate modifies the predicate to which the termset is attached; this is a weird but I think useful maneuver.
2. a **termset2**

Empty termsets (**guu** by itself) have been eliminated (to make it more likely that the word **gu** by itself will close what is intended in many cases). This has been done by changing the way that **guu** closes termsets: **guu** does not appear as a final element of a termset, but optionally following termsets in appropriate contexts where the termset class appears. See the PEG grammar for a description.

Termset is an important class, having been introduced earlier as a black box internal feature of the predicate classes.

We now consider the tightly bound lists (culminating in link sets (class `linkargs`)).

A `juelink` is **jue** followed by a term.

A `links1` is a sequence of `juelinks` optionally closed with GUE (either the word **gue**, optionally flanked with explicit pauses on one or both sides, or a gap).

A `links` is one of the following:

1. a `links1`
2. a KA word followed by a `links` followed by a KI word followed by a `links1` (medial freemods allowed).
3. a sequence of items of one of the previous two types linked by A connectives (left grouped as usual).

A `jelink` is **je** followed by a term.

A `linkargs1` is a `jelink` followed by a `links` optionally closed with a GUE. The `links` and the GUE cannot both appear; this avoids double closure issues.

A tightly bound argument list or link set (`linkargs`) is one of the following:

1. a `linkargs1`
2. a KA word followed by a `linkargs` followed by a KI word followed by a `linkargs1` (medial freemods allowed).
3. a sequence of items of one of the previous two types linked by A connectives (left grouped as usual).

The idea is that in these tightly bound argument lists first arguments are attached with **je** and second and subsequent arguments are attached with **jue**, which often reduces the occasion for explicit closures with **gue**. Recall that the class of link sets enters into the construction of atomic predicate units.

9 Appendix: The Current and Recent Active Proposals (and some draft proposals of mine in preparation) with Comments

Any member of the Academy is eligible to have a Proposal posted here, and moreover also to have Comments in their own names posted here. Members of the list are welcome to bother us!

Some of the new Proposals added at the end (which amount to a step by step agenda leading to adoption of as much of this document as the Academy cares to adopt as official) need to have more text added. Proposals which still require that considerable additional language be added are qualified as Draft Proposals.

Proposal 3 2013: (John Cowan): Introduce a word ZAO which when placed between predicates has the same effect as complex formation, and abandon the attempt to form complexes using borrowings.

Proposal 3B 2013 (Randall Holmes): Introduce ZAO as in Cowan's proposal while taking no negative action (complexes with borrowings continue to be allowed, but ZAO is available to paraphrase these or indeed any complexes).

Comments: This proposal is fully implemented in the provisional parser (in the 3B form). It appears as part of the definition of the class of predicate words.

I would encourage prompt action, though I am not pushing action at this time. I support this proposal in the weak sense of 3B: I think after doing work to implement borrowing affixes, that we can keep them. But the *zao* approach has merit.

Proposal 5 2013: (Randall Holmes) Eliminate *noka* and all similar words.

Comments: I do not think this proposal requires any particular action, because I think it is a mistake in the dictionary. I do not think there is much danger of either my parser or LIP ever thinking that it is reading such a word. I have already corrected my parser so that it does not recognize such words. **So one can expect that this proposal will soon disappear from the list.**

Proposal 6 (John Cowan): Eliminate the djifoa (affixes) with the repeated vowels aa/ee/oo and do the required dictionary work to rebuild affected complexes. [he has suggested a more limited proposal to eliminate the EE and OO djifoa]

Comments: My parser does not implement this. It would require massive dictionary work. A revised version leaving the AA djifoa would have a more modest impact. I do not support this, but it is a plausibly motivated proposal and I am happy to leave it out for discussion.

I am not against working on this proposal (perhaps think about eliminating the few EE and OO djifoa), but I think the AA djifoa are too numerous and widely used. In spirit, I agree with John, but this is one of those charming features the language is already committed to.

Proposal 7 2013 (John Cowan – revised to incorporate Proposal 4 text):

1. The sounds of **x**, **q**, **w** to be removed from Loglan. They are permitted only in names, and are relatively low-frequency sounds in the world's languages.
2. The letter **h** to be allowed with either IPA /h/ (its current sound) or IPA /x/ (the current sound of **x**). This will make life easier for Spanish, Russian, and Chinese loglanists, who have /x/ in their languages but not /h/. (Hindi, English, and Japanese have /h/ only, German has both, French has neither.)
3. Extension of **gao**: Currently it is permitted only before "Ceo" and "Vfi" words to make Greek upper case letters. It is to be permitted before any phonological word to make a new word of nurcmapua TAI.
4. Specific new words of TAI to be added to the dictionary: "gaohei" = x, "gaohai" = X, "gaokei" = q, "gaokai" = Q, "gaovei" = w, "gaovai" = W, "gao,alef" = ? (Hebrew letter alef). These replace "xei", "xai", "qei", "qai", "wsi", "wma", and nothing respectively.

[replaces original proposals 4 and 7]

Comments: My parser implements this fully. I have a different proposal for names for the common foreign letters.

I agree with this proposal, with a proviso. I do think that we need CVV words for the Latin letters thus eliminated from the alphabet. They occur commonly in mathematics and in foreign words.

I agree that x,q,w should be eliminated, but I want CVV words for at least lowercase versions of these letters.

I urge immediate discussion (if needed) and ratification (hopefully) of this proposal. Addition of CVV letterals for qwx would then be advisable.

Proposal 8 (Randall Holmes): A predunit appearing in a name must be prefixed with CI. Rescind the earlier decision that we have an additional pause phoneme used only in serial names.

rationale: very simple: this makes La Djan, blanu a sentence rather than a name again, and without multiple grades of pauses.

cautions: make sure there are no ambiguities with existing uses of CI.

La Djan, blanu once again means "John is blue".

La Djan, ci blanu, mrenu becomes "John the Blue is a man". (yes, the pause works to mark the predicate, though this may not be a good practice).

Comments: My parser implements this. This makes an actual incompatibility between my parser and LIP; there are things which each parses which the other does not parse, as predunits are put into serial names in incompatible ways.

In fact, my parser implements the further requirement that a name component following a predunit component must be marked with CI as well. This is all part of a global solution to the name marker problem.

This proposal has passed. It is still on this list because I have not yet updated Appendix H.

There are further related refinements to the definition of serial names implicit in my provisional parser

Proposal 10 (John Cowan): The Loglan Project uses the terms "affix" and "lexeme" in ways that contradict standard linguistic usage. Our complexes are composed entirely of affixes, but an affix to a linguist is either a suffix or a prefix: there must be a root to which they are

attached. I suggest we switch to the neutral term "combining form" until we have a Loglan term analogous to Lojban "rafsi".

Similarly, a "lexeme" is not a word class based on syntactic interchangeability, but one based on sharing an underlying form to which different inflections are added. Thus "run", "runs", "running", "ran" are all members of the "run" lexeme in English ("runner" is not, as the "-er" ending derives a new lexeme). We should instead use "nurcmapua", X is a little-word class including Y.

Of course, this applies only to formal proposals and documentation and where clarity is needed, not to casual loglandic chitchat.

Comment (Randall Holmes) This proposal ties into my program of developing a full Loglan vocabulary for our own grammar. The grammar terms should have English translations that a linguist would understand (and possibly alternative English translations which are traditional in the L community but misleading for linguists, and labelled as such). An implementation of Proposal 10 might be part of an implementation of the Loglan grammar terminology project.

Comment: I would say that this metalevel proposal has in effect been implemented (and thank you John). Please continue virtuously saying "djifoa" or "combining form" instead of "affix" except when alluding to historical documents, fellow logli! I am not as good about "lexeme"; I am trying to say things like "word class".

Also note "syllabic consonant"

Proposal 11 (Randall Holmes): I hereby officially suggest the introduction of an infix -zie- which can be used to merge PA class operators with A-zie-B meaning roughly A-and then-B or "proceeding from A to B"

then replacing each of the compound location operators with a -zie-form

that is, vuva would be replaced by vuzieva

The rationale has been discussed: it is part of the general program of eliminating structure word breaks.

vu, va preda really cannot be construed as meaning the same thing as vuva preda

My parser does not as yet implement this. It would require modest dictionary work to change the compound location operators.

Comments: This is not high on my priority list but it is needed eventually (or something like it). It would appear as part of the Lexer layer of my grand proposal if I had attended to it (which I have not). The exact CVV used needs to be changed because ZIE is now intended to be lower case Latin e. I suggest JIU.

This is not a high priority but it does bear on the issue of compositionality of structure words.

Proposal 12 (John Cowan): Currently, we have NAHU compounds for every NA word which create time, place, and manner questions. Grammatically these are freemods, which means they can appear almost anywhere. I think it would be sufficient to treat these just as regular tagged arguments. "Na hu" as two words would mean "at the same time as what?" which is entirely synonymous with "nahu" meaning "when?"

The only downside is that sentences like "I tu sonli nahu dzoru?", which is one way of saying "When do you sleepwalk?", would have to have the "nahu" moved to somewhere else in the sentence. However, this is only a trivial syntactic change; there is no semantic benefit to having it between two predunits.⁵⁸

Comments: My parser now implements this. It has no effect, for example, on the NB3 corpus. I tu sonli jenahu dzoru would work under my proposal below, so I suggest approving both.

Proposal 13 (Randall Holmes): A change to jelink and juelink.

JE and JUE can currently only be followed by arguments; it should also be permissible to allow them to be followed by modifiers

the rule should be changed to

jelink j- JE term from jelink j- JE argument

(leaving out freemods for clarity)

This is clearly grammatically harmless and allows much finer use of modifiers (PA clauses).

⁵⁸In fact, another proposal makes this work: **I tu sonli je nahu dzoru?**

examples

le mrenu je vi la hasfa bi la Djan

The man who is at the house is John

le bilti je vi lo cutri, nirli ga gudbi sucmi.

The beautiful-in-the-water girl swims well

Notice that this allows tight application of modifier clauses as here in metaphors.

This interacts with John's proposal 12, restoring a lot of the freedom of placement of *nahu* if it becomes a modifier instead of a *freemod*.

I think that JEPA and JUEPA will feel like new classes of words, though there is no need to add them to the grammar:

one is likely to write "le mrenu jevi la hasfa".

My parser implements this.

I would like it if this were ratified reasonably promptly (so I suppose I am in favor of ratifying the previous one at the same time); it is already in my provisional grammar.

Proposal 14 2013: clean up uses of MO (John Cowan): It was pointed out that homonymous uses of *mo* create endless opportunities for LW breaks which must be marked by pauses. I implemented this by eliminating the *-mo* letter construction completely and replacing the 000 numeral with *moa* in my parser.

The similar changes to MA that he suggested are not needed, as other changes that I make disambiguate the uses of MA.

Proposal 1 2014: introduce SIE: I propose the introduction of a new word *sie* expressing apology rather than mere regret: *uu* currently expresses both, and it is an important distinction to draw. I run into this problem in speech in English frequently and I have encountered it in Loglan.

If one says *Sie* by itself (I'm sorry) I think that *Siu* would be an appropriate response (rather in the spirit of "don't mention it", which is also a phrase which can be used in place of "you are welcome", the current official translation of *siu*).

I'm very fond of this very modest proposal: I would like to see it ratified. I have installed the word in the provisional parser.

Proposal 2 2014: eliminate vowel-initial letterals: The vowel-initial letterals are a pain. They create the only situation where CVV-V occurs in compound little words (in acronyms, and strictly speaking this will not be entirely eliminated) and they appear to require an additional clause in the formation of predicates to handle compounds like X-ray with the letter a vowel (A-train). I modestly propose that we introduce CVV letterals for vowels. ZIA, ZIE, ZII, ZIO, ZIU are free. One might want the ZUv series as well for upper case. We could then eliminate all the vowel initial letterals and the need for special rules in various situations. I would assume one would keep the ability to abbreviate vowels in acronyms.

Note that one would not want to use -zie- as the linker for compound location operators in this case. I have proposed JIU instead.

There are now no Cvv/V joints at all in the PEG grammar, even in acronyms, if the VCV letterals are dropped. I leave the phonetic possibility open, but I eliminated it in acronyms without VCVs by requiring z before an abbreviated vowel in an acronym.

comment added 2/5/2016: I am in no particular hurry to delete the VCV letterals, as they do appear extensively in old Loglan text. In fact, I have done maintenance on them, ensuring that one does not need to precede them with explicit pauses and that they can be used in acronyms. I do however encourage the use of the new forms in modern text.

Currently the ZIV and ZIVma (capitalized) vowels are present as an alternative to the VCV letterals.

Proposal 3 2014, 3/9/2014: I have withdrawn my proposal to move words like rana from PA to mod1. I am convinced that they can sensibly be used as prepositions.

Proposal 4, 2014 (Randall Holmes): I found the word riyhasgru in the dictionary, which my parser views as an error, but I am told by James that legacy software does assume CVy djifoa correlated with CV cma-pua. I do not recall that CVy djifoa are documented anywhere, and I

do not like them. I propose that the CVh djifoa be assigned to correlate with those CV cmapua which do not already have djifoa. None of these are in use (they are not ideal as they must be y hyphenated) – their pronunciation can be more definite than their spelling suggests because the hard pronunciation of h as ch in loch can be used. This requires no parser changes and would change *riyhasgru* to *rihyhasgru* in the dictionary. There might be other words of this kind which would have similar systematic modifications.

This is not something I regard as highest priority. The alternative would be to implement CVy djifoa, to which I have phonetic objections (too easily confused with unstressed CVV djifoa); John convinced me that my alternative scheme with CVry and CVny djifoa was a bad idea.

Proposal 1 2015, 9/5/2015: I propose, following the cue of style objections to this kind of sentence raised by Steve Rice in L3, that a sentence optionally beginning with one or more modifiers followed by a tense-marked predicate should always be understood as a gasent; if the final **GA terms** clause is missing, **ga ba** should be understood. So **Donsu ti mi** means “Give this to me”, as before, but **Fazi dons u ti mi** is no longer a deprecated imperative, but instead is to be understood as **Fazi dons u ti mi (ga ba)**, “Someone is about to give this to me”. **Vi le hasfa fazi dons u ti mi** means “Someone is about to give it to me in the house” (what it means now is actually open to some debate); if the term **vi le hasfa** is replaced by an argument we get an ordinary sentence in which **donsu** has too many arguments. This proposal removes the case **terms gasent** from the grammar, as it is rather difficult to tell what to do with an argument appearing before a gasent; the intent of the framers must have been to allow initial modifiers.

```
statement <- (gasent / (terms (freemod)? gasent) /
              (terms (freemod)? predicate))
```

is replaced by

```
statement <- gasent / modifiers freemod? gasent/
           terms freemod? predicate
```

`modifiers` is a new class, a string of modifiers. The class `gasent` also has to be modified to allow the final GA terms component to be omitted.

This will shortly be implemented in the provisional grammar.

Addendum: I have also arranged for sentences in which initial modifiers are followed by an unmarked predicate to be understood as imperatives (in `sen1` rather than `statement`), which really must be the intention. It is useful to note that there is clear discussion of the rules we are changing here and their motivations in the commentary on Group J grammar rules in NB3. It is quite clear that misrecognizing a sentence like `Na la Ven, donsu ta mi` as a declarative sentence goes right back to the NB3 period.

Further Addendum: There are multiple claims in NB3 (and correlated claims made in recent loglanist discussion) that recognizing forms intended to be imperatives is syntactically difficult. I respectfully disagree. Looking at what the old parser does (it parses a sentence as an imperative exactly if it is a predicate by itself), and considering that fronted modifiers should not affect imperative status, gives a definition of imperative sentence which is easy to implement in PEG format and would be equally easy to implement in BNF: `predicate`, or `modifiers predicate`. Then consideration of the style point raised by Steve Rice, and the desirability of avoiding radical rethinking of argument places during the reading of a sentence, suggests that optional `modifiers` followed by a tensed predicate should be construed as a `gasent`, with the `ga terms` component construed as `ga ba` if it never appears [there is an interesting question here as to what to do if the `ga terms` component contains more than one argument]. We do note however, that on reflection we understand what the issue was: the provision that the terms before the predicate contain one argument, or that the terms in a `ga terms` suffix to a `gasent` contain exactly one element, are quite difficult to express with a BNF grammar of the kind which can be automatically checked for ambiguity.

With regard to recent loglanist discussions, the issue was raised of the effect on these considerations of the presence of case tags. My response is that case tags have no effect on the status of a term as argument or modifier; if it has a case tag it is an argument. Telling whether a sentence is imperative in form or not remains easy. What can be difficult is determining exactly what the sentence means, but this is because the system of case tags creates confusing questions re assignment of arguments to argument places of predicates (see Proposal 12 below). Sentences with strange patterns of case tags are hard to understand even without the complication of being imperatives.

An interesting point about this proposal is that it has hardly any effect on whether any sentence is parsable. It does forbid formation of **terms gasent** sentences in which the terms include an argument, and JCB says in NB3 that such sentences do not make sense. The actual effect is to redraw the boundary between declarative sentences and imperatives: a sentence in which no terms or only modifiers appear before the unmarked predicate is imperative, and a sentence in which no terms or only modifiers appear before a marked predicate is a declarative sentence with indefinite subject (understood as a **gasent** with omitted final **ga ba**).

Further Addendum: We propose that the **ga terms** component of a **gasent**, if present, should contain exactly one argument, or all the arguments in the sentence, to avoid retrospective shifting of all arguments appearing before the **ga terms** suffix. This is now enforced by the parser.

Proposal 2, 2015 (Randall Holmes): I propose that the phonetic and word form parsing in the provisional parser be accepted as it stands.

The rationale is that as far as permitted shapes of **cmapua**, predicate and name words go this is conservative. Name words are restricted to those which can be resolved into syllables, but no name actually used has had to have its spelling changed except for doubling of syllabic consonants, a spelling rule which is actually proposed in L1 (1989). The changes from the 1989 language that this proposal requires for predicates have already been ratified by the Academy. (In parsing the *Visit*, I also encountered issues with doubled non-continuant consonants and with names ending with three or more consonants (often fixable

by doubling a continuant).

The phonetic parsing aspect causes explicit comma pauses to be permitted in more places than LIP permits them (anywhere that one can pause). Whitespace is not permitted in places where one cannot pause (in the middle of words). The close-comma for syllable breaks is replaced with the hyphen, which can thus no longer be used to abbreviate **y**. None of these are major changes. One gains the ability to explicitly indicate ordinary and emphatic stress, which can be useful rhetorically even if one is not writing phonetically. The ability to write and parse phonetic transcripts is a brand new capability we have never had before.

My belief about this proposal is that so far as the orthographic style goes I have implemented the intentions of NB3 faithfully, with considerable effort, and having a fully precise definition of Loglan phonetics, including a definition of the syllable, is a great advance. The introduction of phonetic transcripts is an entirely new opportunity for language testing, and I have made considerable use of it in connection with other problems, such as the false name marker issue.

Proposal 3, 2015 (Randall Holmes): I propose that the definitions of word classes in the provisional parser be adopted as they stand, subject to discussion of some particular points, and apart from the handling of acronyms and quotation/foreign text constructions. The APA words might be worth deleting, but I think the IPA words, which have the same phonetic problems but occur only as sentence connectives, would need to stay as they are, and the point can be made that the solution for the IPA words also works for the APA words, which are rather common in the NB3 corpus. The question of the semantics of **efa** versus **erau** should be considered. My belief is that they should stay reversed; we should agree to treat APA and AKOU differently (and similarly for IPA and IKOU).

There are specific points regarding terminating forms for PA and NI words which the Academy might want to examine. There is a change in internally logically connected PA words which I can show is necessary to avoid ambiguities (a structure word break issue).

This proposal needs careful review of the structure of the large word classes, A and its relatives (including the vexed APA words) and their I analogues, the large PA class, the large NI class, and the LE class

(not as complex). I might have overlooked something in this list. The classes implemented by my parser are demonstrably not the same as those implemented in LIP, but I believe that all commonly used words are supported, and this is a systematic and precise definition, which is something we need. Some language about individual word classes may be wanted as part of the proposal.

Proposal 4, 2015 (Randall Holmes): The overhaul of acronyms embodied in the provisional parser should be adopted. Acronymic predicates are to be replaced with acronymic names, which automatically solves the problem of marking where acronyms begin and end. Use of **me** can recover predicates where desired. Acronymic dimension suffixes acquire an initial marker **mue** and must end with pauses. There is no need for pauses between letteral pronouns appearing as successive arguments, because there is no way that such a chain of letterals can be confused with an acronym once acronyms are front marked and terminated by pauses.

Single vowel items in acronyms are eliminated in favor of -zV- items.

The names of the vowels with the form **Vfi** and **Vma** are to be eliminated in favor of **ziV** and **ziVma** (including **ziy** and **ziyma** as an irregular form). This eliminates a weird phonetic irregularity.

Draft Proposal 5, 2015 (Randall Holmes): The treatment of quotation and alien text constructions in the provisional parser is to be adopted.

The only point which I think may be controversial is the quite different strong quotation construction, which I will explain in an essay.

An aspect of this is that we endorse Steve Rice's position that the construction with **lao** intended for Linnaean names by JCB should be used for all foreign names, and so names with **la** should be phonetic: **la Ainctain** but **lao Einstein**.

John Cowan has expressed objections to the -za and -zi (?) qualifiers for spoken and text quotation introduced by the previous Academy in the late 90's; details such as this could be discussed.

Draft Proposal 6, 2015 (Randall Holmes): The suite of changes required to solve the false name marker problem is to be implemented. These

need to be itemized carefully in the proposal text here, and the principles explained. Two changes along these lines have already been ratified, the elimination of unmarked vocatives and the use of **ci** to mark predunits in serial names (which removes the need for two pause phonemes), but there are others.

To complete the overhaul of serial names, a name word appearing after a predunit also needs to be marked with **ci**, whether it contains a false name marker or not.

A name word appearing as the final component of an **arg1** as in **le blanu, Djan**, must be preceded by an explicit comma-marked pause and must be marked with **ci** if it contains a false name marker. Acronymic names must always be marked.

After the name markers **la, hoi, hue** a name word is read in preference to anything else. If one of these is to be followed by a different grammatical construction which closes with a name word, some pause will be needed to indicate this. Similar remarks apply to **ci** in the context of a serial name.

It is always permissible to put an explicit comma pause between a name marker and the following name, and indeed this is the way to make a false name marker a true one if necessary.

False name markers are restricted to occurrences of name markers in name words with the property that the tail after the name marker is itself a well-formed name word.

The theory is that the left boundary of a name word should now be easier to recognize. The right boundary is always easy to spot.

In an example such as **ladjan, clu'valameris** the parser (here working in phonetic mode) does not mistake **clu'valameris** for the second part of a serial name, because **clu'valameris** contains a name marker (in this case indicating an actual name) and so would have to be marked if it were a name. This is an example of why unmarked vocatives had to be eliminated: the ability to put an unmarked name anywhere a free modifier could go would create all sorts of unintended parses. In the current grammar, an unmarked name can occur only after a name in a serial name, or in the special case of **arg1** exemplified above. In both cases, we forbid the unmarked name from containing a name marker, so

its context and the fact that it is consonant final indicate unequivocally that it really is a name (no final segment of it will turn out to be an intended name or an alien text construction – the latter involve pauses). It is the speaker's responsibility to insert pauses to ensure that non-name constructions following name markers are articulated correctly.

Draft Proposal 7, 2015 (Randall Holmes): The restructuring of the grammar of **po** predicates and **le po** clauses (these being separate constructions, both closed with a single GUO) is to be accepted. Examples and clarification to be added in the proposal text.

Draft [Proposal 8, 2015 (Randall Holmes): The structure of logically connected predicates and shared termsets embodied in the classes backpred through predicate2 is to be adopted. This requires a supporting essay on why it is safe to abandon earlier distinctions between marked and unmarked forms, and why the ACI connective have been given the same privileges as A connectives (but binding more tightly), and a discussion of the issue of termsets shared by logically connected predicates.

Proposal 9, 2015 (Randall Holmes): After action on the previous proposals and upon review by the Academy (and a check by myself that I have not smuggled in any further major grammatical changes that need to be made into separate proposals), the provisional PEG parser, as possibly modified due to action on previous proposals, is to be adopted as the official TLI Loglan parser. The official parser would be frozen at the point of adoption [to be changed only by official Academy action], and would remain distinct from my working parser of the moment, to which I would keep making experimental changes.

I do note that some minor changes were made in the conversion of the trial.85 grammar into the PEG grammar. A particular change which I made in several places was allowing freer use of logical connections of some forms.

Draft Proposal 10 (2015): This is a semantic not a grammatical proposal (and in its present form unfinished and mostly a note to myself). Where an indefinite pronoun of the BA class appears at its first occurrence with a subordinate clause formed with JI/JIO, its position for purposes of interpreting order of quantifiers is not that position but its first position

in the subordinate clause. Give examples to make clear why this is important.

In **teba jio tobe cluva ba**, we are considering three people who are loved by the same two people; in **teba jio ba nu cluva tobe** we are considering three people each of whom is loved by two people (who may vary depending which of the three we are talking about).

My understanding is that this problem was considered in Lojban and they at least considered allowing subordinate clauses to appear before the things they are attached to to avoid this quantifier order problem. I think the solution here is much simpler.

Proposal 11, 2015 (Randall Holmes): I suggested originally that where an answer is to be given which will be in the form of a predicate, and so confusable with an imperative, that the freemod **soi dapli** be used to signal that here we have an answer not a command.

Tu he speni

Gudbi!

The first speaker asks “How are you doing?”. Is the reply “Well” or “Be good!”?

Gudbi, soi dapli is clearly the former.

The dialogue

Tu he speni

Gudbi, soi korji!

is the unlikely “How are you dong? Be good!”. (Thanks to Gleki for pressing me to include this).

However, I now further propose an official answer attitudinal **seu**, attachable to any utterance which is the answer to a question (a vocative can be used further to indicate who is being answered) and strongly suggested or mandatory in the case of a predicate answer. Thus

bf Tu he speni letu likcke?

reply 1: **Seu gudbi** (it was good!)

reply 2: **Gudbi!** (Be good! this likely means, don't ask...)

The attitudinal **seu** might be useful in helping a listener to accept as utterances some of the very unlikely things which can be answers (especially if the question was not heard), and may have a further use to give “answers” where no question was actually asked (this use would call for a vocative to indicate whose utterance is your target).

Proposal 12 (Randall Holmes, with acknowledgements to John Cowan)

:

This proposal concerns how to fill argument places of a predicate when it is supplied with a mixture of tagged and untagged arguments (tagged with numerical or non-numerical case tags).

We first consider any sentence without a **headterms GI** component, and whose predicate is not logically connected with CE connectives and/or equipped with tightly bound arguments using JE/JUE. In such a sentence, we fill argument places, reading left to right, assigning each tagged argument to the argument place that the tag gives it (more than one argument may be assigned to the same place, which has the effect of a logical conjunction, or of things being mutually related in the case of a predicate with more than one argument place with the same non-numerical tag), and assigning each untagged argument place the first argument place distinct from the places already assigned (this may be the first argument place, as in **Zue da blanu de**, which is synonymous with **De blanu da**. In a gasent or imperative, the first argument place is not regarded as available (it is reserved implicitly for the person addressed or an indefinite subject which may eventually appear in a **ga** initial phrase. Note that a later tagged argument may be assigned the same argument place of the predicate as an untagged argument assigned the same place earlier, as in **Tu donsu zua la Djan, ta**, “You and John give that to someone”.

There are serious issues concerning what case tags *mean*, notably numerical case tags, when applied to arguments of tightly bound predicates linked with CE logical connectives and with arguments attached to them by JE/JUE links. It is arguable that these predicates have their own argument place structures, and the numerical case tags for these will not correlate with the numerical case tags for the component predicates. Examples to be presented. The non-numerical case tags for such connected forms will be even weirder (they will only make sense

when a numerically indexed argument of the composite predicate has the same non-numerical tag with respect to each component predicate).

In a sentence which begins with a **headterms GI** component, we propose to remove any dependence on the last argument place of any predicate. We propose that the first argument in a **headterms GI** prefix should be tagged as a matter of style [or perhaps as a grammar rule] and that untagged arguments should have their argument places set relative to the most recent tagged argument, with untagged arguments being assigned to the argument place succeeding the argument place of the previous argument, argument places being excluded only if they are known to be filled, which can only be due to having two previous numerically tagged arguments. An initial untagged argument, if permitted, should be retroactively assigned the first place not known to have been used at the end of reading the entire sentence, and subsequent arguments are then assigned to subsequent places until a tagged one is reached. This should not happen as a matter of style. Where a case tag is used to set an argument place, of course we do not know where the block of arguments starts until we know what the predicate is, and we cannot have sufficient information to skip an argument place.

Proposal 13, Randall Holmes, 2016: I propose the addition of new series of consonant lower case letterals **C-eiu** and uppercase letterals **C-aiu**.

These are both useful potentially for anaphora and provide us with names for certain foreign letters:

Haiu, heiu for X, x.

Kaiu, keiu for Q, q.

Vaiu, veiu for W, w.

Who knows what symbols the other letters in this series may stand for?

Further, completing the elimination of X from the dictionary, I propose the borrowing **haiukre** for X-ray, admitting immediately that it is a hack!

This proposal is fully implemented in my parser and in my versions of the dictionary.

Proposal 14, Randall Holmes, 2016: I propose that the short scope abstraction operators be **poi**, **pui**, **zoi**, for phonetic regularity. I propose that **zoa** be the single prime and **zoo** the double prime in class NI.

I propose the addition of new long scope PO words with the shape **poiV**, **puiV**, **zoiV**. These will build abstract descriptions and predicates with new closure words of the shape **guoV** with the same final vowel. The reason for doing this is that one can expect to be able to close any reasonable number of abstractions with a single word, as long as one follows the discipline of using different abstraction constructors at different levels of nesting.

Alternatively (if forms like **poia** are thought too likely to break up into **po ia**) forms PO-z-(a/i/u) and closures GUO-z(a/i/u) are an alternative form of the proposal.

This proposal is fully implemented in my parser (in both phonetic forms). The first line is implemented in my versions of the dictionary.

This proposal and the previous one make use of the additional shape Cvv-V of a little word unit which we have from NB3 but have never used, except in nasty deprecated ways in acronyms using the VCV letterals.

Omnibus Phonetic Proposal, 2017, Randall Holmes See above: it is a section in this document.

10 The PEG implementing the Phonetics Proposal, the new provisional grammar (the following one is preserved because the comments are valuable)

This version is mildly buggy; use the separately posted version on my page. I'll update it eventually; I'd like to work things out so that I do not have to correct right margins running off the page!

```
# In this file I will develop the entire Loglan grammar on top of the
phonetic proposal
```

```
# This is now done, in a first pass. That is, the grammar is adapted
and appears to work, more or less.
```

```
# What is needed is comments on the lexicography and the grammar...
Phonetics has now pretty clearly been sorted
```

```
# from the grammar (there are some places where the phonetics
accept grammar information with regard to punctuation).
```

```
# Alien text is now handled somewhat differently. Some issues to
do with quoting names are not finalized and have not been tested.
```

```
# I added -iy and -uy as VV forms allowed in general in cmapua
but not in other words; they are always monosyllabic. What this
# immediately allows me to do is to give Y a name which is not
phonetically irregular! <ziy> is supported: <yfi> is too, now.
```

```
# capitalization is roughly back to where it was in the original,
but all-caps are allowed.
```

```
# acronyms are liable to be horrible.
```

```
# Fixed the recursion problem in a way which will not be visible
in ordinary parses. Streams of cmapua will always
```

```
# be broken at name or alien text markers (instead of using
```

lookahead to check that we do not stand at the beginning
of a name word or alien text word). The next cycle will
then check for a name or alien text, and also check for
badnamemarkers; no lookahead is happening while a
stream of cmapua is being read except checking for
the markers of names and alien text. This will change
the way phonetic parses look (streams of cmapua will
break (and sometimes resume) at name markers or
alien text markers, but it will not change any grammatical
parses.

#Part I Phonetics

Mod bugs, I have implemented all of Loglan phonetics
as described in my proposal.
Borrowing djifoa are pretty tricky.

I have now parsed all the words in the dictionary,
and all single words of appropriate classes parse successfully.
I have added alien text and quotation constructions which
do not conform to these rules; so actually
all Loglan text should parse, mod some punctuation

and capitalization issues. The conventions for
alien text here are not the same as those in the
current provisional parser.

I believe the conventions for forcing comma pauses
before vowel initial cmapua and after names
except in special contexts have been enforced.
In a full grammar, one probably would want
to disable pauses before vowel initial letterals.
This grammar also does not support the lingering
irregularities in acronyms.

This grammar is entirely about phonetics:
all it does is parse text into names (with associated initial
pauses or name markers), cmapua (qua unanalyzed

```
streams of cmapua units),
# borrowings and complexes, along with interspersed
  comma pauses and marks
# of terminal punctuation. It does support conventions
  about where commas are required
# and a simple capitalization rule.

# a likely locus for odd bugs is the group
of predstartX rules which detect apparent
  cmapua which
# are actually preambles to predicates.
These are tricky! (and I did indeed find some lingering
# problems when I parsed the dictionary).

# In reviewing this, I think that very little
  is different from 1990's Loglan (the borrowing djifoa
# are post-1989 L1, but not my creation).
  Some things add precision without making
  anything in 1990's Loglan incorrect.
# The requirement that syllabic consonants
be doubled is new, and makes some 1990's
  Loglan names incorrect.
# The requirement that names resolve into
  syllables is new, and makes some 1990's
  Loglan names incorrect,
# usually because they end in three consonants.
# The rule restricting final consonant pairs
from being noncontinuant/continuant is new, but
# does not affect any actual predicate ever proposed.
# Enhancing the VccV rule to also forbid
CVVV...ccV caused one predicate to be changed
# (<haiukre> became <haiukrre>, and
haiukre was a novelty anyway, using a new name for X in X-ray)
# The exact definition of syllables and use of
syllable breaks and stress marks is new (the close comma
# was replaced with the hyphen,
so Lo,is becomes Lo-is); but this does not
make anything in 1990's Loglan
```

incorrect, it merely increases precision
and makes phonetic transcript possible.
Forbidding doubled vowels in borrowings
was new, was already approved, and caused us to change
<alkooli> to <alkoholi>.
Formally allowing the CVccVV and CVcccV
predicates without y-hyphens took a proposal
in 2013 because
Appendix H was careless in describing their
abandonment of the slinkui test, but the dictionary
makes it evident that this was their intent
all along. The slinkui test had already been
abandoned in the 1990s.
Formally abandoning qwx was already
something that the dictionary workers
in the 1990's were working
on; we completed it.
Allowing glottal stop in vowel pairs and
forbidding it as an allophone of pause is a new phonetic
feature in the proposal but not reflected
in the parser, of course. Alternative pronunciations of
y and h and allowing h in final position are invisible
or do not make any 1990's Loglan incorrect.
Permitting false name markers in names was already
afoot in the 1990's and the basic outlines of our
approach were already in place. The rule requiring
explicit pauses between a name marker not starting
a name word and the beginning of the next name
word is new, but reflects something which was already
a fact about 1990's Loglan pronunciation: those
pauses had to be made in speech! The requirement
that names resolve into syllables restricts which
literal occurrences of name markers are actually
false name markers (the tail they induce in the
name must itself resolve into syllables).
Working out the full details of borrowing djifoa
was interesting: I'm not sure that I've done anything
new there; explicitly noting the stress shift

```
in borrowing djifoa might be viewed as something
# new but it is a logical consequence of JCB's
  permission to pause after a borrowing djifoa,
  which contains
# explicit language about how it is to be stressed,
  and the
# final definition of a borrowing djifoa as simply
  a borrowing followed by -y. The shift strikes
# me as a really good idea anyway, because
it marks djifoa with a pause after it as phonetically different
# in an additional way other than ending with the
  very indistinct vowel y. My rules as given here do not
# directly enforce the rule that a borrowing djifoa
  must be preceded by y but I think they indirectly
# enforce it in all or almost all cases: the parser
  tries to read a borrowing djifoa before reading
# any other kind of djifoa, so it is hard to see
  how to deploy a short djifoa in such a way that
  it would
# fall off the head of a borrowing without using y.
# These phonetics do not support certain i
rregularities in acronyms. I note that some of those
# could be fixed by changing all Ceo letterals to Ceio.

#Sounds

#all vowels

V1 <- [aeiouyAEIOUY]

#regular vowels

V2 <- [aeiouAEIOU]

#consonants

C1 <- [bcd fghjklmnp rstvz BCDFGHJKLMNPRSTVZ]
```



```

# letters

letter <- (![qwxQWX] [a-zA-Z])

# a capitalization convention which allows what our
# current one allows and also allows all-caps.
# if case goes down from upper case to lower case,
# it can only go back up in certain cases. This
# does allow capitalization of initial segments of words.
# There is a forward reference to the grammar
# in that free capitalization of embedded literals
# is permitted, and capitalization of vowels
# guarded with z in literals as in DaiNaizA.

lowercase <- (![qwx] [a-z])

uppercase <- (![QWX] [A-Z])

caprule <- [\"? &((letter TAI0)/([z] V1)/
!(lowercase uppercase).) letter (&((letter TAI0)/
([z] V1)/!(lowercase uppercase).) (letter/juncture))*
!(letter/juncture)

# to really get all Loglan text, we should add
# the alien text constructions and the markers of alien text,
# <lie>, <lao>, <sao>, <sue> and certain
# quotations which violate the phonetic rules.

# we adopt the convention that all alien text
# may be but does not have to be enclosed in quotes.
# it needs to be understood that in quoted alien text,
# whitespace is understood as <, y,>; in the unquoted
# version this is shown explicitly.

# this is a little different from what is allowed
# in the previous provisional parser, but similar.
# A difference is that all the alien text markers
# are allowed to be followed by the same sorts of alien text.

```

```

# the forms with <hoi> and <hue> are required
  to have following quotes in written form to avoid
# unintended parses, which otherwise become
  likely in case of typos in non-alien text cases.

AlienText <- ([,]? [ ]+ [\""] (![\""].)+ [\""]/ [,]? [ ]+
(![, ]!terminal .)+ ([,]? [ ]+ [y] [,]? [ ]+ (![, ]!terminal .)+)*

AlienWord <- &caprule ([Hh] [Oo] [Ii] juncture?
&([,]? [ ]+ [\""])/[Hh][Uu] juncture? [Ee] juncture?
&([,]? [ ]+ [\""]) / [Ll] [Ii] juncture? [Ee] juncture?/
[Ll] [Aa] juncture? [Oo]/[Ss] [Aa] juncture? [Oo] juncture?/
[Ss] [Uu] juncture? [Ee] juncture?) AlienText

# while reading streams of cmapua,
the parser will watch for the markers of alien text.

alienmarker <- ([Hh] [Oo] [Ii] juncture?
&([,]? [ ]+ [\""])/[Hh][Uu] juncture? [Ee] juncture?
&([,]? [ ]+ [\""]) / [Ll] [Ii] juncture? [Ee] juncture? /
[Ll] [Aa] juncture? [Oo] juncture? /
[Ss] [Aa] juncture? [Oo] juncture?/
[Ss] [Uu] juncture? [Ee] juncture?)

# syllable markers:  the hyphen is always
medial so must be followed by a letter.
# the stress marks can be syllable final
and word final.  A juncture is never followed
# by another juncture.

juncture <- (([-] &letter)/[\'*\']) !juncture

stress <- [\'*\'] !juncture

# terminal punctuation

terminal <- ([.:?!;])

```

```

# characters which can occur in words

character <- (letter/juncture)

# the continuant consonants
  and the syllabic pairs they can form

continuant <- [mnlrMNLr]

syllabic <- (([mM] [mM] !(juncture? [mM]))/
([nN] [nN] !(juncture? [nN]))/
([rR] [rR] !(juncture? [rR]))/
([lL] [lL] !(juncture? [lL])))

# the obligatory monosyllables,
  and these syllables when
  broken by a usually bad syllable juncture

MustMono <- (([aeoAEO]
[iI])/([aA] [oO]))

BrokenMono <- (([aeoAEO] juncture
[iI])/([aA] juncture [oO]))

# the obligatory and optional monosyllables

Mono <- (MustMono/([iI] V2)/([uU] V2))

# vowel pairs of the form found in cmapua and djifoa.

VV <- (!BrokenMono V2 juncture? V2)

# the next vocalic unit to be chosen from a stream of vowels
# in a predicate or name. This is different than in our Sources
# and formally described in the proposal.

NextVowels <- (MustMono/(V2 &MustMono)/Mono/V2)

```

the doubled vowels that trigger the rule that one of them must be stressed

```
DoubleVowel <- (([aA] juncture? [aA])/
([eE] juncture? [eE])/([oO] juncture? [oO])/
([iI] juncture [iI])/([uU] juncture [uU]))
```

the mandatory "vowel" component of a syllable

```
Vocalic <- (NextVowels/syllabic/[y])
```

the permissible initial pairs of consonants, and the same pairs possibly
broken by syllable junctures.

```
Initial <- (([Bb] [Ll])/([Bb] [Rr])/
([Cc] [Kk])/([Cc] [Ll])/([Cc] [Mm])/
([Cc] [Nn])/([Cc] [Pp])/([Cc] [Rr])/
([Cc] [Tt])/([Dd] [Jj])/([Dd] [Rr])/
([Dd] [Zz])/([Ff] [Ll])/([Ff] [Rr])/
([Gg] [Ll])/([Gg] [Rr])/([Jj] [Mm])/
([Kk] [Ll])/([Kk] [Rr])/([Mm] [Rr])/
([Pp] [Ll])/([Pp] [Rr])/([Ss] [Kk])/
([Ss] [Ll])/([Ss] [Mm]) / [Ss] [Nn] /
([Ss] [Pp])/([Ss] [Rr])/([Ss] [Tt])/
([Ss] [Vv])/([Tt] [Cc])/([Tt] [Rr])/
([Tt] [Ss])/([Vv] [Ll])/([Vv] [Rr])/
([Zz] [Bb])/([Zz] [Ll])/([Zz] [Vv]))
```

```
MaybeInitial <- (([Bb] juncture? [Ll])/([Bb] juncture? [Rr])/
([Cc] juncture? [Kk])/([Cc] juncture? [Ll])/([Cc] juncture?
[Mm])/([Cc] juncture? [Nn])/([Cc] juncture? [Pp])/
([Cc] juncture? [Rr])/([Cc] juncture? [Tt])/([Dd] juncture?
[Jj])/([Dd] juncture? [Rr])/([Dd] juncture? [Zz])/
([Ff] juncture? [Ll])/([Ff] juncture? [Rr])/([Gg] juncture?
[Ll])/([Gg] juncture? [Rr])/([Jj] juncture? [Mm])/
([Kk] juncture? [Ll])/([Kk] juncture? [Rr])/([Mm] juncture?
[Rr])/([Pp] juncture? [Ll])/([Pp] juncture? [Rr])/
([Ss] juncture? [Kk])/([Ss] juncture? [Ll])/([Ss] juncture?
```

```

[Mm]) / [Ss] juncture? [Nn] / ([Ss] juncture? [Pp]) /
([Ss] juncture? [Rr]) / ([Ss] juncture? [Tt]) / ([Ss] juncture?
[Vv]) / ([Tt] juncture? [Cc]) / ([Tt] juncture? [Rr]) /
([Tt] juncture? [Ss]) / ([Vv] juncture? [Ll]) / ([Vv] juncture?
[Rr]) / ([Zz] juncture? [Bb]) / ([Zz] juncture? [Ll]) /
([Zz] juncture? [Vv])

# the permissible initial consonant groups in a syllable.
  Adjacent consonants should be initial pairs.
# Triples of continuants should not be formed.
  Such a group is of course followed by a vocalic unit.

# this rule for initial consonant groups is stated in NB3.

# I forbid a three-consonant initial group
  to be followed by a syllabic pair. This seems obvious.

InitialConsonants <- ((!syllabic C1 &Vocalic) /
(! (C1 syllabic) Initial &Vocalic) /
(&Initial C1 !(C1 syllabic) Initial !syllabic &Vocalic))

# the forbidden medial pairs and triples.
  These are forbidden regardless of placement
# of syllable breaks.

# each of these is actually a single consonant
  followed by an initial, and the idea was to
  identify CVC-CCV junctions which
# would be hard to pronounce. But the
placement of the syllable break is not relevant
  to the exclusion of the sequence.
# Notice that the continuant syllabic pairs are excluded:
  this prevents final consonants from being included in such pairs.

NoMedial2 <- ([Bb] juncture? [Bb]) / ([Cc] juncture? [Cc]) /
([Dd] juncture? [Dd]) / ([Ff] juncture? [Ff]) / ([Gg] juncture?
[Gg]) / ([Hh] juncture? C1) / ([Jj] juncture? [Jj]) / ([Kk] juncture?
[Kk]) / ([Ll] juncture? [Ll]) / ([Mm] juncture? [Mm]) / ([Nn]

```

```

juncture? [Nn])/([Pp] juncture? [Pp])/([Rr] juncture? [Rr])/
([Ss] juncture? [Ss])/([Tt] juncture? [Tt])/([Vv] juncture?
[Vv])/([Zz] juncture? [Zz])/([CJSZcjsz] juncture? [CJSZcjsz])/
([Ff] juncture? [Vv])/([Kk] juncture? [Gg])/([Pp] juncture?
[Bb])/([Tt] juncture? [Dd])/([FKPTfkpt] juncture? [JZjz])/
([Bb] juncture? [Jj])/([Ss] juncture? [Bb]))

```

```

NoMedial3 <- (([Cc] juncture? [Dd] juncture? [Zz])/
([Cc] juncture? [Vv] juncture? [Ll])/([Nn] juncture?
[Dd] juncture? [Jj])/([Nn] juncture? [Dd] juncture?
[Zz])/([Dd] juncture? [Cc] juncture? [Mm])/
([Dd] juncture? [Cc] juncture? [Tt])/([Dd] juncture?
[Tt] juncture? [Ss])/([Pp] juncture? [Dd] juncture?
[Zz])/([Gg] juncture? [Tt] juncture? [Ss])/
([Gg] juncture? [Zz] juncture? [Bb])/([Ss] juncture?
[Vv] juncture? [Ll])/([Jj] juncture? [Dd] juncture?
[Jj])/([Jj] juncture? [Tt] juncture? [Cc])/
([Jj] juncture? [Tt] juncture? [Ss])/([Jj] juncture?
[Vv] juncture? [Rr])/([Tt] juncture? [Vv] juncture?
[Ll])/([Kk] juncture? [Dd] juncture? [Zz])/([Vv] juncture?
[Tt] juncture? [Ss])/([Mm] juncture? [Zz] juncture? [Bb]))

```

The syllable.

there are no formal rules about syllables
as such in our Sources, which is odd since
the definition of predicates depends on the
placement of stresses on syllables.

The first rule enforces the special
point needed in complexes that
a CVC syllable is preferred to a CV syllable
where possible; we economically apply
the same rule for default placement of
syllable breaks everywhere, which is, with
that exception, that the break comes as soon as possible.

the SyllableB approach is taken

```
if the following syllable would otherwise start with a syllabic pair.

# the reason for this approach is
that if one syllabizes a well formed complex in this way...
# the syllable breaks magically fall
on the djifoa boundaries. This does mean that the
# default break in <cabro> is <cab-ro>,
which feels funny but is harmless. Explicitly breaking
# it <ca-bro> will also parse correctly.

SyllableA <- (C1 V2 FinalConsonant? (!Syllable FinalConsonant?)) !syllabic

SyllableB <- (InitialConsonants? Vocalic
  (!Syllable FinalConsonant)? (!Syllable FinalConsonant?))

Syllable <- ((SyllableA/SyllableB) juncture?)

# The final consonant in a syllable.
  There may be one or two
final consonants. A pair of final
# consonants may not be a non-continuant
followed by a continuant. A final consonant may not
# start a forbidden medial pair or triple.

# The rule that a final consonant pair
may not be a non-continuant followed by a continuant
# is natural and obvious but not in our Sources.
  Such a pair of consonants would seem to
# naturally form another syllable.

FinalConsonant <- !syllabic (!(continuant C1 !Syllable continuant)
  !NoMedial2 !NoMedial3 C1 !(juncture? V2))

# Here are various flavors of syllable we may need.

# this is a portmanteau definition of
a bad syllable (the sort not allowed in a borrowing).
```

```
SyllableD <- &(InitialConsonants? ([y]/
DoubleVowel/BrokenMono/&Mono V2 DoubleVowel/
!MustMono &Mono V2 BrokenMono)) Syllable

# this (below) is the kind of syllable
# which can exist in a borrowed predicate:
# it cannot start with a continuant pair,
# it cannot have a y as vocalic unit,
# and its vocalic unit (whether it
# has one or two regular vowels)
# cannot be involved in a double vowel
# or an explicitly broken
# mandatory monosyllable.

BorrowingSyllable <- !syllabic (!SyllableD) Syllable

# this is the final syllable of a predicate. It cannot be followed
# without pause by a regular vowel.

VowelFinal <- InitialConsonants? Vocalic juncture? !V2

# syllables with syllabic consonant vocalic units
# this is only used in borrowings, and we *could* reasonably
# require it to be followed by a vowel. But I won't for now.
# for gluing this works, but we might literally borrow predicates
# with syllabic continuant pronunciations.

SyllableC <- (&(InitialConsonants? syllabic) Syllable)

# syllables with y

SyllableY <- (&(InitialConsonants? [y]) Syllable)

# an explicitly stressed syllable.

StressedSyllable <- ((SyllableA/SyllableB) [\'*])

# a final syllable in a word, ending in a consonant.
```



```

NameEndSyllable <- (InitialConsonants?
  (syllabic/Vocalic &FinalConsonant)
  FinalConsonant? FinalConsonant? !letter)

# the pause classes actually hang on the letter before the pause.

# whitespace which might or might not be a pause.

maybepause <- (V1 [\']*? [ ]+ C1)

# explicit pauses: these are whitespace
before a vowel or after a consonant, or comma marked pauses.

pause <- ((C1 [\']*? [ ]+ &letter)/
  (letter [\']*? [ ]+ &V1)/(letter [\']*? [,] [ ]+ &letter))

# these are final syllables in words followed
by whitespace which might not be a pause.
# the definition actually doesnt mention
the maybepause class.

MaybePauseSyllable <- InitialConsonants?
  Vocalic ['*]? &([ ]+ &C1)

# this class requires pauses before it, after all the phonetic word classes.
# what is being recognized is the beginning of a logical connective.

# To avoid horrible recursion problems,
  giving this a concrete phonetic definition
# without much lookahead.
This can go right up in the phonetics section if it works.

connective <- [ ]* (([Nn] [Uu] juncture? &([Uu]/[Nn]))?
  ([Nn] [Oo] juncture?)? V2 juncture? !V2
  !([Ff] [Ii]) !([Mm] [Aa]) !([Zz] [Ii]) !(C1 juncture? C1)

# The full analysis of names.

```

```

# a name word (without initial marking)
  is resolvable into syllables and ends with a consonant.

PreName <- ((Syllable &Syllable)* NameEndSyllable)

# this is a busted name word with whitespace in it --
  but not whitespace at which one has to pause.

BadPreName <- (MaybePauseSyllable [ ]+/
Syllable &Syllable)* NameEndSyllable

# This is a name marker followed by
a consonant initial name word without pause.

# I deployed a minimal set of name marker words;
I can add the others whenever.

MarkedName <- &caprule ((([Ll] !pause [Aa] juncture?)/
([Hh] [Oo] !pause [Ii] juncture?) / ([Hh] [Uu] juncture?
!pause [Ee] juncture?) / ([Cc] !pause [Ii] juncture?)/
([Ll] [Ii] juncture? !pause [Uu] juncture?)/[Gg][Aa] !pause
[Oo] juncture?) [ ]* &C1 &caprule PreName)

# This is an unmarked name word with a false name marker in it.

FalseMarked <- (&PreName (!MarkedName character)* MarkedName)

# This is the full definition of name words.
  These are either marked consonant initial names
without pause defined above,
# names without false name markers
beginning with explicit pauses (either
comma marked or vowel-initial)
# and name markers followed, with or
  without pause, by name words.
  In the latter case there must be at least
# whitespace before a vowel initial name.

```

```

# a series of names without false name markers
  and names marked with ci, separated by spaces,
  may be appended.

# there is a look ahead at the grammar:
a NameWord can be followed without explicit pause
  (there is whitespace and
# a pause in speech!) by another
# kind of utterance only in a serial name when
  what follows is of the form <ci> predunit, to be included
# in the name.

NameWord <- (&caprule MarkedName/
  ([,] [ ]+ !FalseMarked &caprule PreName)/
  (&V1 !FalseMarked &caprule PreName)/
  &caprule ((([Ll] [Aa] juncture?)/([Hh] [Oo] [Ii] juncture?)/
  ([Cc] [Ii] juncture?)/([Ll] [Ii] [Uu] juncture?)) !V1 [,]? [ ]*
  &caprule PreName))([,]?[ ]+ !FalseMarked &caprule
  PreName/[,]?[ ]+ &([Cc] [Ii]) NameWord)*
  &([ ]* [Cc] [Ii] predunit/&([,] [ ]+ /terminal/!.) ./!.)

# this is the minimal set of name marker words we are using.
  We may add more.

# I am contemplating adding the words of
  social lubrication as name markers, but in a more restricted
# way that in the last provisional parser, in which I made
  them full-fledged vocative markers.

namemarker <- ([Ll] [Aa] juncture?/[Hh] [Oo] [Ii] juncture?/
  ([Hh] [Uu] juncture? [Ee] juncture?)/[Cc] [Ii] juncture?/
  [Ll] [Ii] juncture? [Uu] juncture?/[Gg] [Aa] [Oo] juncture?) !V1

# this is the bad name marker phenomenon
  that needs to be excluded. This captures the idea
# that what follows the name could be pronounced
  without pause as a name word according to the

```

```
# orthography, but the fact that whitespace is
present shows that this is not the intention.

badnamemarker <- namemarker !V1 [, ]? [ ]* BadPreName

# we could exclude every cmapua unit from
being of class badnamemarker, or more economically exclude each
# individual name marker word where it appears
in the grammar as a cmapua word. In this document we
# will take the former approach because we do not
have the grammar yet. (At the moment I am adding
# the full grammar and keeping the check for
badnamemarker condition everywhere).

# We have at any rate completely solved
the phonetic problem of names and their markers.

# predicate start tests: the idea is the
same as class "connective" above, to recognize
# the start of a predicate without recursive
appeals to the whole nasty definition of predicate.
# The reason to do it is to recognize when
CV^n followed by CC cannot be a cmapua unit.

# we note, though we have taken no action that
# (1) we don't see any reason for there to be
any predicates starting CVVVV..CC; there is
at least one starting CVVV
# (2) actually, any CVVVV...CC segment can *only*
start a predicate, if one takes all rules into account. But
# since this is incredible we haven't worried about it.
# If we forbade CVVVV... to start a predicate
this would sharply bound the lookahead here.
# However, CVVVV... never occurs in legal Loglan text.

# not all of the things identified are actually
starts of predicates: some are simply illegal.
```

```

# these tests are designed to ensure
that an initial CV^n in a predicate is recognized
# by the rule scanning for cmapua.

# if we encounter a CC which is not initial
the (C)V^n before it is part of the predicate.

predstart1 <- (C1? (V2 juncture?)+
  (&MaybeInitial C1 juncture?)* !MaybeInitial C1 juncture? C1)

# (C)V^n<stress>CC starts a predicate.
  The stress might fall between the consonants!
# it might also be a hidden stress
  visible only via the doubled vowel rule.

predstart2 <- (C1? (V2 juncture? &V2)* V2['*'] C1 juncture? C1)

predstart2a <- (C1? (V2 juncture? &V2)* V2 C1 ['*'] C1)

predstart3 <- (C1? (V2 juncture? &(V2 juncture? V2))*
  DoubleVowel C1 juncture? C1)

# CCVV and CCCVV predicates are ruled out.
So a CV^n preamble before CCVV or CCCVV
must be part of a predicate.

predstart4 <- (C1? (V2 juncture?)+ (&Initial InitialConsonants/
&MaybeInitial C1 juncture Initial) V2 juncture? V2 juncture? !character)

predstart4a <- (C1? (V2 juncture?)+
  (&Initial InitialConsonants/
  &MaybeInitial C1 juncture Initial) V2 ['*'] !Mono V2)

# this is a special condition under which
an initial string (C)V^n is permitted to fall off.
# this handles the forbidden VCCV and
  CVVV...CCV predicates by allowing the front to fall off
# (even if finally stressed, which is a bit odd).

```

The prohibition of VCCV predicates is from the
 # 1990's; such predicates would not fall
 apart themselves, but their borrowing djifoa would.
 # The same is true of some CVVV...CCV predicates,
 and we forbid these as well.

```
predstart5 <- (C1? V2 juncture?
(V2 juncture? &V2)+ V2/VV/V2) juncture? MaybeInitial V2
```

```
# this must start a predicate
because there is hardly anything after the initial pair.
```

```
predstart6 <- (C1? (VV/V2) juncture?
(&Initial InitialConsonants/&MaybeInitial C1 juncture Initial) V2 !character)
```

```
# this is a special case found in words like <hidrroterapi>.
The rule forbidding
# borrowings to start with syllables
with continuant glue forces this.
```

```
predstart8 <- (C1? (V2 juncture?)+ (Initial/MaybeInitial/C1) syllabic)
```

```
# these cases recognize affixing of short djifoa to the fronts
# of predicates with y-hyphens.
```

```
predstart9 <- C1 VV juncture? [y]
```

```
predstart10 <- (C1 V2 juncture? C1 juncture? [y])
```

```
# attachment of the CVCCY form
```

```
predstart11 <- (C1 V2 juncture? C1 juncture? C1 [y])
```

```
# by throwing in Maybeinitial I finished
enforcing the stress rule: <abrodi>
# is a cmapua followed by the predicate
<brodi>, but <a'brodi> raises alarms because
# the cmapua is stressed and <brodi> is in predstart.
```

```

# and here at last is predstart.
This function does lookahead to a
  consonant pair and decides
# whether what it passes over
is an initial segment of a predicate (in practice, it is trying to
# decide whether it can be a cmapua).

predstart <- !predstart5 (predstart1 / predstart2 /
predstart2a /predstart3/predstart4/predstart4a/
predstart6/predstart8/predstart9/predstart10/predstart11/MaybeInitial)

# cmapua units starting with consonants.
This is the exact description from NB3. The fancy tail in each of the
# three cases is enforcing the rule about
pausing before a following predicate if stressed.

# consonant initial cmapua units may not be followed by vowels without pause.

# I am adding <iy> and <uy> (always monosyllable, yuh and wuh)
  as vowel pairs permitted in VV and CVV cmapua units.
# The use for this envisaged is that the name <ziy> of Y
  becomes easy to introduce. Adding word space
# is always nice, and these words seem pronounceable.
  I also made <yfi> possible: Y now has phonetically
# regular names.

CmapuaUnit <- (C1 Mono juncture? V2 !(['*] [ ]* predstart) juncture? !V1/
C1 (VV/[Ii][Yy]/[Uu][Yy]) !(['*] [ ]* predstart) juncture? !V1/
C1 V2 !(['*] [ ]* predstart) juncture? !V1)

# A stream of cmapua is read until the start of a predicate
  or a name marker word or an alien text marker word is encountered.
# the stream might resume with a name marker word if it does not
  in fact start a name word and does not potentially start a name
# word due to inexplicit whitespace
(doesn't satisfy the bad name marker condition).

```

```

# we force explicit comma pauses before logical connectives,
  but not before vowel initial cmapua in general;
# other conditions force at least whitespace,
  which does stand for a pause, before such words.

# a special provision is made
  for NO UI forms as single words. <yfi> is supported.

Cmapua <- &caprule !badnamemarker (!predstart [Nn] [Oo] juncture?
  !predstart (VV/[Ii][Yy]/[Uu][Yy]) !(['*] [ ]* predstart) juncture?/
  ((!predstart (VV/[Ii][Yy]/[Uu][Yy]) !(['*] [ ]* predstart) juncture?)+ /
  ((!predstart V1 !(['*] [ ]* predstart) juncture?)/ !predstart CmapuaUnit)
  (!namemarker !alienmarker !predstart CmapuaUnit)*)/
  !predstart V2 !(['*] [ ]* predstart) juncture?) !V1 !([ ]* (connective))

# I have apparently now completely solved
  the problem of parsing cmapua as well as name words.

# Now for predicates.

# the elementary djifoa (not borrowings)

# various special flavors of these djifoa will be needed.
# These are the general definitions.

# The NOY and Bad forms are for use
  for testing candidate borrowings for resolution
# with bad syllable break placements. Borrowings do not contain Y...

# CVV djifoa with phonetic hyphens.

CVV <- C1 VV
(juncture? [y] [-]? &letter /[r] juncture? &C1/[n] juncture? &[r]/juncture?)

CVVNoHyphen <- C1 VV juncture?

CVVHiddenStress <- C1 &DoubleVowel V1 [-] V1
([-]? [y] [-]? &letter /[r] [-]? &C1/[n] [-]? &[r]/[-]?)

```



```

CVVFinalStress <- C1 VV (['*]
[y] [-]? &letter /[r] ['*] &C1/[n] ['*] &[r]/['*])

CVVNOY <- C1 VV ([r] juncture? &C1/[n] juncture? &[r]/juncture?)

CVVNOYFinalStress <- C1 VV ([r] ['*] &C1/[n] ['*] &[r]/['*])

CVVNOYMedialStress <- C1 V2 ['*] V2 [-]?

FinalCVV <- (CVV !character/CVVNOYMedialStress/CVVHiddenStress)

# CCV djifoa with phonetic hyphens.

CCV <- Initial V2 (juncture? [y] [-]? &letter/juncture?)

CCVStressed <- Initial V2 (['*] [y] [-]? &letter/['*])

CCVNOY <- Initial V2 juncture?

CCVBad <- MaybeInitial V2 juncture?

CCVBadStressed <- MaybeInitial V2 ['*]

FinalCCV <- (CCV !character/CCVStressed)

# CVC djifoa with phonetic hyphens.
These cannot be final and are always followed by a consonant (well, the
# -y form may be followed by a vowel...
# an eccentric syllable break is supported if the CVC is y-hyphenated:
# <me-ky-kiu> and <mek-y-kiu> are both legal. The default is the latter.

CVC <- (C1 V2 !NoMedial2 !NoMedial3 C1
(juncture? [y] [-]? &letter/juncture? &C1)/C1 V2 juncture C1 [y] [-]? &letter)

CVCStressed <- (C1 V2 !NoMedial2 !NoMedial3 C1 (['*] [y] [-]? &letter/
['*] &letter)/C1 V2 ['*] C1 [y] [-]? &letter)

```

```
CVCNOY <- C1 V2 !NoMedial2 !NoMedial3 C1 juncture? &C1
CVCBad <- C1 V2 !NoMedial2 !NoMedial3 juncture? C1 &C1
CVCNOYStressed <- C1 V2 !NoMedial2 !NoMedial3 C1 ['*'] &C1
CVCBadStressed <- C1 V2 !NoMedial2 !NoMedial3 ['*'] C1 &C1
# the five letter forms (always final in complexes)
CCVCV <- Initial V2 juncture? C1 V2 [-]?
CCVCVStressed <- Initial V2 ['*'] C1 V2 [-]?
CCVCVBad <- MaybeInitial V2 juncture? C1 V2 [-]?
CCVCVBadStressed <- MaybeInitial V2 ['*'] C1 V2 [-]?
CVCCV <- (C1 V2 juncture? Initial V2 [-]?/
C1 V2 !NoMedial2 C1 juncture? C1 V2 [-]?)
CVCCVStressed <- (C1 V2 ['*'] Initial V2 [-]?/
C1 V2 !NoMedial2 C1 ['*'] C1 V2 [-]?)
# the medial five letter djifoa
CCVCY <- Initial V2 juncture? C1 [y] [-]?
CVCCY <- (C1 V2 juncture? Initial [y] [-]?/
C1 V2 !NoMedial2 C1 juncture? C1 [y] [-]?)
CCVCYStressed <- Initial V2 ['*'] C1 [y] [-]?
CVCCYStressed <- (C1 V2 ['*'] Initial [y] [-]?/
C1 V2 !NoMedial2 C1 ['*'] C1 [y] [-]?)
# to reason about resolution of borrowings into
```

```

both syllables and djifoa (we want to exclude the latter
# but we need to define it adequately) we need to
  recognize where to stop. A predicate word ends either
# at a non-character (not a letter or syllable mark:
whitespace, comma or terminal punctuation) or it
# has an explicit or deducible penultimate stress.
Borrowings do not contain doubled vowels, so they
# have to have explicit stress in the latter case.

# analysis: the stressed tail consists of
a stressed syllable followed by an unstressed syllable.
# identifying an unstressed final syllable is
complicated by recognizing which CVV combinations can
# be one syllable. This will either be an explicitly
  stressed syllable followed by a single syllable
# or a syllable suitable to be stressed followed
  by an explicitly final syllable. CVV djifoa can
# contain both syllables in a tail and of course
the five letter djifoa have to be tails. A never stressed
# SyllableC (with a continuant) may intervene.

# tail of a borrowing with an explicit stress

BorrowingTail1 <- !SyllableC &StressedSyllable BorrowingSyllable
  (!StressedSyllable &SyllableC BorrowingSyllable)?
  !StressedSyllable &BorrowingSyllable VowelFinal

# tail of a borrowing or borrowing djifoa with no explicit stress

BorrowingTail2 <- !SyllableC BorrowingSyllable
  (!StressedSyllable &SyllableC BorrowingSyllable)?
  !StressedSyllable &BorrowingSyllable VowelFinal (&[y]!/character)

# tail of a stressed borrowing djifoa,
different because stress is shifted to the end

BorrowingTail3 <- !SyllableC !StressedSyllable BorrowingSyllable
  (!StressedSyllable &SyllableC BorrowingSyllable)?

```

```
&BorrowingSyllable InitialConsonants? Vocalic ['*] &[y]

BorrowingTail <- BorrowingTail1 / BorrowingTail2

# short forms that are ruled out: CCVV and CCCVV forms.

CCVV <- (InitialConsonants V2 juncture? V2 juncture? !character /
  InitialConsonants V2 ['*] !Mono V2 juncture?)

# the CVVV...CCV and VCCV forms are ruled out
(rule predstart5 above is about this)

# a continuant syllable cannot be initial in a
  borrowing and there cannot be successive continuant
# syllables. There really ought to be no more than one!

# borrowing, before checking that it doesnt resolve into djifoa

PreBorrowing <- !predstart5!CCVV!Cmapua!SyllableC
(!BorrowingTail!(StressedSyllable)!(SyllableC SyllableC)BorrowingSyllable)*
  BorrowingTail

# ditto for an explicitly stressed borrowing

StressedPreBorrowing <- !predstart5!CCVV!Cmapua!SyllableC
(!BorrowingTail!(StressedSyllable)!(SyllableC SyllableC)BorrowingSyllable)*
  BorrowingTail

# borrowing djifoa without explicit stress (before resolution check)

PreBorrowing2 <- !predstart5!CCVV!Cmapua!SyllableC(!BorrowingTail
!(StressedSyllable)!(SyllableC SyllableC)BorrowingSyllable)* BorrowingTail2

# stressed borrowing djifoa (before resolution check).

PreBorrowing3 <- !predstart5!CCVV!Cmapua!SyllableC
(!BorrowingTail3!(StressedSyllable)!(SyllableC SyllableC)BorrowingSyllable)*
  BorrowingTail3
```

```

# Now comes the problem of trying to say
that a preborrowing cannot resolve into cmapua. The difficulty is with
# recognizing the tail, so making sure that
  the two resolutions stop in the same place.

# we know because it is a borrowing that
there is at most one explicit stress, and it has to fall
# in one of the cmapua! This should make it doable.

# borrowing djifoa are terminated with y,
  so the final djifoa needs to take this into account

RFinalDjifoa <- (CCVCVBad/CVCCV/
CVVNoHyphen/CCVBad/CVCBad) (&[y]/!character)

RMediallyStressed <- (CCVCVBadStressed/
CVCCVStressed/CVVNOYMedialStress)

RFinallyStressed <- (CVVNOYFinalStress/
CCVBadStressed/CVCBadStressed/CVCNOYStressed)

BorrowingComplexTail <- (RMediallyStressed/
RFinallyStressed (CVVNOY/CCVBad)/RFinalDjifoa)

ResolvedBorrowing <- (!BorrowingComplexTail
(CVVNOY/CCVBad/CVCBad))* BorrowingComplexTail

# borrowed predicates

Borrowing <- !ResolvedBorrowing &caprule PreBorrowing !([ ]* (connective))

# explicitly stressed borrowed predicates

StressedBorrowing <- !ResolvedBorrowing &caprule
StressedPreBorrowing !([ ]* &V1 Cmapua)

#This is the shape of non-final borrowing djifoa.

```

```
Notice that a final stress is allowed.
#The curious provision for explicitly stressing
  a borrowing djifoa and pausing is supported.

# borrowing djifoa without explicit stress
  (stressed ones are not of this class!)
# Note that one can pause after these (explicitly, with a comma)

BorrowingDjifoa <- !ResolvedBorrowing &caprule
PreBorrowing2 (['*] y [,] [ ]+/juncture? [y] [-]?)

# stressed borrowing djifoa finally implemented!

StressedBorrowingDjifoa <- !ResolvedBorrowing
&caprule PreBorrowing3 [y] [-]? ([,] [ ]+)?

# We resolve complexes twice, once into syllables
  and once into djifoa. We again have to ensure that
# we end up in the same place!
The syllable resolution is very similar to that of borrowings;
# the unstressed middle syllable of the tail can be a SyllableY,
  and can also be a
# SyllableC if the final djifoa is a borrowing.

# A stressed borrowing djifoa
with the property that the tail is still a phonetic complex is
# a unit for this analysis.

PhoneticComplexTail1 <- !SyllableC !SyllableY
&StressedSyllable Syllable (!StressedSyllable
&(SyllableC/SyllableY) Syllable)? !StressedSyllable !SyllableY VowelFinal

PhoneticComplexTail2 <- !SyllableC !SyllableY Syllable
(!StressedSyllable &(SyllableC/SyllableY) Syllable)?
!StressedSyllable !SyllableY VowelFinal !character

PhoneticComplexTail <- PhoneticComplexTail1 / PhoneticComplexTail2
```

```
PhoneticComplex <- !CCVV!Cmapua!SyllableC
(StressedBorrowingDjifoa &PhoneticComplex/
!PhoneticComplexTail!(StressedSyllable)
!(SyllableC SyllableC)Syllable)* PhoneticComplexTail

# the analysis of final djifoa and stressed djifoa
# differs only in details from
# what is above for resolution of borrowings.
# The issues about CVV djifoa with doubled
# vowels are rather exciting.

# a stressed borrowing djifoa with the tail
# still a phonetic complex is a black box unit for
# this construction.

# My approach imposes the restriction on JCB's
# "pause after a borrowing djifoa" idea that what follows
# the pause must itself contain a penultimate stress:
# <igllu'y, mao> is a predicate but <igllu'y, mao> is not.
# while <iglluy', gudmao> is a predicate.

FinalDjifoa <- (Borrowing/StressedBorrowing/
CCVCV/CVCCV/CVVNoHyphen/CCV) !character

MediallyStressed <- (CCVCVStressed/
CVCCVStressed/CVVHiddenStress/CVVNOYMedialStress)

FinallyStressed <-(StressedBorrowingDjifoa/
CCVCYStressed/CVCCYStressed/CVVHiddenStress/
CVVFinalStress/CCVStressed/CVCStressed)

ComplexTail <- (FinallyStressed (&(C1 Mono) CVVNoHyphen/
CCV)/MediallyStressed/FinalDjifoa)

PreComplex <- (!CVVHiddenStress (!ComplexTail)
(StressedBorrowingDjifoa &PhoneticComplex/
BorrowingDjifoa/CVCCY/CCVCY/CVV/CCV/CVC))* ComplexTail
```

```

# the initial condition forces a CVC which can be removed
still leaving a complex to be
# y-hyphenated.

Complex <- !(C1 V2 &(C1 juncture? !FinalCVV !FinalCCV PreComplex) MaybeInitial)
&caprule &PreComplex PhoneticComplex !([ ]* (connective))

# format for the LI quote

LiQuote <- &caprule [Ll][Ii]juncture? comma2?
[" "] phoneticutterance [" "] comma2?
&caprule [Ll][Uu]juncture? !([ ]* connective)

Word <- (NameWord / Cmapua !([ ]*Cmapua)/ Complex)

# it is an odd point that all borrowings parse as complexes --
  so when I parsed all the words they all
# parsed as complexes.
A borrowing is a complex consisting of a single final borrowing djifoa!

SingleWord <- (Borrowing !./Complex !./
  Word !./PreName !. /CVV/CCV/CVC) !.

# name word appearing initially without leading spaces is important,
  because one type of NameWord includes a leading comma.

phoneticutterance1 <- (NameWord /
  [ ]* LiQuote/[ ]* NameWord/[ ]* AlienWord/
  [ ]*Cmapua/[ ]* '--'/[ ]* '...'/[ ]* Borrowing![y]/[ ]* Complex)+

phoneticutterance <- (phoneticutterance1/[ ,][ ]+/terminal)+

# Part II Lexicography

# In this section I develop the grammar of words in Loglan.
  I'll work by editing the original provisional PEG grammar.

# consonants and vowel groups in cmapua

```



```
badstress <- ['*] [ ]* predstart  
  
B <- (!predstart [Bb])  
  
C <- (!predstart [Cc])  
  
D <- (!predstart [Dd])  
  
F <- (!predstart [Ff])  
  
G <- (!predstart [Gg])  
  
H <- (!predstart [Hh])  
  
J <- (!predstart [Jj])  
  
K <- (!predstart [Kk])  
  
L <- (!predstart [Ll])  
  
M <- (!predstart [Mm])  
  
N <- (!predstart [Nn])  
  
P <- (!predstart [Pp])  
  
R <- (!predstart [Rr])  
  
S <- (!predstart [Ss])  
  
T <- (!predstart [Tt])  
  
V <- (!predstart [Vv])  
  
Z <- (!predstart [Zz])  
  
a <- ([Aa] !badstress juncture? !V1)
```

```
e <- ([Ee] !badstress juncture? !V1)
i <- ([Ii] !badstress juncture? !V1)
o <- ([Oo] !badstress juncture? !V1)
u <- ([Uu] !badstress juncture? !V1)

V3 <- juncture? V2 !badstress

AA <- ([Aa] juncture? [a] !badstress juncture? !V1)
AE <- ([Aa] juncture? [e] !badstress juncture? !V1)
AI <- ([Aa] [i] !badstress juncture? !(V1 V1))
AO <- ([Aa] [o] !badstress juncture? !(V1 V1))
AU <- ([Aa] juncture? [u] !badstress juncture? !V1)
EA <- ([Ee] juncture? [a] !badstress juncture? !V1)
EE <- ([Ee] juncture? [e] !badstress juncture? !V1)
EI <- ([Ee] [i] !badstress juncture? !(V1 V1))
EO <- ([Ee] juncture? [o] !badstress juncture? !V1)
EU <- ([Ee] juncture? [u] !badstress juncture? !V1)
IA <- ([Ii] juncture? [a] !badstress juncture? !(V1 V1))
IE <- ([Ii] juncture? [e] !badstress juncture? !(V1 V1))
II <- ([Ii] juncture? [i] !badstress juncture? !(V1 V1))
IO <- ([Ii] juncture? [o] !badstress juncture? !(V1 V1))
```

```

IU <- ([Ii] juncture? [u] !badstress juncture? !(V1 V1))
OA <- ([Oo] juncture? [a] !badstress juncture? !V1)
OE <- ([Oo] juncture? [e] !badstress juncture? !V1)
OI <- ([Oo] [i] !badstress juncture? !(V1 V1))
OO <- ([Oo] juncture? [o] !badstress juncture? !V1)
OU <- ([Oo] juncture? [u] !badstress juncture? !V1)
UA <- ([Uu] juncture? [a] !badstress juncture? !(V1 V1))
UE <- ([Uu] juncture? [e] !badstress juncture? !(V1 V1))
UI <- ([Uu] juncture? [i] !badstress juncture? !(V1 V1))
UO <- ([Uu] juncture? [o] !badstress juncture? !(V1 V1))
UU <- ([Uu] juncture? [u] !badstress juncture? !(V1 V1))

# adding the new IY and UY, which might see use some time.
# they are mandatory monosyllables but do not take a possible additional
# following vowel as the regular ones do. So far only used in <ziy>.

IY <- [Ii] [Yy] !badstress juncture? !V1
UY <- [Uu] [Yy] !badstress juncture? !V1

# this is a pause not required by the phonetics

PAUSE <- [,] [ ]+ !(V1/connective) &caprule

# more punctuation

comma <- [,] [ ]+ &caprule

```

```

comma2 <- [,]? [ ]+ &caprule

end <- (([ ]* '#' [ ]+ utterance)/([ ]+ !.)/!.)

# this rule allows terminal punctuation to be followed by an inverse vocative,
# a frequent occurrence in Leith's novel, and something which makes sense.

period <- (([!.:;?] (&end/([ ]+ &caprule))) (invvoc period?))

# Letters with y will be special cases
# idea: allow IY and UY (always monosyllables)
# as vowel combinations in cmapua only.
# done: Y has a name now. <yfi> is also added.

# the classes in this section after this point
# are the cmapua word classes of Loglan
# (if they begin with [ ]* or a word class).
# I suppose the alien text classes are not really word classes,
# but they are lexicographic items, as it were.
# Paradoxically, the PA and NI classes admit internal explicit pauses.
# So of course do predicate words!

# Loglan does admit true multisyllable cmapua:
# there are words made of cmapua units which have joints between
# units at which one cannot pause without breaking the word.
# Lojban, I am told, does not.

# this version has the general feature that
# the quotation and alien text constructions are not hacked:
# they are supported by the phonetic rules
# (as dire exceptions, of course) and the grammatical constructions
# conform with the phonetic layer.
# Alien text and utterances quoted with <li>...<lu>
# can be enclosed in double quotes.
# LI only supports full utterances, for the moment.
# All alien text constructors take the same class as argument:
# the vocative and inverse vocative *require* quotes

```

to avoid misreading ungrammatical expressions with typos
 # as correct (inverse) vocatives.

the names <yfi>, <ziy> for Y are supported.

The Ceo names are left as they are. I decided that a second short series
 # of letteral pronouns is actually a reasonable use of short words,
 and the Ceio words are there for other uses.

```
TAIO <- (V1 juncture? M a/V1 juncture? F i/
V1 juncture? Z i/!predstart C1 AI u?/
!predstart C1 EI (u)?/!predstart C1 EO/
Z [i] V1 !badstress juncture? !V1 (M a)?)
```

```
NOI <- (N OI)
```

```
A0 <- &Cmapua (a/e/o/u/H a)
```

```
A <- [ ]* !predstart !TAIO
(N [u] &(u/ N [o]))? (N [o])? AO NOI?
!([ ]+ PANOPAUSES PAUSE) !(PANOPAUSES !PAUSE [ ,])
(PANOPAUSES ((F i)/&PAUSE))?
```

```
ANOFI <- [ ]* (!predstart !TAIO
((N [u]) &(u/(N [o]))))? (N [o])? AO NOI? PANOPAUSES?)
```

```
A1 <- A
```

```
ACI <- (ANOFI C i)
```

```
AGE <- (ANOFI G e)
```

```
CAO <- (((N o)? ((C a)/(C e)/(C o)/(C u)/(Z e)/(C i H a))) NOI?)
```

```
CA1 <- (((N u) &((C u)/(N o)))? (N o)? CAO
!([ ]+ PANOPAUSES PAUSE) !(PANOPAUSES !PAUSE [ ,]) (PANOPAUSES ((F i)/&PAUSE))
```

```
CA1NOFI <- (((N u) &((C u)/(N o)))? (N o)? CAO PANOPAUSES?)
```

```
CA <- ([ ]* CA1)

ZE2 <- ([ ]* (Z e))

I <- ([ ]* !predstart !TAIO i !([ ]+ PANOPAUSES PAUSE)
!(PANOPAUSES !PAUSE [ ,]) (PANOPAUSES ((F i)/&PAUSE))?)

ICA <- ([ ]* i ((H a)/CA1))

ICI <- ([ ]* i CA1NOFI? C i)

IGE <- ([ ]* i CA1NOFI? G e)

KAO <- ((K a)/(K e)/(K o)/(K u)/(K i H a))

KOU <- ((K OU)/(M OI)/(R AU)/(S OA)/(M OU)/(C IU))

KOU1 <- (((N u N o)/(N u)/(N o)) KOU)

KA <- ([ ]* (((N u) &(K u))? KAO)/((KOU1/KOU) K i)) NOI?)

KI <- ([ ]* (K i) NOI?)

KOU2 <- (KOU1 !KI)

BadNIStress <- ((C1 V2 V2? stress (M a)? (M OA)? NI RA)/
(C1 V2 stress V2 (M a)? (M OA)? NI RA))

NIO <- (!BadNIStress ((K UA)/(G IE)/(G IU)/
(H IE)/(H IU)/(K UE)/(N EA)/(N IO)/(P EA)/(P IO)/
(S UU)/(S UA)/(T IA)/(Z OA)/(Z OO)/(H o)/(N i)/
(N e)/(T o)/(T e)/(F o)/(F e)/(V o)/(V e)/(P i)/
(R e)/(R u)/(S e)/(S o)/(H i)))

SA <- (!BadNIStress ((S a)/(S i)/(S u)/(IE (comma2? !IE SA?))) NOI?)

RA <- (!BadNIStress ((R a)/(R i)/(R o)))
```

```

NI1 <- ((NIO (!BadNISTress M a)?
(!BadNISTress M OA NIO*)?) (comma2 !(NI RA) &NI)?)

RA1 <- ((RA (!BadNISTress M a)?
(!BadNISTress M OA NIO*)?) (comma2 !(NI RA) &NI)?)

NI2 <- (((SA? (NI1+/RA1))/SA) NOI?
(CAO ((SA? (NI1+/RA1))/SA) NOI?)*)

NI <- ([ ]* NI2 (&(M UE) Acronym
(comma/&end/&period) !(C u))? (C u)?)

mex <- ([ ]* NI)

CI <- ([ ]* (C i))

Acronym <- ([ ]* ((M UE)/TAIO/(Z V2 !V2))
((comma &Acronym M UE)/NI1/TAIO/(Z V2 (!V2/(Z &V2))))+)

TAI <- ([ ]* (TAIO/((G AO) !V2 [ ]* (PreName/Predicate/CmapuaUnit))))

DAO <- ((T AO)/(T IO)/(T UA)/(M IO)/(M IU)/(M UO)/
(M JU)/(T OA)/(T OI)/(T OO)/(T OU)/(T UO)/(T UU)/
(S UO)/(H u)/(B a)/(B e)/(B o)/(B u)/(D a)/(D e)/(D i)/
(D o)/(D u)/(M i)/(T u)/(M u)/(T i)/(T a)/(M o))

DA1 <- ((TAIO/DAO) (C i ![ ] NIO)?)

DA <- ([ ]* DA1)

PAO <- ((N u !KOU)? ((G IA)/(G UA)/(P AU)/
(P IA)/(P UA)/(N IA)/(N UA)/(B IU)/(F EA)/
(F IA)/(F UA)/(V IA)/(V II)/(V IU)/(C OI)/
(D AU)/(D II)/(D UO)/(F OI)/(F UI)/(G AU)/
(H EA)/(K AU)/(K II)/(K UI)/(L IA)/(L UI)/
(M IA)/(N UI)/(P EU)/(R OI)/(R UI)/(S EA)/
(S IO)/(T IE)/(V a)/(V i)/(V u)/(P a)/(N a)/(F a)/
(V a)/(KOU !(N OI) !KI)) (N OI)?)

```

```
PANOPAUSES <- ((!PA0 NI)? (KOU2/PA0)+
  ((comma2? CA0 comma2?) (KOU2/PA0)+)* ZI?)

PA3 <- ([ ]* PANOPAUSES)

PA <- ((!PA0 NI)? (KOU2/PA0)+
  (((comma2? CA0 comma2?)/
  (comma2 !mod1a)) (KOU2/PA0)+)* ZI?)

PA2 <- ([ ]* PA)

GA <- ([ ]* (G a))

PA1 <- ((PA2/GA))

ZI <- ((Z i)/(Z a)/(Z u))

LE <- ([ ]* ((L EA)/(L EU)/(L OE)/(L EE)/(L AA)/(L e)/(L o)/(L a)))

LEFORPO <- ([ ]* ((L e)/(L o)/NI2))

LIO <- ([ ]* (L IO))

LAU <- ([ ]* (L AU))

LOU <- ([ ]* (L OU))

LUA <- ([ ]* (L UA))

LUO <- ([ ]* (L UO))

ZEIA <- ([ ]* Z EI a)

ZEIO <- ([ ]* Z EI o)

LI1 <- (L i)
```



```
LU1 <- (L u)

LI <- ([ ]* LI1 comma2? utterance0 comma2? LU1/
[ ]* LI1 comma2? ["\"] utterance0 ["\"] comma2? LU1)

LAO <- ([ ]* &([Ll] [Aa] [Oo]junction?) AlienWord)

LIE <- ([ ]* &([Ll] [Ii] juncture? [Ee]junction?) AlienWord)

LW <- Cmapua

LIU0 <- ((L IU)/(N IU))

# this now imposes the condition that an explicit comma pause
# (or terminal punctuation, or end) must appear at the end of the
# Word or PreName quoted with <liu>. This seems like a good idea, anyway.

LIU1 <- ([ ]* [Ll][iI] juncture? [Uu] juncture? !V1 comma2?
(PreName/Word) &(comma/terminal/end) /[ ]*(L II TAI ))

SUE <- ([ ]* &([Ss] [Uu] juncture? [Ee] juncture?/[
[Ss] [Aa] [Oo] juncture?) AlienWord)

CUI <- ([ ]* (C UI) )

GA2 <- ([ ]* (G a) )

GE <- ([ ]* (G e) )

GEU <- ([ ]* ((C UE)/(G EU)) )

GI <- ([ ]* ((G i)/(G OI)) )

GO <- ([ ]* (G o) )

GIO <- ([ ]* (G IO) )

GU <- ([ ]* (G u) )
```

GUIZA <- ([]* (G UI) (Z a))
GUIZI <- ([]* (G UI) (Z i))
GUIZU <- ([]* (G UI) (Z u))
GUI <- (!GUIZA !GUIZI !GUIZU ([]* (G UI)))
GUO <- ([]* (G UO))
GUOA <- ([]* (G UO Z? a))
GUOE <- ([]* (G UO e))
GUOI <- ([]* (G UO Z? i))
GUOO <- ([]* (G UO o))
GUOU <- ([]* (G UO Z? u))
GUU <- ([]* (G UU))
GUUA <- ([]* (G UU a))
GIUO <- ([]* (G IU o))
GUE <- ([]* (G UE))
GUEA <- ([]* (G UE a))
JE <- ([]* (J e))
JUE <- ([]* (J UE))
JIZA <- ([]* ((J IE)/(J AE)/(P e)/(J i)/(J a)/(N u J i)) (Z a))
JIOZA <- ([]* ((J IO)/(J AO)) (Z a))

```

JIZI <- ([ ]* ((J IE)/(J AE)/(P e)/(J i)/(J a)/(N u J i)) (Z i) )
JIOZI <- ([ ]* ((J IO)/(J AO)) (Z i) )
JIZU <- ([ ]* ((J IE)/(J AE)/(P e)/(J i)/(J a)/(N u J i)) (Z u) )
JIOZU <- ([ ]* ((J IO)/(J AO)) (Z u) )
JI <- (!JIZA !JIZI !JIZU ([ ]* ((J IE)/(J AE)/(P e)/(J i)/(J a)/(N u J i)) ))
JIO <- (!JIOZA !JIOZI !JIOZU ([ ]* ((J IO)/(J AO)) ))
DIO <- ([ ]* ((B EU)/(C AU)/(D IO)/(F OA)/(K AO)/(J UI)/
(N EU)/(P OU)/(G OA)/(S AU)/(V EU)/
(Z UA)/(Z UE)/(Z UI)/(Z UO)/(Z UU) )
LAE <- ([ ]* ((L AE)/(L UE)) )
ME <- ([ ]* ((M EA)/(M e)) )
MEU <- ([ ]* M EU )
NUO <- ((N UO)/(F UO)/(J UO)/(N u)/(F u)/(J u))
NU <- ([ ]* (NUO !([ ]+ (NIO/RA)) (NIO/RA)? freemod?)+ )
PO1 <- ([ ]* ((P o)/(P u)/(Z o)))
PO1A <- ([ ]* ((P OI a)/(P UI a)/(Z OI a)/(P o Z a)/(P u Z a)/(Z o Z a)))
PO1E <- ([ ]* ((P OI e)/(P UI e)/(Z OI e)))
PO1I <- ([ ]* ((P OI i)/(P UI i)/(Z OI i)/(P o Z i)/(P u Z i)/(Z o Z i)))
PO1O <- ([ ]* ((P OI o)/(P UI o)/(Z OI o)))
PO1U <- ([ ]* ((P OI u)/(P UI u)/(Z OI u)/(P o Z u)/(P u Z u)/(Z o Z u)))

```

```

POSHORT1 <- ([ ]* ((P OI)/(P UI)/(Z OI)))

PO <- ([ ]* P O1 )

POA <- ([ ]* P O1A )

POE <- ([ ]* P O1E )

POI <- ([ ]* P O1E )

POO <- ([ ]* P O1O )

POU <- ([ ]* P O1U )

POSHORT <- ([ ]* POSHORT1 )

DIE <- ([ ]* ((D IE)/(F IE)/(K AE)/(N UE)/(R IE)) )

HOI <- ([ ]* ((H OI)/(L OI)/(L OA)/(S IA)/(S IE)/(S IU)) )

JO <- ([ ]* (NIO/RA)? (J o) )

KIE <- ([ ]* (K IE) )

KIU <- ([ ]* (K IU) )

KIE2 <- (([ ]* [( ] (K IE) ))/([ ]* (K IE) [( ] ))

KIU2 <- (([ ]* (K IU) [( ] ))/([ ]* [( ] (K IU) ))

SOI <- ([ ]* (S OI) )

UIO <- (!predstart (!( [Ii] juncture? [Ee]) VV juncture?/
(B EA)/(B UO)/(C EA)/(C IA)/(C OA)/(D OU)/(F AE)/(F AO)/
(F EU)/(G EA)/(K UO)/(K UU)/(R EA)/(N AO)/(N IE)/(P AE)/
(P IU)/(S AA)/(S UI)/(T AA)/(T OE)/(V OI)/(Z OU)/((L OI))/
((L OA))/((S IA))/(S II)/(T OE)/((S IU))/(C AO)/

```

```
(C EU)/((S IE))/(S EU))

NOUI <- (([ ]* N [o] juncture? [ ]* UIO )/([ ]* UIO NOI ))

UI1 <- ([ ]* (UIO+/(NI F i)) )

HUE <- ([ ]* (H UE))

NO1 <- ([ ]* !KOU1 !NOUI (N o) !([ ]* KOU)
!([ ]* (JIO/JI/JIZA/JIOZA/JIZI/JIOZI/JIZU/JIOZU)) )

AcronymicName <- Acronym &(comma/period/end)

DJAN <- (PreName/AcronymicName)

BI <- ([ ]* (N u)? ((B IA)/(B IE)/
(C IE)/(C IO)/(B IA)/(B [i])) )

LWPREDA <- ((H e)/(D UA)/(D UI)/(B UA)/(B UI))

# here I should reinstall the <zao> proposal.

Predicate <- Complex

PREDA <- ([ ]* &caprule (Predicate/LWPREDA/(![ ] NI RA)) )

# Part 3: The Grammar Proper

guoa <- (PAUSE? (GUOA/GU) freemod?)

guoe <- (PAUSE? (GUOE/GU) freemod?)

guoi <- (PAUSE? (GUOI/GU) freemod?)

guoo <- (PAUSE? (GUOO/GU) freemod?)

guou <- (PAUSE? (GUOU/GU) freemod?)
```

```
guo <- (!guoa !guoi !guou (PAUSE? (GUO/GU) freemod?))
guiza <- (PAUSE? (GUIZA/GU) freemod?)
guizi <- (PAUSE? (GUIZI/GU) freemod?)
guizu <- (PAUSE? (GUIZU/GU) freemod?)
gui <- (PAUSE? (GUI/GU) freemod?)
gue <- (PAUSE? (GUE/GU) freemod?)
guea <- (PAUSE? (GUEA/GU) freemod?)
guu <- (PAUSE? (GUU/GU) freemod?)
guua <- (PAUSE? (GUUA/GU) freemod?)
giuo <- (PAUSE? (GIUO/GU) freemod?)
meu <- (PAUSE? (MEU/GU) freemod?)
geu <- GEU
gap <- (PAUSE? GU freemod?)
juelink <- (JUE freemod? (term/(PA2 freemod? gap?)))
links1 <- (juelink (freemod? juelink)* gue?)
links <- ((links1/(KA freemod? links freemod?
KI freemod? links1)) (freemod? A1 freemod? links1)*)
jelink <- (JE freemod? (term/(PA2 freemod? gap?)))
linkargs1 <- (jelink freemod? (links/gue)?)
linkargs <- ((linkargs1/(KA freemod? linkargs freemod?
```

```

KI freemod? linkargs1)) (freemod? A1 freemod? linkargs1)*)

abstractpred <- ((POA freemod? uttAx guoa?)/
(POA freemod? sentence guoa?)/(POE freemod? uttAx guoe?)/
(POE freemod? sentence guoe?)/(POI freemod? uttAx guoi?)/
(POI freemod? sentence guoi?)/(POO freemod? uttAx guoo?)/
(POO freemod? sentence guoo?)/(POU freemod? uttAx guou?)/
(POU freemod? sentence guou?)/(PO freemod? uttAx guo?)/
(PO freemod? sentence guo?))

predunit1 <- ((SUE/
(NU freemod? GE freemod? despredE (freemod? geu comma?))/
(NU freemod? PREDA)/
(comma? GE freemod? descpred (freemod? geu comma?))/
abstractpred/(ME freemod? argument1 meu?)/PREDA) freemod?)

predunit2 <- ((NO1 freemod?)* predunit1)

NO2 <- (!predunit2 NO1)

predunit3 <- ((predunit2 freemod? linkargs)/predunit2)

predunit <- ((POSHORT freemod?)? predunit3)

kekpredunit <- ((NO1 freemod?)*
KA freemod? predicate freemod? KI freemod? predicate guu?)

despredA <- ((predunit/kekpredunit)
(freemod? CI freemod? (predunit/kekpredunit))*)

despredB <- ((!PREDA CUI freemod? despredC freemod?
CA freemod? despredB)/despredA)

despredC <- (despredB (freemod? despredB)*)

despredD <- (despredB (freemod?
CA freemod? despredB)*)

```

```

despredE <- (despredD (freemod? despredD)*)

descpred <- ((despredE freemod?
GO freemod? descpred)/despredE)

sentpred <- ((despredE freemod?
GO freemod? barepred)/despredE)

mod1a <- (PA3 freemod? argument1 guua?)

mod1 <- ((PA3 freemod? argument1 guua?)/(PA2 freemod? !barepred gap?))

kekmod <- ((NO1 freemod?)* (KA freemod? modifier freemod? KI freemod? mod))

mod <- (mod1/((NO1 freemod?)* mod1)/kekmod)

modifier <- (mod (A1 freemod? mod)*)

# the serial name is a horrid heterogenous construction!

name <- (PreName/AcronymicName) (comma2? !FalseMarked PreName/
comma2? &([Cc] [Ii]) NameWord/
comma2? CI predunit !(comma2? (!FalseMarked PreName))/
comma2? CI AcronymicName)* freemod?

LANAME <- ([ ]* [Ll] [Aa] juncture? comma2? name)

#HOIO <- ([ ]* ((([Hh] OI)/([Ll] OI)/
([Ll] OA)/([Ss] IA)/([Ss] IE)/([Ss] IU))))

voc <- ([ ]* [Hh] [Oo] [Ii] juncture? comma2? name /
(HOI freemod? descpred guea? (comma2 &(!FalseMarked PreName/
[Cc][Ii] juncture? comma2 (PreName/AcronymicName)) name)?)/
(HOI freemod? argument1 guua?)/[ ]* &([Hh] [Oo] [Ii] juncture?) AlienWord)

descriptn <- (!LANAME ((LAU wordset1)/(LOU wordset2)/
(LE freemod? ((!mex arg1a freemod?)? (PA2 freemod?)?)? mex freemod? descpred)/
(LE freemod? ((!mex arg1a freemod?)? (PA2 freemod?)?)? mex freemod? arg1a)/

```



```
(GE freemod? mex freemod? descpred)/
(LE freemod? ((!mex arg1a freemod?)? (PA2 freemod?)?)? descpred)))
```

```
abstractn <- ((LEFORPO freemod? POA freemod? uttAx guoa?)/
(LEFORPO freemod? POA freemod? sentence guoa?)/
(LEFORPO freemod? POE freemod? uttAx guoe?)/
(LEFORPO freemod? POE freemod? sentence guoe?)/
(LEFORPO freemod? POI freemod? uttAx guoi?)/
(LEFORPO freemod? POI freemod? sentence guoi?)/
(LEFORPO freemod? POO freemod? uttAx guoo?)/
(LEFORPO freemod? POO freemod? sentence guoo?)/
(LEFORPO freemod? POU freemod? uttAx guou?)/
(LEFORPO freemod? POU freemod? sentence guou?)/
(LEFORPO freemod? PO freemod? uttAx guo?)/
(LEFORPO freemod? PO freemod? sentence guo?))
```

```
arg1 <- (abstractn/(LIO freemod? descpred guea?)/
(LIO freemod? argument1 guua?)/
(LIO freemod? mex gap?)/(LIO stringnospaces)/
LAO/LANAME/
(descriptn guua? (comma2 &(!FalseMarked PreName/
[Cc][Ii] juncture? comma2
(PreName/AcronymicName)) name?)/LIU1/LIE/LI)
```

```
arg1a <- ((DA/TAI/arg1/(GE freemod? arg1a)) freemod?)
```

```
argmod1 <- ((([ ]* (N o) [ ]*)? ((JI freemod? predicate)/
(JIO freemod? sentence)/(JIO freemod? uttAx)/
(JI freemod? modifier)/(JI freemod? argument1))))/
((( [ ]* (N o) [ ]*)? (((JIZA freemod? predicate) guiza?)/
((JIOZA freemod? sentence) guiza?)/
((JIOZA freemod? uttAx) guiza?)/
((JIZA freemod? modifier) guiza?)/
(JIZA freemod? argument1 guiza?)))/
((( [ ]* (N o) [ ]*)? ((JIZI freemod? predicate guizi?)/
(JIOZI freemod? sentence guizi?)/(JIOZI freemod? uttAx guizi?)/
(JIZI freemod? modifier guizi?)/(JIZI freemod? argument1 guizi?)))/
((( [ ]* (N o) [ ]*)? ((JIZU freemod? predicate guizu?)/
```

```
(JIOZU freemod? sentence guizu?)/(JIOZU freemod? uttAx guizu?)/
(JIZU freemod? modifier guizu?)/(JIZU freemod? argument1 guizu?)))

argmod <- (argmod1 (A1 freemod? argmod1)* gui?)

arg2 <- (arg1a freemod? argmod*)

arg3 <- (arg2/(mex freemod? arg2))

indef1 <- (mex freemod? descpred)

indef2 <- (indef1 guua? argmod*)

indefinite <- indef2

arg4 <- ((arg3/indefinite) (ZE2 freemod? (arg3/indefinite))*

arg5 <- (arg4/(KA freemod? argument1 freemod? KI freemod? argx))

argx <- ((NO1 freemod?)* (LAE freemod?)* arg5)

arg7 <- (argx freemod? (ACI freemod? argx)?)

arg8 <- (!GE (arg7 freemod? (A1 freemod? arg7)*))

argument1 <- (((arg8 freemod?
AGE freemod? argument1)/arg8) (GUU freemod? argmod)*

argument <- ((NO1 freemod?)* (DIO freemod?)* argument1)

argumentA <- argument

argumentB <- argument

argumentC <- argument

argumentD <- argument
```

```
argxx <- (&((N01 freemod?)* DIO) argument)

term <- (argument/modifier)

modifiers <- (modifier (freemod? modifier)*)

modifiersx <- ((modifier/argxx) (freemod? (modifier/argxx))*)

terms <- ((modifiersx? argumentA (freemod? modifiersx)?
argumentB? (freemod? modifiersx)?
argumentC? (freemod? modifiersx)?
argumentD?)/modifiersx)

word <- (arg1a/indef2)

words1 <- (word (ZEIA word)*)

words2 <- (word (ZEIO word)*)

wordset1 <- (words1? LUA)

wordset2 <- (words2? LU0)

termset1 <- (terms/(KA freemod? termset2 freemod? guu? KI freemod? termset1))

termset2 <- (termset1 (guu &A1)? (A1 freemod? termset1 (guu &A1)?)*)

termset <- ((terms freemod? G0 freemod? barepred)/termset2)

barepred <- (sentpred freemod? ((termset guu?)/(guu &termset))?)

markpred <- (PA1 freemod? barepred)

backpred1 <- ((N02 freemod?)* (barepred/markpred))

backpred <- (((backpred1 (ACI freemod? backpred1)+ freemod?
((termset guu?)/(guu &termset))?) ((ACI freemod? backpred)+ freemod?
((termset guu?)/(guu &termset))?)?)/backpred1)
```

```

predicate2 <- (!GE (((backpred (A1 !GE freemod? backpred)+ freemod?
  ((termset guu?)/(guu &termset)))? ((A1 freemod? predicate2)+ freemod?
  ((termset guu?)/(guu &termset)))?)/backpred))

predicate1 <- ((predicate2 AGE freemod? predicate1)/predicate2)

identpred <- ((NO1 freemod?)* (BI freemod? argument1 guu?))

predicate <- (predicate1/identpred)

subject <- ((modifiers freemod?)? ((argxx subject)/
  (argument (modifiersx freemod?)?)))

gasent1 <- ((NO1 freemod?)* (PA1 freemod? barepred (GA2 freemod? subject)?))

gasent2 <- ((NO1 freemod?)* (PA1 freemod? sentpred modifiers?
  (GA2 freemod? subject freemod? GIO? freemod? terms?)))

gasent <- (gasent2/gasent1)

statement <- (gasent/(modifiers freemod? gasent)/
  (subject freemod? (GIO freemod? terms)? predicate))

keksent <- ((NO1 freemod?)*
  ((KA freemod? sentence freemod? KI freemod? uttA1)/
  (KA sentence freemod? KI freemod? uttA1)/
  (KA freemod? headterms freemod? sentence freemod? KI freemod? uttA1)))

neghead <- ((NO1 freemod? gap)/(NO2 PAUSE))

sen1 <- ((neghead freemod?)*
  ((modifiers freemod? !gasent predicate)/
  statement/predicate/keksent))

sentence <- (sen1 (ICA freemod? sen1)*)

headterms <- (terms GI)+

```

```

uttAx <- (headterms freemod? sentence giuo?)

HUE0 <- ([ ]* ([Hh] UE))

invvoc <- ([ ]* &caprule [Hh] [Uu] juncture? [Ee] juncture? comma2? name/
HUE freemod? descpred guea?
(comma2 &(!FalseMarked PreName/
[Cc][Ii] juncture? comma2 (PreName/AcronymicName)) name)?/
(HUE freemod? statement giuo?)/
(HUE freemod? argument1 guu?)/
[ ]* &([Hh] [Uu] juncture? [Ee] juncture?) AlienWord)

freemod <- ((NOUI/(SOI freemod? descpred guea?)/
DIE/(NO1 DIE)/(KIE comma? utterance0 comma? KIU)/
(KIE2 comma? utterance0 comma? KIU2)/
invvoc/voc/(comma !(!FalseMarked PreName)))/
JO/UI1/([ ]* '...' ([ ]* &letter?)/([ ]* '--' ([ ]* &letter?)) freemod?)

uttA <- ((A1/mex) freemod?)

uttA1 <- ((sen1/uttAx/links/linkargs/argmod/
(modifiers freemod? keksent)/terms/uttA/NO1) freemod? period?)

uttC <- ((neghead freemod? uttC)/uttA1)

uttD <- ((sentence period? !ICI !ICA)/(uttC (ICI freemod? uttD)*))

uttE <- (uttD (ICA freemod? uttD)*)

uttF <- (uttE (I freemod? uttE)*)

utterance0 <- (!GE (!PAUSE freemod period? utterance0)/
(!PAUSE freemod period?)/(uttF IGE utterance0)/uttF/
(I freemod? uttF?)/(I freemod? period?)/
(ICA freemod? uttF)) (&I utterance0?)

utterance <- &(phoneticutterance !.)

```

```
(!GE ((!PAUSE freemod period? utterance)/
(!PAUSE freemod period? (&I utterance)? end)/
(uttF IGE utterance)/(I freemod? period? (&I utterance)? end)/
(uttF (&I utterance)? end)/(I freemod? uttF (&I utterance)? end)/
(ICA freemod? uttF (&I utterance)? end)))
```

11 The previous PEG test Grammar, fresh annotations completed

This is the text of the PEG grammar output by the ML version of the grammar in `loglantest.sml` in 9/4/2016, with annotations.

This parser is superseded by the new one implementing the Phonetics Proposal. The PEG for phonetics and orthography and shapes of words should be quite different; in the grammar section it should be generally similar. The comments may be valuable.

```
lowercase <- (!( [qwx] ) [a-z])
uppercase <- (!( [QWX] ) [A-Z])
letter <- (!( [QWXqwx] ) [A-Za-z])
```

Letters, excluding **qwx**, which it is proposed that we abandon. The elimination of these letters from the dictionary was well underway in the 1990's and was completed in 2015.

```
junction <- (([-] &(letter)) / ([\']* !(junction)))
stress <- ([\']* !(junction))
junction2 <- ((([-] &(letter)) / ([\']* !((( [ ])*
&(C1) Predicate))
((', ' ([ ])* &(C1) &(Predicate)))?)) !(junction))
```

The syllable separator - (hyphen) and the ordinary stress ' and emphatic stress *. The grammar treats the ordinary and emphatic stress in exactly the same way, but NB3 says the difference between them is phonemic, so we provide them.

The rule `junction2` enforces the rule that a stressed `cmapua` (structure word) syllable followed by a predicate must be separated from it by a comma-marked pause. This rule goes back to the beginning of the language, but has no actual manifestation in LIP because LIP does not represent stress.

```
Lowercase <- (lowercase / (junction (letter)?))
```

```
Letter <- (letter / junction)
```

The rule `Lowercase` reads a lower case letter or a junction (syllable separator, hyphen or stress) optionally followed by a letter which can be of either case. This is useful in building the capitalization rule.

The rule `Letter` reads a letter or junction indifferently.

```
comma <- ([,] ([ ])+ &(letter))
```

```
comma2 <- (([,])? ([ ])+ &(letter) &caprule)
```

`comma` is a comma, intended to represent an explicit pause in speech, which must be followed by at least one space then a letter. `comma2` is the same construction but with the actual comma optional (it may be just whitespace followed by a letter) In addition (4/28/17) the capitalization rule propagates through `comma2`, since it is intended to be used for internal pauses allowed inside instances of some word classes.

```
end <- ((([ ])* '#' ([ ])+ utterance) /  
  (([ ])+ !(.)) / !(.)
```

This is the end of a Loglan utterance. It is either optional whitespace followed by `#` followed by a new utterance (attested in old Loglan sources), or optional whitespace followed by end of text.


```
period <- (([!.:;?] (&(end) / (([ ]+ &(letter))))
  (((invvoc) (period)?))?)
```

This is terminal punctuation. This is an exclamation point, period, colon, semicolon or question mark, followed either by end of utterance or by whitespace followed by a letter; it can in addition consume a following inverse vocative construction and a further **period** (the ability to do this last maneuver was very important in parsing the Visit to Loglandia).

changed 11/4/2016 to explicitly use the class **invvoc** of inverse vocatives instead of a slightly buggy indirect description of this class.

```
V1 <- [AEIOUYaeiouy]
```

```
V2 <- [AEIOUaeiou]
```

```
C1 <- (!(V1) letter)
```

Classes of letters, respectively vowels, regular vowels (excluding **y**) and consonants.

```
Mono <- ((([Aa] [o])) /
  ((V2 [i]) !([i])) /
  ([Ii] ![i] V2) / ([Uu] V2))
```

```
EMono <- ((([Aa] [o])) /
  (([AEOaeo] [i]) !([i])))
```

Vowel diphthongs (monosyllabic vowel pairs). The first class contains the pairs which **may** be monosyllable; the second contains the pairs which **must** be monosyllabic.

The monosyllables ending in **i** cannot be followed by an instance of the same letter. This does not make such vowel sequences illegal; it changes their grouping.

```
NextVowels <- (EMono / (V2 &(EMono)) / Mono / V2)
```

This rule applied repeatedly resolves long streams of vowels in predicates or names. Long streams of vowels in cmapua are grouped lockstep in pairs: this only happens in compound attitudinals.

By preference choose an exclusive monosyllable; if this cannot be read, take a single vowel if it is followed by an exclusive monosyllable, and if this cannot be read take an optional monosyllable, then as the last possible choice take a single vowel. (Then repeat; this rule describes a single step).

This is not the same as the rules presented in earlier sources, but it works.

```
BrokenMono <- (([a] juncture [o]) / ([aeo] juncture [i]))
```

This rule describes a mandatory monosyllable broken by an explicit syllable break.

```
CVVSyll <- (C1 Mono)
```

```
LWunit <- (((CVVSyll (juncture)? V2) /
  (C1 !(BrokenMono) V2 (juncture)? V2) /
  (C1 V2)) (juncture2)?)
```

This block describes the units from which cmapua are built. CVVSyll is a CVV monosyllable.

LWunit contains the things from which multisyllable cmapua are made according to NB3. These are Cvv-V units, CV-V units where the V-V does not break a mandatory monosyllable (and where the syllable break may optionally be explicitly expressed), and CV syllables. These are followed by a `juncture2` to enforce the rule about finally stressed cmapua before predicates. This is **only** used after the **liu** word quotation article. The parsing of cmapua words is handled entirely by the grammar.

```
caprule <- [\"(\)])? ((uppercase / lowercase) ((('z' V1) /
lowercase / (juncture (caprule)? / TAI0))* !(letter))
```

The capitalization convention. In an unbroken string of letters and junctures, one capitalizes only initial letters, letters immediately after junctures, vowels after z (useful in acronyms) or vowels initial in occurrences of letter names (useful in acronyms and also to implement a commonly used convention in phrases like **leSai**).

12/2/17 update allows strings to start with a double quote or a parenthesis as long as the rest of the string satisfies the condition. This supports punctuation of quoted strings and parenthesis enclosed strings.

Things like **leAma** attested in the sources (with legacy VCV letterals) are no longer words, but **le Ama** will work.

```
InitialCC <- ('bl' / 'br' / 'ck' / 'cl' /
'cm' / 'cn' / 'cp' / 'cr' / 'ct' / 'dj' /
'dr' / 'dz' / 'fl' / 'fr' / 'gl' / 'gr' / 'jm' /
'kl' / 'kr' / 'mr' / 'pl' / 'pr' / 'sk' / 'sl' /
'sm' / 'sn' / 'sp' / 'sr' / 'st' / 'tc' / 'tr' /
'ts' / 'vl' / 'vr' / 'zb' / 'zv' / 'zl' / 'sv' /
'Bl' / 'Br' / 'Ck' / 'Cl' / 'Cm' / 'Cn' /
' Cp' / 'Cr' / 'Ct' / 'Dj' / 'Dr' / 'Dz' /
'Fl' / 'Fr' / 'Gl' / 'Gr' / 'Jm' / 'Kl' / 'Kr' /
'Mr' / 'Pl' / 'Pr' / 'Sk' / 'Sl' / 'Sm' / 'Sn' /
'Sp' / 'Sr' / 'St' / 'Tc' / 'Tr' / 'Ts' / 'Vl' /
'Vr' / 'Zb' / 'Zv' / 'Zl' / 'Sv')
```

```
MaybeInitialCC <- (([Bb] (juncture)? [l]) /
([Bb] (juncture)? [r]) / ([Cc] (juncture)? [k]) /
([Cc] (juncture)? [l]) / ([Cc] (juncture)? [m]) /
([Cc] (juncture)? [n]) / ([Cc] (juncture)? [p]) /
([Cc] (juncture)? [r]) / ([Cc] (juncture)? [t]) /
([Dd] (juncture)? [j]) / ([Dd] (juncture)? [r]) /
([Dd] (juncture)? [z]) / ([Ff] (juncture)? [l]) /
```

```

([Ff] (junction)? [r]) / ([Gg] (junction)? [l]) /
([Gg] (junction)? [r]) / ([Jj] (junction)? [m]) /
([Kk] (junction)? [l]) / ([Kk] (junction)? [r]) /
([Mm] (junction)? [r]) / ([Pp] (junction)? [l]) /
([Pp] (junction)? [r]) / ([Ss] (junction)? [k]) /
([Ss] (junction)? [l]) / ([Ss] (junction)? [m]) /
([Ss] (junction)? [n]) / ([Ss] (junction)? [p]) /
([Ss] (junction)? [r]) / ([Ss] (junction)? [t]) /
([Tt] (junction)? [c]) / ([Tt] (junction)? [r]) /
([Tt] (junction)? [s]) / ([Vv] (junction)? [l]) /
([Vv] (junction)? [r]) / ([Zz] (junction)? [b]) /
([Zz] (junction)? [v]) / ([Zz] (junction)? [l]) / ([
Ss] (junction)? [v]))

```

The pairs of consonants which may begin a Loglan syllable, and the same class of pairs of consonants possibly broken by a juncture. The pairs **sv** and **zl** had accidentally been omitted.

```

NonmedialCC <- (([b] (junction)? [b]) /
([c] (junction)? [c]) / ([d] (junction)? [d]) /
([f] (junction)? [f]) / ([g] (junction)? [g]) /
([h] (junction)? [h]) / ([j] (junction)? [j]) /
([k] (junction)? [k]) / ([l] (junction)? [l]) /
([m] (junction)? [m]) / ([n] (junction)? [n]) /
([p] (junction)? [p]) / ([q] (junction)? [q]) /
([r] (junction)? [r]) / ([s] (junction)? [s]) /
([t] (junction)? [t]) / ([v] (junction)? [v]) /
([z] (junction)? [z]) / ([h] (junction)? C1) /
([cjsz] (junction)? [cjsz]) / ([f] (junction)? [v]) /
([k] (junction)? [g]) / ([p] (junction)? [b]) /
([t] (junction)? [d]) / ([fkpt] (junction)? [jz]) /
([b] (junction)? [j]) / ([s] (junction)? [b]))

NonjointCCC <- (([c] (junction)? [d] (junction)? [z]) /
([c] (junction)? [v] (junction)? [l]) /
([n] (junction)? [d] (junction)? [j]) /

```

```

([n] (junction)? [d] (junction)? [z]) /
([d] (junction)? [c] (junction)? [m]) /
  ([d] (junction)? [c] (junction)? [t]) /
  ([d] (junction)? [t] (junction)? [s]) /
([p] (junction)? [d] (junction)? [z]) /
([g] (junction)? [t] (junction)? [s]) /
  ([g] (junction)? [z] (junction)? [b]) /
([s] (junction)? [v] (junction)? [l]) /
([j] (junction)? [d] (junction)? [j]) /
([j] (junction)? [t] (junction)? [c]) /
  ([j] (junction)? [t] (junction)? [s]) /
([j] (junction)? [v] (junction)? [r]) /
([t] (junction)? [v] (junction)? [l]) /
([k] (junction)? [d] (junction)? [z]) /
([v] (junction)? [t] (junction)? [s]) /
([m] (junction)? [z] (junction)? [b]))

```

Pairs and triples of consonants which may not occur, even across a syllable boundary. It is worth noting specifically that these include all pairs of doubled consonants. Doubled continuants **mnlr** can occur as syllabic consonants, though they are excluded by this particular rule. No other doubled consonants occur in Loglan except in alien text.

```

Oddvowel <- ((junction)? (((V2 (junction)?
  V2 (junction)?))* V2) (junction)?)

```

This rule detects a sequence of vowels of odd length. This is useful in defending the vowel structure of *cmapua*.

```

RepeatedVowel <- ((([Aa] (junction)? [a]) /
  ([Ee] (junction)? [e]) / ([Oo] (junction)? [o]) /
  ([Ii] junction [i]) / ([Uu] junction [u]))

```

Doubled vowels one of which must be stressed. Note that this rule applies to doubled **i** or **u** only if an explicit juncture is present: a stress is forced in these cases only if the disyllabic pronunciation of these optional monosyllabic vowel pairs is used.

```
RepeatedVocalic <- (([Mm] [m]) / ([Nn] [n]) /
([Ll] [l]) / ([Rr] [r]))
```

```
Syllabic <- [lmnr]
```

```
Nonsyllabic <- (!(Syllabic) C1)
```

The first class captures syllabic consonant pairs. The other classes capture single continuant or non-continuant consonants.

```
Badfinalpair <- (Nonsyllabic !('mr') !(RepeatedVocalic)
Syllabic !((V2 / [y] / RepeatedVocalic)))
```

This rule enforces the condition that a pair of consonants final in a syllable cannot be a continuant followed by a non-continuant. This is a new rule but entirely unexceptionable: such a final consonant pair would be forced to be pronounced as another syllable. The various additional conditions capture conditions under which one can tell that the first consonant is not at the beginning of the final pair of consonants in a syllable, due to either being initial in a syllable or immediately followed by a syllable break.

```
FirstConsonants <- (((!((C1 C1 RepeatedVocalic))
&(InitialCC) (C1 InitialCC)) /
(!((C1 RepeatedVocalic)) InitialCC) /
(!((RepeatedVocalic) C1) !([y]))) !(juncture))
```

```
FirstConsonants2 <- (((!((C1 C1 RepeatedVocalic))
&(InitialCC) (C1 InitialCC)) /
(!((C1 RepeatedVocalic)) InitialCC) /
```

```
(!(RepeatedVocalic) C1)) !(junction))
```

The initial consonant group of a syllable, in two flavors, the first for predicates and the second for names.

This can be a single consonant, a permissible initial pair, or a triple in which each adjacent pair is a permissible initial.

The initial consonant group cannot overlap with a syllabic consonant pair.

In a predicate, the initial consonant group cannot be followed by y.

```
VowelSegment<-(!(BrokenMono (NextVowels !RepeatedVocalic))/
!(C1 RepeatedVocalic) RepeatedVocalic))
```

```
VowelSegment2 <- (NextVowels /
!((C1 RepeatedVocalic)) RepeatedVocalic))
```

The vocalic segment of a syllable, again in a flavor for predicates and a flavor for names. This can be either the vowel or pair of vowels selected by the `NextVowels` rule above or a syllabic consonant.

In a predicate, a vowel segment cannot be followed by a syllabic consonant.

A syllabic consonant cannot be followed by another occurrence of the same consonant.

```
SyllableA <- ((C1 V2 &(C1) !(Badfinalpair)
(FinalConsonant)? (!(Syllable) FinalConsonant)))? (junction)?
```

```
SyllableB <- ((FirstConsonants)? !(RepeatedVowel)
!((&(Mono) V2 RepeatedVowel)) VowelSegment !(Badfinalpair)
(!(Syllable) FinalConsonant)))?
(!(Syllable) FinalConsonant)))? (junction)?
```

```
Syllable <- (SyllableA / SyllableB)
```

Classes of Loglan syllable which occur in borrowed predicates.

A syllable consists of an optional initial consonant group, followed by a mandatory vowel segment, followed optionally by one or two consonants. Its vowel segment will not be *aa*, *ee*, or *oo* or overlap with any repeated vowel. The final pair of consonants (if there are two) will not be a continuant followed by a noncontinuant. Neither of the final consonants will be initial in a well-formed Loglan syllable, except that a syllable of this class beginning CV will always pick up at least one following consonant if there is one it is allowed to pick up (this is the point of the distinction between *SyllableA* and *SyllableB*: *SyllableA* begins CV, is followed by a consonant, and may take that consonant as a final consonant even if it starts a syllable).

A syllable always includes an optional following juncture.

```
BrokenInitialCC <- (&(MaybeInitialCC) C1 juncture C1 &(V2))
```

```
JunctureFix <- ((InitialCC V2 BrokenInitialCC) /
  (((C1 V2))? V2 BrokenInitialCC) /
  (C1 V2 !(stress) juncture InitialCC V2 Letter) /
  (C1 BrokenInitialCC V2))
```

This rule describes certain conditions where an explicit juncture occurs which are forbidden in borrowed predicates.

The purpose of this rule is to make it impossible to explicitly articulate a borrowed predicate into syllables in a way which would result from moving junctures in a candidate complex predicate in a way which did not respect djifoa boundaries. This prevents illegal complex predicates from being parsed as borrowings. There is a full analysis in the reference grammar.

Pronouncing borrowed predicates in a way which violates this rule is not a problem: the purpose of the rule is orthographic.

```
SyllableFinal1 <- ((FirstConsonants)? !(RepeatedVocalic)
VowelSegment !(stress) (juncture)? !(V2) (&(Syllable) /
  &([y]) / !(Letter)))
```

```
SyllableFinal2 <- ((FirstConsonants)? !(RepeatedVocalic)
```



```
VowelSegment !(stress) (juncture)? (&[y]) / !(Letter))
```

```
SyllableFinal2a<-(FirstConsonants? !RepeatedVocalic  
VowelSegment juncture? &[y])
```

```
SyllableFinal2b<-(FirstConsonants? !RepeatedVocalic  
VowelSegment stress &[y])
```

An assortment of possible final syllables for borrowed predicates or borrowing djifoa. A final syllable of a borrowing predicate has to be vowel-final and unstressed. `SyllableFinal2` must be a final syllable; `SyllableFinal1` is not necessarily one if what follows it is a syllable, but it is a candidate.

`SyllableFinal2a` and `SyllableFinal2b` can be final syllables of borrowing djifoa (these can be stressed).

```
StressedSyllable <- (((FirstConsonants)? !(RepeatedVowel)  
!((&(Mono) V2 RepeatedVowel)) VowelSegment !(Badfinalpair)  
(FinalConsonant)? (FinalConsonant)?) stress)
```

An explicitly stressed syllable in a borrowing predicate.

```
FinalConsonant <- (!(RepeatedVocalic) !(NonmedialCC)  
!(NonjointCCC) C1 !(((juncture)? V2)))
```

One of the two consonants final in a syllable . It cannot start a forbidden consonant sequence (even one extending into the next syllable) nor can it be part of a syllabic consonant pair. It will not be followed by an explicit juncture then a vowel: a vowel-initial syllable in a predicate or name always follows a vowel.

```
Syllable2 <- (((FirstConsonants2)? (VowelSegment2 / [y])  
!(Badfinalpair) (!(Syllable2) FinalConsonant))?  
(!(Syllable2) FinalConsonant)?) (juncture)?
```

The syllable in the form appropriate for names, the most general form of Loglan syllable. The vowel segment may be *y*. The vowel segment may be followed by a syllabic consonant (in the next syllable, of course). Neither of the final consonants may start a well-formed name syllable: syllables end as soon as possible. You may recall that there is an exception to this rule in borrowings.

The final consonant in the correct form for names.

```
Name <- (([ ])* &(((uppercase / lowercase)
  (((!(C1 (stress)? !(Letter))) Lowercase))* C1 (stress)?
  !(Letter) (&(end) / comma / &(period) / &(Name) / &(CI))))
  ((Syllable2)+ (&(end) / comma / &(period) / &(Name) / &(CI))))
```

The class of name words. These resolve into syllables without any stress requirements, and are always consonant-final (the consonant may be followed by a final stress). A name word must either end in an explicit comma pause (included in the name) or be followed by the end of the utterance, terminal punctuation, another name word, or the *cmapua ci*: these things are not included in the name word, but allow it not to end with a comma pause.

```
CCSyllableB <- (((FirstConsonants)? RepeatedVocalic
  !(Badfinalpair) ((!(Syllable) FinalConsonant)))?
  ((!(Syllable) FinalConsonant)))? (juncture)?
```

This is the form of a syllable in a borrowed predicate with a syllabic consonant as its vowel segment.

```
BorrowingTail <- ((!(JunctureFix) !(CCSyllableB)
  StressedSyllable ((!(StressedSyllable) CCSyllableB)))?
  !(StressedSyllable) SyllableFinal1) / ((!(CCSyllableB)
  !(JunctureFix) Syllable ((!(StressedSyllable) CCSyllableB)))?
  !(StressedSyllable) SyllableFinal2))
```

This describes the last two or three syllables of a borrowing (called a borrowing tail). It consists of an explicitly stressed syllable not containing a syllabic consonant followed optionally by an unstressed syllabic consonant syllable followed by a required `SyllableFinal1` (since the stress is explicitly shown the first syllable of the next word can follow immediately), or of a non-syllabic-consonant syllable followed by an optional unstressed syllabic consonant syllable followed by a `SyllableFinal2` (because the stress is not explicitly marked the word must end in whitespace or punctuation to signal that the stress is there).

```
PreBorrowing <- (((!(BorrowingTail) !(StressedSyllable)
!(JunctureFix) !((CCSyllableB CCSyllableB)) Syllable))*
!(CCSyllableB) BorrowingTail)
```

A pre-borrowing is a sequence of syllables none of which start a borrowing tail, or are explicitly stressed, or form a sequence of two syllabic consonant syllables, followed by a borrowing tail.

```
HasCCPair <- (((C1)? ((V2 (!(stress) juncture)))?)+
!(Borrowing) !(&(MaybeInitialCC) C1
(!(stress) juncture) !(CCVV) PreBorrowing)) (stress)?)?
C1 (juncture)? C1)
```

```
CVCBreak <- (C1 V2 (juncture)? &(MaybeInitialCC)
C1 (juncture)? &((PreComplex / ComplexTail)))
```

```
CCVV <- (&(BorrowingTail) C1 C1 (C1)?
V2 stress !(Mono) V2) / (&(BorrowingTail)
C1 C1 (C1)? V2 (juncture)? V2 (!(Letter) / ((juncture)? [y])))
```

```
Borrowing <- (&(HasCCPair) !(CVCBreak) !(CCVV)
!((((C1)? (V2 (juncture)?) ((V2 (juncture)?
&(V2)))? V2 (juncture)?
MaybeInitialCC V2)) !(CCSyllableB)
((!(BorrowingTail) !(StressedSyllable)
```

```
!((CCSyllableB CCSyllableB)) !(JunctureFix) Syllable))*
!(CCSyllableB) BorrowingTail))
```

A borrowed predicate is a pre-borrowing that satisfies some additional conditions. It must contain a CC pair, which is either initial or preceded by a consonant followed by a sequence of vowels which cannot be read as a mapua: so the CC pair cannot start a well-formed borrowing (even after deletion of an explicit juncture between the consonants in the pair). The details of `HasCCPair` and `CVCBreak` and details of the prefix to the borrowing class itself have to do with preventing $C(V)^n$ from falling off the front of the predicate. The shapes `CCVV` and `CCCVV` for a borrowing predicate are forbidden.

There is more discussion of these rules in the reference grammar.

```
PreBorrowingAffix <- (((!(StressedSyllable)
  !(SyllableFinal2a) !((CCSyllableB CCSyllableB))
!(JunctureFix) Syllable))+ SyllableFinal2a) (juncture)?
[y] !(stress) (juncture)? ([ ,] ([ ])*))?)

BorrowingAffix <- (&(HasCCPair) !(CVCBreak) !(CCVV)
!((((C1)? (V2 (juncture)? ((V2 (juncture)? &(V2))))+))?)
V2 (juncture)? MaybeInitialCC V2)) !((CCSyllableB
  (((!(StressedSyllable) !(SyllableFinal2a) !((CCSyllableB CCSyllableB))
  !(JunctureFix) Syllable))+ SyllableFinal2a)
(juncture)? [y] !(stress) (juncture)? (comma)?)

StressedBorrowingAffix <- (&(HasCCPair) !(CVCBreak)
  !(CCVV) !((((C1)? (V2 (juncture)? ((V2 (juncture)?
&(V2))))+))?) V2 (juncture)? MaybeInitialCC V2)) !((CCSyllableB
  (((!(StressedSyllable) !(SyllableFinal2a) !((CCSyllableB CCSyllableB))
  !(JunctureFix) Syllable))* SyllableFinal2b) (juncture)?
  [y] !(stress) (juncture)? ([,]))
```

A borrowing djifoa is obtained by appending `y` to a borrowed predicate and moving the stress to the last syllable of the borrowed predicate. The solution for parsing this is very similar to the solution for borrowings.

The last class is the explicitly stressed borrowing djifoa.

```
yhyphen <- ((juncture)? [y] !(stress) (juncture)? !([y]) &(letter))
```

This is the y which can be appended to a djifoa to insulate it from a following djifoa under some circumstances. One thing it is not is a literal hyphen: we do not support writing it in this way as was suggested in NB3, as the hyphen has a different use as the syllable separator.

It is an unstressed y, optionally set off on one or both sides by junctures, followed by a letter but not by y.

```
CV <- (C1 V2 !(stress) (juncture)? !(V2))
```

The final CV syllable of a five letter predicate.

```
Cfinal<-(((juncture &(C1 !juncture))? C1 yhyphen)/
(!NonmedialCC !NonjointCCC C1 !(juncture? V2)))
```

The final consonant of a CVC djifoa (which may incorporate a y hyphen and may not be followed by a regular vowel). 4/27 fix allows a syllable break initial to this form.

```
hyphen <- (!(NonmedialCC) !(NonjointCCC)
([[r] !(((juncture)? [r])) !(((juncture)? V2))) /
([n] (juncture)? &([r])) /
((juncture)? [y] !(stress))) ((juncture)? &(letter))
!(((juncture)? [y])))
```

```
noyhyphen <- (!(NonmedialCC) !(NonjointCCC)
([[r] !(((juncture)? [r])) !(((juncture)? V2))) /
([n] (juncture)? &([r])) &(((juncture)? &(letter)))
!(((juncture)? [y])))
```

More phonetic hyphenation. A CVV djifoa may be glued to following djifoa by a following **r** (if not followed by an **r**) or **n** (if followed by **r**) or a **y** hyphen as above.

The second class excludes **y** hyphens.

An item of either of these classes may incorporate a following juncture and will be followed by a letter other than **y**.

```
StressedSyllable2 <- (((FirstConsonants)? VowelSegment
  !(Badfinalpair) (FinalConsonant)? (FinalConsonant)?) stress (yhyphen)?)
```

This is a very general form of a stressed syllable used for tests.

```
CVVStressed <- (((C1 &(RepeatedVowel) V2 !(stress) (juncture)?
  !(RepeatedVowel) V2 (nohyphen)?) (juncture)? (yhyphen)?) /
  (C1 !(BrokenMono) V2 !(stress) juncture V2
  (nohyphen)? stress (yhyphen)?) /
  (C1 !(Mono) V2 V2 (nohyphen)? stress (yhyphen)?))
```

```
CVVStressed2 <-
  (C1 Mono (nohyphen)? stress (yhyphen)?)
```

```
CVV <- (!(((C1 V2 stress V2 (hyphen)? stress))
  ((C1 !(BrokenMono) V2 (juncture)?
  !(RepeatedVowel) V2 (nohyphen)?)
  (juncture)? !(V2) (yhyphen)?))
```

CVV djifoa. The first describes CVV syllables which are or may be disyllables with final stress, including those with doubled vowels that force stress.

The second is a stressed CVV monosyllable.

The third is a completely general CVV djifoa. It will not be a disyllable with both components explicitly stressed. It will not have a monosyllable broken by an explicit juncture. It will not be followed by a regular vowel. It may be phonetically hyphenated in any of the three ways described.

The rule `nohyphen` is used here because the consonantal phonetic hyphens would appear before an explicit syllable juncture and the y hyphen would appear after such a juncture.

```
CVVFinal1 <- (C1 !(BrokenMono) V2 stress
!(RepeatedVowel) V2 !(stress) (juncture)? !(V2))
```

```
CVVFinal2 <- (((C1 !(Mono) V2 V2) /
(C1 !(BrokenMono) V2 juncture
!(RepeatedVowel) V2)) !(Letter))
```

```
CVVFinal3 <- (C1 &(Mono) V2 V2
!(stress) (juncture)? !(V2))
```

```
CVVFinal4 <- (C1 Mono !(Letter))
```

```
CVVFinal5 <- (((C1 !(Mono) V2 V2) /
(C1 !(BrokenMono) V2 juncture V2))
&(((juncture)? [y])))
```

CVV djifoa which are or might be final in a complex. The first is a medially stressed disyllable, definitely final. The second is definitely final because it is followed by a non-letter. `CVVFinal5` is followed by y and has a technical use. `CVVFinal3` is a possibly final CVV monosyllable (not stressed). `CVVFinal4` is a definitely final CVV monosyllable (followed by a non-letter).

```
CVC <- ((C1 V2 Cfinal) (juncture)?)
```

```
CVCStressed<-((C1 V2
!NonmedialCC !NonjointCCC C1 stress !V2 yhyphen?)/
(C1 V2 stress C1 !juncture yhyphen))
```

CVC djifoa, general and explicitly stressed. Of course there are no final CVC forms.

```
CCV <- (InitialCC !(RepeatedVowel) V2 (juncture)? !(V2) (yhyphen)?)
```

```
CCVStressed <- (InitialCC !(RepeatedVowel) V2 stress !(V2) (yhyphen)?)
```

The general CCV djifoa and explicitly stressed CCV djifoa. These can be followed by a y hyphen.

```
CCVFinal1 <- (InitialCC !(RepeatedVowel) V2 !(stress) (juncture)? !(V2))
```

```
CCVFinal2 <- (InitialCC V2 !(Letter))
```

possibly final (because not explicitly stressed) and definitely final (because followed by a non-letter) CCV djifoa.

```
CCVCMedial <- (InitialCC V2 (juncture)?  
C1 [y] !(stress) (juncture)? &(letter))
```

```
CCVCMedialStressed <- (CCV stress  
C1 [y] !(stress) (juncture)? &(letter))
```

```
CCVCFinal1 <- (InitialCC V2 stress CV)
```

```
CCVCFinal2 <- (InitialCC V2 (juncture)?  
CV !(Letter))
```

```
CCVCVY <- (InitialCC V2 (juncture)? CV [y])
```

Forms of the CCVCV five letter djifoa. The medial form has the final vowel suppressed in favor of y. The stressed form is needed because the penultimate stress in a predicate cannot fall on the Cy ending of a medial CCVCV.

Forms with vowels are definitely final: there is a stressed form and a form followed by a non-letter. Finally, there is a form followed by a y hyphen, whose use will be revealed.


```

CVCCVMedial <- (C1 V2
((juncture &(InitialCC)))? !(NonmedialCC)
C1 (juncture)? C1 [y] !(stress) (juncture)? &(letter))

CVCCVMedialStressed <- ((C1 V2 (stress &(InitialCC))
!(NonmedialCC) C1 C1 [y] !(stress) (juncture)? &(letter)) /
(C1 V2 !(NonmedialCC) C1 stress C1 [y] !(stress) (juncture)? &(letter)))

CVCCVFinal1a <- (C1 V2 stress InitialCC V2 !(stress) (juncture)? !(V2))

CVCCVYa <- (C1 V2 (juncture)? InitialCC V2 !(stress) (juncture)? [y])

CVCCVFinal1b <- (C1 V2 !(NonmedialCC) C1 stress CV)

CVCCVYb <- (C1 V2 !(NonmedialCC) C1 (juncture)? CV [y])

CVCCVFinal2 <- (C1 V2 ((juncture &(InitialCC)))?
!(NonmedialCC) C1 (juncture)? CV !(Letter))

```

Forms of the five letter CVCCV djifoa. As above, the medial forms have the final vowel suppressed in favor of *y*. There are two possible placements of the internal syllable juncture, CVC-CV and CV-CCV (of course in the latter case the CC must be a permissible initial pair). This leads to more complex forms and more cases. In general this is similar to the previous block in intent.

```

FiveLetterY <- (CCVCVY /
CVCCVYa / CVCCVYb)

GenericFinal <- (CVVFinal3 / CVVFinal4 /
CCVFinal1 / CCVFinal2)

GenericTerminalFinal <- (CVVFinal4 / CCVFinal2)

FiveLetterFinal <- (CCVCVFinal1 / CCVCVFinal2 /

```

```
CVCCVFinal1a / CVCCVFinal1b / CVCCVFinal2)
```

Convenient classes of final forms. The last consists of the forms which definitely end because followed by a non-letter.

```
Affix1 <- (CCVCVMedial / CVCCVMedial / CCV / CVV / CVC)
```

Non-borrowing djifoa.

```
Peelable <- (&(PreBorrowingAffix) !(CVVFinal1)
  !(CVVFinal5) Affix1 (!(Affix1) /
&((&(PreBorrowingAffix) !(CVVFinal1)
  !(CVVFinal5) Affix1 !(PreBorrowingAffix) !(Affix1)))) / Peelable))
```

```
Peelable2 <- (&(PreBorrowing) !(CVVFinal1)
  !(CVVFinal2) !(CVVFinal5) !(FiveLetterFinal)
Affix1 !(FiveLetterFinal) (!(Affix1) /
&((&(PreBorrowing) !(FiveLetterFinal)
  !(CVVFinal1) !(CVVFinal2) !(CVVFinal5)
Affix1 !(PreBorrowing) !(FiveLetterFinal)
  !(Affix1)))) / Peelable2))
```

`Peelable` and `Peelable2` are inhabited by apparent non-borrowing djifoa which are actually initial segments of (pre-) borrowing djifoa. This is an evilly recursive application of PEG logic.

`Peelable` is peeled off of a non-terminal borrowing affix, and `Peelable2` off an actual (pre-) borrowing appearing as a final component of a complex.

```
Affix <- ((!(Peelable) !(Peelable2) Affix1) / (!(FiveLetterY) BorrowingAffix))
```

Djifoa, excluding the fake djifoa which peel off the fronts of borrowing djifoa.

```
Affix2 <- (!(StressedSyllable2) !(CVVStressed) Affix)
```

djifoa without explicit stress.

```
ComplexTail <- ((Affix GenericTerminalFinal) /
(!(Peelable) Affix1)) !(FiveLetterY)
StressedBorrowingAffix GenericFinal) /
(CVCVMedialStressed GenericFinal) /
(CVCCVMedialStressed GenericFinal) /
(CCVStressed GenericFinal) /
(CVCStressed GenericFinal) /
(CVVStressed GenericFinal) / (CVVStressed2 GenericFinal)
/ (Affix2 CVVFinal1) / (Affix2 CVVFinal2) /
CCVCVFinal1 / CCVCVFinal2 / CVCCVFinal1a /
CVCCVFinal1b / CVCCVFinal2 / (!(CVVStressed /
StressedSyllable2)) Affix !(Peelable2) Affix1))
Borrowing !(((juncture)? [y]))))
```

The last djifoa or two of a complex containing the stress. There is a long story here.

The purpose of `FiveLetterY` is to prevent the formation of borrowing affixes from predicates with the primitive five letter forms.

```
Primitive <- (CCVCVFinal1 / CCVCVFinal2 /
CVCCVFinal1a / CVCCVFinal1b / CVCCVFinal2)
```

The primitive five letter predicates (and their borrowing friends).

```
PreComplex <- (ComplexTail / (!(CVCStressed /
CCVStressed / CVVStressed / ComplexTail /
StressedSyllable2)) Affix) PreComplex))
```

```
Complex <- (!((C1 V2 (juncture)? (V2)? (juncture)?
CVV)) !((C1 V2 !(stress) (juncture)? (V2)? !(stress)
(juncture)? (Primitive / PreComplex / Borrowing / CVV)))
!((C1 V2 (juncture)? &(MaybeInitialCC) C1 (juncture)?
&((PreComplex / ComplexTail)))) PreComplex)
```

A precomplex is a sequence of unstressed djifoa followed by a complex tail.

A complex satisfies initial restrictions on its opening to prevent things from falling off. An initial CVV must be phonetically hyphenated if followed by another CVV. An initial CVC-C must have the CC non initial (a y hyphen fixes this), unless the resulting complex has six letters. This is how the **slinkui** test was removed.

```
Predicate <- (((&(caprule) ((Primitive / Complex / Borrowing)
((( [ ])* Z A0 (', ')? ([ ])* Predicate)))) /
(C1 V2 (V2)? ([ ])* Z A0 (comma)?
([ ])* Predicate)) !(((juncture)? [y])))
```

The general predicate word. A predicate is read as a borrowing only if it cannot be read as a complex (or primitive). The **zao** alternative construction of complexes, proposed by Cowan, is supported.

A block of phonetic forms for building cmapua follows.

```
Fourvowels <- (C1 V2 (juncture)?
V2 (juncture)? V2 (juncture)? V2)
```

The initial consonant of a cmapua cannot be followed by four vowels.

```
B <- (!(Predicate) !(Fourvowels) [Bb])
```

```
C <- (!(Predicate) !(Fourvowels) [Cc])
```

```
D <- (!(Predicate) !(Fourvowels) [Dd])
F <- (!(Predicate) !(Fourvowels) [Ff])
G <- (!(Predicate) !(Fourvowels) [Gg])
H <- (!(Predicate) !(Fourvowels) [Hh])
J <- (!(Predicate) !(Fourvowels) [Jj])
K <- (!(Predicate) !(Fourvowels) [Kk])
L <- (!(Predicate) !(Fourvowels) [Ll])
M <- (!(Predicate) !(Fourvowels) [Mm])
N <- (!(Predicate) !(Fourvowels) [Nn])
P <- (!(Predicate) !(Fourvowels) [Pp])
R <- (!(Predicate) !(Fourvowels) [Rr])
S <- (!(Predicate) !(Fourvowels) [Ss])
T <- (!(Predicate) !(Fourvowels) [Tt])
V <- (!(Predicate) !(Fourvowels) [Vv])
Z <- (!(Predicate) !(Fourvowels) [Zz])
```

The initial consonant of a cmapua does not start a predicate word and is not followed by four vowels.

```
a <- ([Aa] (juncture2)? !(V2))
e <- (([Ee] (juncture2)?) !(V2))
```

```
i <- ([Ii] (juncture2)? !(V2))

o <- ([Oo] (juncture2)? !(V2))

u <- ([Uu] (juncture2)? !(V2))

V3 <- !Predicate V2

AA <- ([Aa] (juncture)? [a] (juncture2)?
      (&((V3 (juncture)? !(V2))) / !(Oddvowel)))

AE <- ([Aa] (juncture)? [e] (juncture2)?
      (&((V3 (juncture)? !(V2))) / !(Oddvowel)))

AI <- ([Aa] [i] (juncture2)?
      (&((V3 (juncture)? !(V2))) / !(Oddvowel)))

AO <- ([Aa] [o] (juncture2)?
      (&((V3 (juncture)? !(V2))) / !(Oddvowel)))

AU <- ([Aa] (juncture)? [u] (juncture2)?
      (&((V3 (juncture)? !(V2))) / !(Oddvowel)))

EA <- ([Ee] (juncture)? [a] (juncture2)?
      (&((V3 (juncture)? !(V2))) / !(Oddvowel)))

EE <- ([Ee] (juncture)? [e] (juncture2)?
      (&((V3 (juncture)? !(V2))) / !(Oddvowel)))

EI <- ([Ee] [i] (juncture2)?
      (&((V3 (juncture)? !(V2))) / !(Oddvowel)))

EO <- ([Ee] (juncture)? [o] (juncture2)?
      (&((V3 (juncture)? !(V2))) / !(Oddvowel)))

EU <- ([Ee] (juncture)? [u] (juncture2)?
      (&((V3 (juncture)? !(V2))) / !(Oddvowel)))
```

```
IA <- ([Ii] (junction)? [a] (junction2)?  
      (&((V3 (junction)? !(V2))) / !(Oddvowel)))  
  
IE <- ([Ii] (junction)? [e] (junction2)?  
      (&((V3 (junction)? !(V2))) / !(Oddvowel)))  
  
II <- ([Ii] (junction)? [i] (junction2)?  
      (&((V3 (junction)? !(V2))) / !(Oddvowel)))  
  
IO <- ([Ii] (junction)? [o] (junction2)?  
      (&((V3 (junction)? !(V2))) / !(Oddvowel)))  
  
IU <- ([Ii] (junction)? [u] (junction2)?  
      (&((V3 (junction)? !(V2))) / !(Oddvowel)))  
  
OA <- ([Oo] (junction)? [a] (junction2)?  
      (&((V3 (junction)? !(V2))) / !(Oddvowel)))  
  
OE <- ([Oo] (junction)? [e] (junction2)?  
      (&((V3 (junction)? !(V2))) / !(Oddvowel)))  
  
OI <- ([Oo] [i] (junction2)?  
      (&((V3 (junction)? !(V2))) / !(Oddvowel)))  
  
OO <- ([Oo] (junction)? [o] (junction2)?  
      (&((V3 (junction)? !(V2))) / !(Oddvowel)))  
  
OU <- ([Oo] (junction)? [u] (junction2)?  
      (&((V3 (junction)? !(V2))) / !(Oddvowel)))  
  
UA <- ([Uu] (junction)? [a] (junction2)?  
      (&((V3 (junction)? !(V2))) / !(Oddvowel)))  
  
UE <- ([Uu] (junction)? [e] (junction2)?  
      (&((V3 (junction)? !(V2))) / !(Oddvowel)))  
  
UI <- ([Uu] (junction)? [i] (junction2)?
```

```
(&((V3 (juncture)? !(V2))) / !(Oddvowel))
```

```
U0 <- ([Uu] (juncture)? [o] (juncture2)?  
&((V3 (juncture)? !(V2))) / !(Oddvowel))
```

```
UU <- ([Uu] (juncture)? [u] (juncture2)?  
&((V3 (juncture)? !(V2))) / !(Oddvowel))
```

Vowel segments of cmapua. The final `juncture2` enforces the rule that a stressed cmapua before a predicate must be followed by an explicit pause. Note that mandatory monosyllables are treated differently than disyllables.

A one letter form is followed by a non-vowel.

A two letter form is either followed by a single vowel followed by a consonant (in a Cvv-V unit) or an even number of vowels (nonzero only in an all vowel attitudinal).

4/27 the new rule V3 ensures that one has to pause after a CVV cmapua before a vowel-initial predicate.

```
__LWinit <- (([ ])* !(Predicate) &(caprule))
```

Rule governing the beginnings of cmapua words. The beginning of a cmapua word cannot be the beginning of a predicate word. A cmapua word is not followed immediately by by an A or I connective (an intervening explicit pause read as a free modifier is required before a connective); this is no longer handled by a separate rule, but simply as `!(connective)`.

```
CANCELPAUSE <- (comma (('y' comma) / (C UU (!connective))))
```

```
PAUSE <- (!(CANCELPAUSE) comma !(connective) !(V1))
```

PAUSE is the class of explicit pauses which are not mandated by phonetic rules. These are the pauses which *could* have semantic significance. At the moment, the only places where this rule is used are after legacy APA and

IPA connectives and after utterance-initial NO. This rule would be heavily used if any form of pause/GU equivalence were implemented.

CANCELPAUSE supports ways to say oops and cancel a possibly semantically significant pause. It also has applications in connection with pauses after name markers.

```
TAIO <- (!(Predicate) (((V1 (juncture)?
  !(Predicate) !(Name) M a (juncture2)?) /
  (V1 (juncture)? !(Predicate) !(Name) F i (juncture2)?) /
  (V1 (juncture)? !(Predicate) !(Name) Z i (juncture2)?) /
  (C1 AI (u)?) / (C1 EI (u)?) / (C1 EO) /
  (Z [i] (juncture)? V1 (juncture2)?
  ((M a))? (juncture2)?)) (!(Oddvowel) /
  (!( [ ]) &(TAIO))))))
```

The class of letteral forms. The Vfi and Vma legacy vowel letterals are supported. Cai, Cei, Ceo forms are supported. The new ZiV forms are preferred for vowel letters. Both sorts of vowel letterals can be capitalized with following ma.

Added the Greek vowels Vzi.

This class is mentioned very early for phonetic reasons.

```
NOI <- (N OI !(Oddvowel))
```

A general purpose negative suffix.

```
AO <- (!(Predicate) !((Mono / BrokenMono))
  (([AEUaeou] / (H a)) (juncture2)? !(V2)))
```

```
A <- (__LWinit !(TAIO) (((N [u]) &((u / (N [o])))))?
  ((N [o]))? AO (NOI)? !((( [ ]) + PANOPAUSES PAUSE))
  !((PANOPAUSES !(PAUSE) [ ,]))
  ((PANOPAUSES ((F i) / &(PAUSE))))?)
```

```
ANOFI <- ( _LWinit !(TAIO) ((N [u])
&((u / (N [o])))))? ((N [o]))? AO (NOI)?
(PANOPAUSES)?)
```

```
A1 <- (A (!connective))
```

```
ACI <- (ANOFI C i (!connective))
```

```
AGE <- (ANOFI G e (!connective))
```

The A logical connectives. The forms are described above in the reference grammar.

The most exciting bit here is management of the PA suffixes of APA connectives. Such connectives must be closed with PAUSE explicit pauses or with **-fi**. Notice that this means that an APA connective preceding a vowel initial word must be closed with the new **-fi**. The reason for the closure is that we have to be able to tell an APA connective from an A connective followed by a modifier. The form with the pause is in principle deprecated, but it is easiest to handle old text by preserving it.

Note that in (A modifier) there must be a space between the A and the modifier even if there is no pause; this is the only exception to the rule that any place where whitespace is mandatory requires a pause.

The ACI and AGE classes are additional classes of logical connectives with different precedence.

TAIO is defined early to avoid confusion of A connectives with Afi or Ama letterals.

```
CAO <- (((N o))? ((C a) / (C e) / (C o) /
(C u) / (Z e) / (C i H a))) (NOI)?
```

```
CA1 <- (((N u) &(((C u) / (N o))))?) ((N o))? CAO
!((( [ ])+ PANOPAUSES PAUSE))
!((PANOPAUSES !(PAUSE) [ ,])) ((PANOPAUSES
(F i) / &(PAUSE)))?)
```

```
CA1NOFI <- (((N u) &(((C u) / (N o))))?) ((N o))? CAO
```

(PANOPAUSES)?)

```
CA <- (__LWinit &(caprule) CA1 (!connective))
```

The CA series, another series with the same semantics as the A connectives but different semantics. The forms are discussed in the reference grammar. The full ability to suffix PA forms gives a larger range of words than is supported by LIP.

```
ZE2 <- (__LWinit (Z e) (!connective))
```

Uses of **ze** which are not in CA (between arguments).

```
I <- (__LWinit !(TAIO) i !((( [ ])+ PANOPAUSES PAUSE))
  !((PANOPAUSES !(PAUSE) [ ,])) ((PANOPAUSES ((F i) /
  &(PAUSE))))? (!connective))
```

```
ICA <- (__LWinit !(Predicate) i ((H a) / CA1) (!connective))
```

```
ICI <- (__LWinit i (CA1NOFI)? C i (!connective))
```

```
IGE <- (__LWinit i (CA1NOFI)? G e (!connective))
```

The I class sentence and utterance connectives. The I class takes PA suffixes. An IPA connective must be closed with **-fi** or a PAUSE explicit pause.

```
connective <- (ACI / AGE / A1 / ICI / ICA / IGE / I / &V1 TAIO)
```

Logical and utterance connectives. This class is used for phonetic tests enforcing the need to pause before these connectives. For phonetic reasons, the vowel initial legacy letterals are included in this class.

```

KAO <- (((K a) / (K e) / (K o) / (K u) / (K i H a)))

KOU <- (((K OU) / (M OI) / (R AU) / (S OA)/C IU/M OU))

KOU1 <- (((N u N o) / (N u) / (N o)) KOU)

KA <- (__LWinit &(caprule) (((((N u) &((K u))))? KAO) /
((KOU1 / KOU) K i)) (NOI)? (!connective))

KOU2 <- KOU1 !KI

KI <- (__LWinit (K i) (NOI)? (!connective))

```

The KA and KI forms for forethought logical connection. PA suffixing is not supported as in LIP; it could be installed if wanted. The KOU1 class of modifiers is introduced early because of its role in the formation of forethought causal connectives. The forms are described in the reference grammar.

3/9 **ciu** and **mou** added to KOU to support formation of words listed in Paradigm K.

```

BadNISTress <- ((C1 V2 (V2)? stress ((M a))? ((M OA))? NI RA) /
(C1 V2 stress V2 ((M a))? ((M OA))? NI RA))

NIO <- (!(BadNISTress) (((K UA) / (G IE) / (G IU) /
(H IE) / (H IU) / (K UE) / (N EA) / (N IO) / (P EA) /
(P IO) / (S UU) / (S UA) / (T IA) / (Z OA) / (Z OO) /
(H o) / (N i) / (N e) / (T o) / (T e) / (F o) / (F e) /
(V o) / (V e) / (P i) / (R e) / (R u) / (S e) / (S o) /
(H i))))

SA<-(!BadNISTress ((S a)/(S i)/(S u)/
(IE (comma2? !IE SA)?)) NOI?)

RA <- (!(BadNISTress) (((R a) / (R i) /

```

```

(R o)))

NI1 <- ((NIO (!(BadNIStress) M a))?
        (!(BadNIStress) M OA (NIO)*))?
        ((comma2 !(NI RA)) &(NI)))?)

RA1 <- ((RA (!(BadNIStress) M a))?
        (!(BadNIStress) M OA (NIO)*))?
        ) ((comma2 !(NI RA)) &(NI)))?)

NI2 <- (((SA)? ((NI1)+ / RA1)) / SA) (NOI)?
        ((CAO (((SA)? ((NI1)+ / RA1)) / SA) (NOI)?))*))

NI <- (__LWinit NI2 ((&((M UE)) Acronym
                    (comma / &(end) /
                    &(period)) !(C u))))? ((C u)))?)

mex <- (__LWinit NI (!connective))

```

The quantifier word formations. The forms are described in the reference grammar. Note in particular that pauses are permitted in certain contexts in NI words (look for the `comma(2)` classes).

`BadNIStress` is designed to enforce the rule of penultimate stress on numerical predicates (the rule attempts to detect a badly placed stress). It doesn't necessarily enforce it perfectly. Normally of course one is not writing explicit stresses in one's numerical predicates.

A NI word will continue through a whitespace or even explicit comma pause between NI1 units, except that it will not absorb a numerical predicate (NI RA): this is achieved in the class NI1 of numeral units by allowing a unit to absorb a following comma or whitespace followed by a numeral NI, unless it is followed by a NI RA numerical predicate.

bug corrected in SA 5/5/17

```
CI <- (__LWinit (C i) (!connective))
```

The little word `ci` has multiple uses.

```
Acronym <- (([ ])* &(caprule) ((M UE) /
TAIO / ([Zz] V2 !(V2))) ((comma &Acronym M UE / NI1 / TAI0 /
([Zz] V2 !(V2) / ([Zz] &(V2)))))))+
```

The class of acronyms, used for acronymic names and dimension suffixes to NI words. Note that we do not support acronymic *predicates* as in 1989 Loglan, replacing these with names. We regard this as both better semantics and better phonetics, for reasons discussed in the reference grammar.

Pauses followed by MUE may be inserted into an acronym.

```
TAI <- (__LWinit (TAIO /
((G AD) !(badspaces) !(V2) ([ ])*
(Name / Predicate / (C1 V2 V2 !(Oddvowel) /
&(TAIO))) / (C1 V2 !(Oddvowel) /
&(TAIO)))))) !(connective))
```

```
DA0 <- (((T AO) / (T IO) / (T UA) /
(M IO) / (M IU) / (M UO) / (M UU) /
(T OA) / (T OI) / (T OO) / (T OU) /
(T UO) / (T UU) / (S UO) / (H u) /
(B a) / (B e) / (B o) / (B u) / (D a) /
(D e) / (D i) / (D o) / (D u) / (M i) /
(T u) / (M u) / (T i) / (T a) / (M o)))
```

```
DA1 <- ((TAIO / DA0) ((C i !([ ]) NIO))?)
```

```
DA <- (__LWinit DA1 (!connective))
```

Pronoun forms (and letter names). The prerequisite class TAI0 appeared early. DA1 allows the attachment of one-unit numerical suffixes to pronouns. Note that a pronoun contains no more than one letteral: it may in addition be linked to a single digit by **-ci-**.

Class TAI includes Cowan's proposed *gao* construction of letterals from words of quite general form.

THINK ABOUT: consider attachment of numerical indices to **gao** form letterals.

```
PAO <- (((G IA) / (G UA) / (P AU) /
(P IA) / (P UA) / (N IA) / (N UA) / (B IU) /
(F EA) / (F IA) / (F UA) / (V IA) / (V II) /
(V IU) / (C OI) / (D AU) / (D II) /
(D UO) / (F OI) / (F UI) / (G AU) / (H EA) /
(K AU) / (K II) / (K UI) / (L IA) / (L UI) /
(M IA) / (N UI) / (P EU) / (R OI) /
(R UI) / (S EA) / (S IO) / (T IE) /
(V a) / (V i) / (V u) /
(P a) / (N a) / (F a) / (V a) / KOU !KI))
```

```
PAO<-((N u !KOU)? ((G IA)/(G UA)/(P AU)/
(P IA)/(P UA)/(N IA)/(N UA)/(B IU)/(F EA)/
(F IA)/(F UA)/(V IA)/(V II)/(V IU)/(C OI)/
(D AU)/(D II)/(D UO)/(F OI)/(F UI)/(G AU)/
(H EA)/(K AU)/(K II)/(K UI)/(L IA)/(L UI)/
(M IA)/(N UI)/(P EU)/(R OI)/(R UI)/(S EA)/
(S IO)/(T IE)/(V a)/(V i)/(V u)/(P a)/(N a)/(F a)/
(V a)/(KOU !(N OI) !KI)) (N OI)?
```

```
PANOPAUSES <- (((!(PAO) NI))? ((KOU2 / PAO))+
((((comma2)? CA0 (comma2)?) ((KOU2 / PAO))+))*
(ZI)?)
```

```
PA3 <- (__LWinit PANOPAUSES (!connective))
```

```
PA <- (((!(PAO) NI))? ((KOU2/ PAO))+
((((comma2)? CA0 (comma2)?) /
(comma2 !(mod1a)))
((KOU2 / PAO)+))* (ZI)?)
```

```
PA2 <- (__LWinit PA (!connective))
```

```
GA <- (__LWinit (G a) (!connective))
```

```
PA1 <- ((PA2 / GA) (!connective))
```

The PA words which serve to mark modifiers and form tensed predicates. The prerequisite KOU1 and KOU classes were mentioned earlier due to their role in forming forethought connectives. PA syllables can be concatenated and linked with CA cores as described in the reference grammar; internal pauses are permitted in some contexts. Pauses next to CA0 links are always allowed; pauses between PA0 units are permitted except in the class PANOPAUSES.

PA3 forms modifiers with an argument; PA2 forms modifiers by itself; PA1 is the class of tense markers (including **ga** as an additional option).

3/9 bug fix prevents KOU followed by KI from being read as a modal operator.

3/18 PA roots which are not KOU roots may be converted with **nu-** and/or negated with **-noi** (experimentally). The forms for the KOU words remain as before.

```
ZI <- ((Z i) / (Z a) / (Z u))
```

Suffix forms with multiple uses.

```
LE <- (__LWinit ((L EA) / (L EU) /
(L OE) / (L EE) / (L AA) /
(L e) / (L o) / ((L a) !(badspaces))) (!connective))
```

```
LEFORPO <- (__LWinit ((L e) / (L o) / NI2) (!connective))
```

```
LIO <- (__LWinit (L IO) (!connective))
```

```
LAU <- (__LWinit (L AU) (!connective))
```

```
LOU <- (__LWinit (L OU) (!connective))
```



```

LUA <- (__LWinit (L UA) (!connective))
LUO <- (__LWinit (L UO) (!connective))
ZEIA <- (__LWinit Z EI a (!connective))
ZEIO <- (__LWinit Z EI o (!connective))
LI1 <- (L i)
LU1 <- (L u)
LI2 <- ((["] L i) / (L i ["]))
LU2 <- ((L u ["]) / (["] L u))

```

Various article forms (ZEIA and ZEIO are “commas” used in set and list forms).

12/2/17 update added forms of **li** and **lu** with directly adjacent double quotes for a punctuation upgrade.

```

Quotemod <- (((Z a) / (Z i)))
LI <- (((__LWinit LI1 !(V2) (Quotemod)?
((([,])? ([ ])+))? utterance0 (', ')? __LWinit LU1 !(connective)) /
(__LWinit LI1 !(V2) (Quotemod)? comma name
(comma)? __LWinit LU1 !(connective))) /
((__LWinit LI2 !(V2) (Quotemod)?
((([,])? ([ ])+))? utterance0 (', ')? __LWinit LU2 !(connective)) /
(__LWinit LI2 !(V2) (Quotemod)? comma name
(comma)? __LWinit LU2 !(connective))))

```

The construction for quotation of Loglan utterances.

12/2/17 update: **li** can be adorned with a directly adjacent double quote (on either side) if and only if the matching **lu** is so adorned.

```

stringnospaces <- (([,])? ((([, ])+ [\"]) / (([, ])+
!(["])) ((!(["],)]? [ ,]) !((["])? (period / !(.))) .))+)
  (&(["]) / (([,])? ([ ])+ &(letter)) / &(period) / &(end)))

stringnospacesclosed <- (([,])? ((([, ])+ [\"]) /
(([, ])+ !(["])) ((!(["],)]? [ ,]) !((["])?
(period / !(.))) .))+) (([,] ([ ])+ / &(period) / &(end)))"

stringnospacesclosedblock <- ((stringnospaces
  ((!(["] stringnospacesclosed)) [y] stringnospaces))*
  ([y] stringnospacesclosed) / stringnospacesclosed)

LAO1 <- (L AO)

LAO <- (([, ])* (LAO1 stringnospaces (([y] stringnospaces))*))

LIE1 <- (L IE)

LIE <- ((([, ])* (LIE1 (Quotemod)? &((([, ])+ [\"])))
stringnospaces (([y] stringnospaces))*))
["] (comma)?) /
(([, ])* (LIE1 (Quotemod)? &((([, ])+ !(["]))))
stringnospaces (([y] stringnospaces))*))"

```

Constructions of alien text and constructions using alien text. **stringnospaces** is the basic alien text construction. It may optionally begin with an explicit comma pause and will at least begin with whitespace, and will end with an explicit pause, period, end of text or whitespace. Phonetically, it will be set off with pauses (its pronunciation will not be set by any Loglan standard).

The other constructions describe alien text blocks not ending with mere whitespace and sequences of such blocks separated with the word **y**, used in special contexts.

lao followed by blocks of alien text set off with the pause word **y** form forms foreign names.

lie followed by the same thing will form strong quotations (this is quite different from the 1989 Loglan construction).

12/2/17 update: `stringnospaces` and `stringnospacesclosed` are modified so that they end before double quotes followed by comma, space, terminal punctuation, or end of file (other than an initial double quote).

Class LIE is modified so that the quoted string can be enclosed in double quotes: if the initial one is present, the final one must also be present.

This has the unintended side effect that alien text blocks other than those with `lie` may begin with a double quote, though they may not end with a double quote. This may I suppose be used as a device to emphasize their alienness.

```
LW <- (&(caprule) (((!(Predicate) V2 V2))+ /
((!(Predicate) (V2)? ((!(Predicate) LWunit))+) / V2)))
```

The NB3 construction of `cmapua` words. It is only used in the immediately following word quotation construction.

```
LIU0 <- ((L IU) / (N IU))
```

```
LIU1 <- (__LWinit ((LIU0 !(badspaces) !(V2) (Quotemod)?
((([,])? ([ ])+))?) (Name / (Predicate (comma)?) /
(CCV (comma)?) / (LW ([[,] ([ ])+ !([,])]) / &(period) /
&(end) / &((([ ])* Predicate)))))) /
(L II (Quotemod)? TAI (!connective))))
```

Quotation of words (and of CCV `djifoa`) with **liu** or **niu** and letters with **lii**. Quoted `cmapua` may need to be closed with explicit pauses.

```
SUE <- (__LWinit ((S UE) / (S A0))
stringnospaces)
```

This handles the phonetically identical though semantically quite different constructions of foreign predicates and onomatopoeic predicates from blocks of alien text.

```
CUI <- (__LWinit (C UI) (!connective))

GA2 <- (__LWinit (G a) (!connective))

GE <- (__LWinit (G e) (!connective))

GEU <- (__LWinit ((C UE) /
  (G EU)) (!connective))

GI <- (__LWinit ((G i) / (G OI)) (!connective))

GO <- (__LWinit (G o) (!connective))

GIO <- (__LWinit (G IO) (!connective))

GU <- (__LWinit (G u) (!connective))

GUIZA <- (__LWinit (G UI) (Z a) !(connective))

GUIZI <- (__LWinit (G UI) (Z i) !(connective))

GUIZU <- (__LWinit (G UI) (Z u) !(connective))

GUI <- (!(GUIZA) !(GUIZI) !(GUIZU) (__LWinit (G UI) !(connective)))

GUO <- (__LWinit (G UO) (!connective))

GIUO <- (__LWinit (G IU o) (!connective))

GUOA <- (__LWinit (G UO (Z)? a) (!connective))

GUOE <- (__LWinit (G UO e) (!connective))

GUOI <- (__LWinit (G UO (Z)? i) (!connective))

GUOO <- (__LWinit (G UO o) (!connective))
```

```
GUOU <- (__LWinit (G UO (Z)? u) (!connective))
```

```
GUU <- (__LWinit (G UU) (!connective))
```

```
GUUA <- (__LWinit (G UU a) (!connective))
```

```
GUE <- (__LWinit (G UE) (!connective))
```

```
GUEA <- (__LWinit (G UE a) (!connective))
```

```
MEU <- (__LWinit (M EU) (!connective))
```

A large collection of opening and closing forms for constructions. The GUOV forms are part of a new proposal for multiple possible closures of abstraction predicates and descriptions. The GUIZV forms are provided as part of the alternative parser with similar motivation re subordinate clauses.

The archaic form **cue** for **geu** is supported.

```
JE <- (__LWinit (J e) (!connective))
```

```
JUE <- (__LWinit (J UE) (!connective))
```

Initial markers for tightly bound arguments and modifiers.

```
JIZA <- (__LWinit ((J IE) / (J AE) / (P e) / (J i) / (J a) /
  (N u J i)) (Z a) !(connective))
```

```
JIOZA <- (__LWinit ((J IO) / (J AO)) (Z a) !(connective))
```

```
JIZI <- (__LWinit ((J IE) / (J AE) / (P e) / (J i) / (J a) /
  (N u J i)) (Z i) !(connective))
```

```
JIOZI <- (__LWinit ((J IO) / (J AO)) (Z i) !(connective))
```

```
JIZU <- (__LWinit ((J IE) / (J AE) / (P e) / (J i) / (J a) /
(N u J i)) (Z u) !(connective))
```

```
JIOZU <- (__LWinit ((J IO) / (J AO)) (Z u) !(connective))
```

```
JI <- (!(JIZA) !(JIZI) !(JIZU) (__LWinit ((J IE) / (J AE) /
(P e) / (J i) / (J a) / (N u J i)) !(connective)))
```

```
JIO <- (!(JIOZA) !(JIOZI) !(JIOZU) (__LWinit ((J IO) /
(J AO)) !(connective)))
```

Initial markers for argument modifiers (subordinate clauses). The extra suffixed forms are provided in the alternative parser, and will probably be added to the official parser.

```
DIO <- (__LWinit ((B EU) / (C AU) / (D IO) /
(F OA) / (K AO) / (J UI) / (N EU) / (P OU) /
(G OA) / (S AU) / (V EU) / (Z UA) /
(Z UE) / (Z UI) / (Z UO) / (Z UU)) !(connective))
```

Case tags, positional and semantic. Notice that **lae** and **lue** are no longer in this class.

```
LAE <- (__LWinit ((L AE) / (L UE)) !(connective))
```

Tags which indicate indirect reference (address or referent).

```
ME <- (__LWinit ((M EA) / (M e)) !(connective))
```

Forms that convert arguments to predicates. I believe **mea** was never needed.

```

NUO <- (((N UO) / (F UO) / (J UO) /
(N u) / (F u) / (J u)))

NU <- (__LWinit ((NUO !((( [ ])+ (NIO / RA)))
((NIO / RA))?) (freemod)?)+ (!connective))

```

Conversion and reflexive operators on predicates. The fourth and fifth place operators and the last place operator (!) are formed using NIO or RA suffixes.

```

P01 <- (__LWinit ((P o) / (P u) / (Z o)))

P01A <- (__LWinit ((P OI a) / (P UI a) / (Z OI a) /
(P o Z a) / (P u Z a) / (Z o Z a)) )

P01E <- (__LWinit ((P OI e) / (P UI e) / (Z OI e))
)

P01I <- (__LWinit ((P OI i) / (P UI i) / (Z OI i) /
(P o Z i) / (P u Z i) / (Z o Z i)) )

P01O <- (__LWinit ((P OI o) / (P UI o) / (Z OI o))
)

P01U <- (__LWinit ((P OI u) / (P UI u) / (Z OI u) /
(P o Z u) / (P u Z u) / (Z o Z u)) )

POSHORT1 <- (__LWinit ((P OI) / (P UI) / (Z OI)) )

PO <- (__LWinit P01 (!connective))

POA <- (__LWinit P01A (!connective))

POE <- (__LWinit P01E (!connective))

```

```
POI <- (__LWinit P01E (!connective))
POO <- (__LWinit P01O (!connective))
POU <- (__LWinit P01U (!connective))
POSHORT <- (__LWinit POSHORT1 (!connective))
```

operators to form abstractions for predicates and descriptions. The additional series are part of a new proposal to allow more effective closures of abstract descriptions (and in theory of predicates as well).

```
DIE <- (__LWinit ((D IE) / (F IE) /
(K AE) / (N UE) / (R IE)) (!connective))
```

Register markers (attitudinals indicating attitude toward the person addressed).

```
HOI <- (__LWinit ((H OI) /
(L OI) / (L OA) / (S IA) / (S IE) / (S IU)) (!connective))
```

The vocative **hoi**; the words of social lubrication may also be used as vocative operators in most cases.

```
JO <- (__LWinit ((NIO / RA))? (J o) (!connective))
```

the “so-called” attitudinal. The number indicates how many previous words are affected.

```
KIE <- (__LWinit (K IE) (!connective))
```



```

KIU <- (__LWinit (K IU) (!connective))

KIE2 <- ((__LWinit [(] (K IE) !(connective)) /
  (__LWinit (K IE) [(] !(connective)))

KIU2 <- ((__LWinit (K IU) [)] !(connective)) /
  (__LWinit [)] (K IU) !(connective)))

```

spoken parentheses to create an attitudinal parenthetic remark.

12/2/17 update: alternative forms are provided with directly adjacent parentheses.

```
SOI <- (__LWinit (S OI) (!connective))
```

The smilie constructor.

```

UIO <- ((UA / UE / UI / UO / UU /
  OA / OE / OI / OU / OO /
  IA / II / IO / IU /
  EA / EE / EI / EO / EU /
  AA / AE / AI / AO / AU /
  (B EA) / (B UO) / (C EA) /
  (C IA) / (C OA) / (D OU) /
  (F AE) / (F AO) / (F EU) /
  (G EA) / (K UO) / (K UU) /
  (R EA) / (N AO) / (N IE) /
  (P AE) / (P IU) / (S AA) /
  (S UI) / (T AA) / (T OE) /
  (V OI) / (Z OU) /
  (L OI) / (L OA) / (S IA) / (S II) /
  (T OE) / (S IU) / (C AO) /
  (C EU) / (S IE) / (S EU)) )

```

A grab bag of attitudinal words. See the reference grammar for meanings.

```
NOUI <- ((__LWinit N [o] juncture? ([ ])* UIO (!connective)) /
  (__LWinit UIO NOI (!connective)))
```

```
UII <- (__LWinit (UIO / (NI F i)) (!connective))
```

Negative attitudinal and attitudinal words.

The use of **noi** as an alternative negative suffix is new.

The phonetics of negative attitudinals are exceptional: notice that four vowels after a consonant are allowed.

```
HUE <- (__LWinit (H UE) (!connective))
```

The inverse vocative marker.

```
NOI <- (__LWinit !(KOU1) !(NOUI) (N o) !((__LWinit KOU))
  !((( [ ])* (JIO / JI / JIZA / JIOZA / JIZI / JIOZI / JIZU / JIOZU)))
  !(connective))
```

real occurrences of **no** as a word, not subsumed in other constructions such as forethought causal connectives, negative attitudinals, and negative subordinate clause constructions. Genuine **no** followed by a KOU word must be marked with an explicit pause.

```
AcronymicName <- (Acronym (&(end) / ', ' /
  &(period) / &(Name) / &(CI)))
```

```
DJAN <- (Name / AcronymicName)
```

Name words, adding in the acronymic names.

```
BI <- (__LWinit ((N u))? ((B IA) / (B IE) /  
  (C IE) / (C IO) / (B IA) / (B [i])) (!connective))  
  
LWPREDA <- (((H e) / (D UA) / (D UI) /  
  (B UA) / (B UI)) )
```

Little words which are semantically predicates.

```
PREDA <- (([ ])* &(caprule)  
  (Predicate / LWPREDA /  
  (!([ ] NI RA)) !(connective))
```

All predicate words other than the BI identity predicates. NI RA are the numerical predicates.

```
guo <- ((PAUSE)? (GUO / GU) (freemod)?)  
  
giuo <- ((PAUSE)? (GIUO / GU) (freemod)?)  
  
guoa <- ((PAUSE)? (GUOA / GU) (freemod)?)  
  
guoe <- ((PAUSE)? (GUOE / GU) (freemod)?)  
  
guoi <- ((PAUSE)? (GUOI / GU) (freemod)?)  
  
guoo <- ((PAUSE)? (GUOO / GU) (freemod)?)  
  
guou <- ((PAUSE)? (GUOU / GU) (freemod)?)  
  
guiza <- ((PAUSE)? (GUIZA / GU) (freemod)?)  
  
guizi <- ((PAUSE)? (GUIZI / GU) (freemod)?)
```

```

guizu <- ((PAUSE)? (GUIZU / GU) (freemod)?)
gui <- ((PAUSE)? (GUI / GU) (freemod)?)
gue <- ((PAUSE)? (GUE / GU) (freemod)?)
guea <- ((PAUSE)? (GUEA / GU) (freemod)?)
guu <- ((PAUSE)? (GUU / GU) (freemod)?)
guua <- ((PAUSE)? (GUUA / GU) (freemod)?)
geu <- GEU
gap <- ((PAUSE)? GU (freemod)?)

```

Closing forms. All except GEU may alternatively be expressed as **gu**, and may be preceded by pauses and followed by free modifiers.

```

juelink <- (JUE (freemod)? (term/PA2 freemod? gap?))
links1 <- (juelink (((freemod)? juelink))* (gue)?)
links <- ((links1 / (KA (freemod)? links (freemod)?
  KI (freemod)? links1)) (((freemod)? A1 (freemod)? links1))*
jelink <- (JE (freemod)? (term/PA2 freemod? gap?))
linkargs1 <- (jelink (freemod)? (links/gue)?)
linkargs <- ((linkargs1 / (KA (freemod)?
linkargs (freemod)? KI (freemod)? linkargs1))
  (((freemod)? A1 (freemod)? linkargs1))*

```

The construction of tightly bound argument lists (link sets). Unlike 1989 Loglan, a JE or JUE link can be either an argument or a modifier. JE links

are first arguments in link sets; JUE links are sutori arguments. GUE will close a link set; the JE/JUE distinction is designed to make fewer explicit uses of GUE necessary. Link sets built just with JUE or with an initial JE link can be linked with forethought and afterthought logical and causal connectives.

Very subtle bug fix to links of the form JE PA or JUE PA 3/18. There is no reason for these to fail if followed immediately by a barepred, as a mod1 would (to avoid confusion with a “tense”).

```
abstractpred <- ((POA (freemod)? uttAx (guoa)?) /
  (POA (freemod)? sentence (guoa)?) /
  (POE (freemod)? uttAx (guoe)?) /
  (POE (freemod)? sentence (guoe)?) /
  (POI (freemod)? uttAx (guoi)?) /
  (POI (freemod)? sentence (guoi)?) /
  (POO (freemod)? uttAx (guoo)?) /
  (POO (freemod)? sentence (guoo)?) /
  (POU (freemod)? uttAx (guou)?) /
  (POU (freemod)? sentence (guou)?) /
  (PO (freemod)? uttAx (guo)?) /
  (PO (freemod)? sentence (guo)?))
```

Abstraction predicates, with the suite of alternative openings and closures (definitely useful for the much more often used abstract descriptions; included here by analogy).

```
predunit1 <- ((SUE /
  (NU (freemod)? GE (freemod)? despredE (((freemod)? geu (comma)?))?) /
  (NU (freemod)? PREDA) /
  ((comma)? GE (freemod)? descpred (((freemod)? geu (comma)?))?) /
  abstractpred / (ME (freemod)? argument1 (meu)?) / PREDA) (freemod)?
```

“atomic” predicate units. These are described item by item in the reference grammar. The inclusion of abstraction predicates formed from sentences in predunit1 is a major change from the Loglan 1989 grammar, but also an obviously needed one.

```
predunit2 <- (((NO1 (freemod)?))* predunit1)
```

```
NO2 <- (!(predunit2) NO1)
```

Possibly multiply negated atomic predicate units.

NO2 is the class of occurrences of **no** with sentence negating effect: NO1's which are not NO2's negate modifying predicates in metaphors.

```
predunit3 <- ((predunit2 (freemod)? linkargs) / predunit2)
```

```
predunit <- (((POSHORT (freemod)?))? predunit3)
```

More predicate units. The predunit3 construction optionally attaches a link set; the predunit construction optionally attaches a POSHORT (the old short scope **po pu zo**, now with the different phonetic shape **poi pui zoi**).

predunit is the class of things we call "predicate units".

```
kekpredunit <- (((NO1 (freemod)?))*  
KA (freemod)? predicate (freemod)?  
KI (freemod)? predicate guu?)
```

Forethought connected predicates built from predicates of the most general form are treated as atomic units. Notice that these are possibly multiply negated.

```
despredA <- ((predunit / kekpredunit)  
(((freemod)? CI (freemod)? (predunit / kekpredunit))))*
```

```
despredB <- (!(PREDA) CUI (freemod)? despredC (freemod)?  
CA (freemod)? despredB) / despredA)
```

```
despredC <- (despredB (((freemod)? despredB)))*
```

```
despredD <- (despredB (((freemod)?
CA (freemod)? despredB))*)
```

```
despredE <- (despredD (((freemod)? despredD))*)
```

```
descpred <- ((despredE (freemod)?
GO (freemod)? descpred) / despredE)
```

The construction of description predicates, described in the reference grammar. `despredE` are called “simple description predicates”.

```
sentpred <- ((despredE (freemod)?
GO (freemod)? barepred) / despredE)
```

The class of sentence predicates, described in the reference grammar. A principal motivation here is preventing sentence predicates from being modified by forethought predicate units at the head; but later an innovation was introduced to allow this. This suggests that the distinction between these two classes of predicates may need to be rethought.

There is another distinction: sentence predicates may have argument attached loosely to the predicate after a **go**, where description predicates would have to have tightly attached link sets. After the revision of 5/1/17, this is the only difference between `descpred` and `sentpred`.

```
mod1a <- (PA3 (freemod)? argument1 (guua)?)
```

```
mod1 <- ((PA3 (freemod)? argument1 (guua)?) /
(PA2 (freemod)? !(barepred) (gap)?))
```

```
kekmod <- (((NO1 (freemod)?))*
(KA (freemod)? modifier (freemod)?
KI (freemod)? mod))
```

```
mod<-(mod1/((N01 freemod?)* mod1)/
kekmod)
```

```
modifier<-(mod (A1 freemod? mod)*)
```

The construction of modifiers (relative clauses modifying predicates).

```
maybebreak <- (V1 (stress)? ' ' !((( [ ])* V1)))
```

```
realbreak <- (!(maybebreak) letter (stress)? ((([,])? ' ') /
period / &(end)))
```

```
consonantbreak <- (C1 (stress)? ((([,])? ' ') /
period / &(end)))
```

```
badspaces <- (!(([,] ' ')) (!(maybebreak / realbreak) .))*
maybebreak (!(realbreak) .))* consonantbreak)
```

Fancy tools for making sure that left boundaries of name words are plainly marked. `maybebreak` is whitespace between a vowel and a consonant, where an actual phonetic pause may occur but one cannot be certain that it will occur. A `realbreak` is a break which is definite because of an explicit pause, terminal punctuation or end of text. A `consonantbreak` is a consonant followed by an explicit comma pause, terminal punctuation or end of text (something that looks like the end of a name word). A `badspaces` situation exists where a `maybebreak` or `maybebreaks` exists before a `consonantbreak` without an intervening `realbreak`: this situation makes ambiguity about where the name word ending at the `consonantbreak` a possibility, and so is forbidden. This is used by requiring that the `badspaces` situation cannot obtain after a name marker word (notice that if an explicit comma pause follows a name marker word the `badspaces` situation cannot obtain). This forces the writer to indicate an explicit pause before a name word where one must be placed to avert ambiguity. The kinds of ambiguities averted are quite specific: the problems which arise occur where there might be doubts as to whether a phonetically occurring name marker is actually a name marker or is part of a following name word.


```

namemarker <- ((([ ])* ((L a) / (H OI) /
(L OI) / (L OA) / (S IE) /
(S IA) / (S IU) / (C i) /
(H UE) / (L IU) / (G AO))) !(badspaces))

nonamemarkers<-([ ]* (!(namemarker Name) Letter)+
!Letter)

```

The class of name marker words.

The class of name words which do not contain a false name marker (a phonetic copy of a name marker followed by a well-formed name word).

```

CIO<-([Cc] i &([ ]* C1))

name<-(DJAN ((CIO DJAN)/
(CI !badspaces comma? predunit
!(&nonamemarkers Name)))/
(CI comma? DJAN)/
(&nonamemarkers Name))* freemod?)

```

CIO is the bare phonetic form of the little word CI as used in the following class.

name is the class of serial names. A serial name begins with a name word and is a series of name words and predunits. Any predunit must be initially marked with **ci** and the unit following it must be marked with **ci**. An acronymic name word or a name word containing a false name marker must be marked with **ci**.

```

LAO <- (([Ll] a) !(badspaces))

LANAME <- (([ ])* LAO (CANCELPAUSE / (([ ])* &(C1))) name )

LANAME2 <- (([ ])* LAO ((',' ([ ])+) / (([ ])* &(V1))) name )

```

LA0 is **la** used as a name marker.

What follows are two forms of name marked with **la**: the first form is a consonant initial name with no pause after the **la**; the second form is **la** followed by an explicit pause or by a vowel-initial name, which forces an explicit pause.

The reason for the distinction is that in the LANAME situation the parser prefers to read what follows as a name; in the LANAME2 situation it reads the consonant final segment following the **la** as a name only after attempting to read it in other ways. In a phonetic transcript, for example, there is no reason to believe that a consonant final block of letters actually is a name word.

This distinction does have the effect that whitespace following **la** and preceding a consonant initial and final block should be assumed not to be a pause, as inserting an explicit pause may change the parse. Notice that CANCELPAUSE can be used to cancel an unintended pause before a consonant initial name, with this in mind.

```
HOIO <- ((([Hh] OI) / ([Ll] OI) / ([Ll] OA) /
  ([Ss] IA) / ([Ss] IE) / ([Ss] IU)) !(badspaces))

voc <- ((([ ])* HOIO (CANCELPAUSE /
  ([ ])* &(C1))) name ) /
  (HOI !(badspaces) (freemod)? descpred guea? (((((comma)? CI (comma)?) /
  (comma &(nonamemarkers) !(AcronymicName))) name)))? /
  (HOI !(badspaces) (freemod)? argument1 (guua)?) /
  ([ ])* HOIO ((',' ([ ])+) / ([ ])* &(V1))) name ) /
  (H OI stringnospacesclosedblock))
```

HOIO is **hoi** or a word of social lubrication used as a name marker.

voc is the vocative construction. What follows the name marker starting the vocative construction may be a consonant initial name without a pause after the name marker, or a description predicate, possibly with a name appended, or an argument, or a vowel-initial name or name preceded by an explicit pause, read in that order. Note that a consonant initial name without a preceding pause is read by preference as a name; all other apparent names are read as non-names if this is possible.

A name appended to a descriptive predicate must either be preceded by an explicit pause or by **ci**: if it contains a false name marker it must be preceded by **ci**.

```
descriptn <- (!(LANAME) ((LAU wordset1) /
  (LOU wordset2) / (LE (freemod)?
  ((((!mex) arg1a (freemod)?))?) ((PA2 (freemod)?))?)?)?
  mex (freemod)? descpred) /
  (LE (freemod)? ((((!mex) arg1a (freemod)?))?)
  ((PA2 (freemod)?))?)?)? mex (freemod)? arg1a) /
  (GE (freemod)? mex (freemod)? descpred) /
  (LE (freemod)? ((((!mex) arg1a (freemod)?))?)
  ((PA2 (freemod)?))?)?)? descpred)))
```

Descriptions are a class of simple arguments formed with articles, described in the reference grammar. Note that this class includes the extended possessive construction: **lemina hasfa** parses as **le mi na hasfa** (so that **lemina** does not need to be a word as in 1989 Loglan) and by extension one can say **le, la Djan, na hasfa**, which one could not say in 1989 Loglan (“John’s present house”).

Also notice that explicit set and list constructions are incorporated there. Their status in 1989 Loglan was entirely unsatisfactory. The fact that the modifying argument in the possessive construction cannot begin with a **mex** (abstract descriptions can start with a **mex**) illustrates how fragile this construction is: I don’t advocate lots of use of it!

```
abstractn <- ((LEFORPO (freemod)? POA (freemod)? uttAx (guoa)?) /
  (LEFORPO (freemod)? POA (freemod)? sentence (guoa)?) /
  (LEFORPO (freemod)? POE (freemod)? uttAx (guoe)?) /
  (LEFORPO (freemod)? POE (freemod)? sentence (guoe)?) /
  (LEFORPO (freemod)? POI (freemod)? uttAx (guoi)?) /
  (LEFORPO (freemod)? POI (freemod)? sentence (guoi)?) /
  (LEFORPO (freemod)? POO (freemod)? uttAx (guoo)?) /
  (LEFORPO (freemod)? POO (freemod)? sentence (guoo)?) /
  (LEFORPO (freemod)? POU (freemod)? uttAx (guou)?) /
```

```
(LEFORPO (freemod)? POU (freemod)? sentence (guou)?) /
(LEFORPO (freemod)? PO (freemod)? uttAx (guo)?) /
(LEFORPO (freemod)? PO (freemod)? sentence (guo)?))
```

This is the class of abstract descriptions. This incorporates my proposal of many additional options of openers and closers for this construction, which should minimize the need for **guo guo** and in general assist with the serious problem of recognizing when an abstract description is closed. LEFORPO is used because an abstract description can start with certain quantifiers.

Notice that an abstract description (LE PO stuff) does **not** include an abstract predicate (PO stuff) as a component and that GUO closes both kinds of construction. An expression LE (PO stuff) GUO predicate is read as an abstract description followed by a predicate; if one wants (PO stuff) GUO predicate to be read as a metaphor, write it as LE GE PO stuff GUO predicate, a single abstract description. This allows us to avoid the need for double closures, first of an abstract predicate then of the abstract description, with which the sister language is afflicted.

```
arg1 <- (abstractn / (LIO (freemod)? descpred (guea)?) /
(LIO (freemod)? argument1 (guua)?) / (LIO (freemod)? mex (gap)?) /
(LIO stringnospaces) / LAO / LANAME /
(descriptn guua? (((((comma)? CI (comma)?) /
(comma &(nonamemarkers) !(AcronymicName))) name)))? /
LANAME2 / LIU1 / LIE / LI)
```

```
arg1a <- ((DA / TAI / arg1 / (GE (freemod)? arg1a)) (freemod)?)
```

arg1 is a general construction of arguments. Note that LANAME is read by preference to a description and LANAME2 is read only after a description. Notice the construction appending a name to a description, with either an explicit pause or a **ci** marker, with a mere pause only before a non-acronymic name without false name markers.

arg1a adds the pronouns and a construction fronting an argument of this same class with **ge** whose uses I should study.

```

argmod1 <- (((__LWinit (N o) ([ ])*))?) ((JI (freemod)? predicate) /
  (JIO (freemod)? sentence) / (JIO (freemod)? uttAx) /
  (JI (freemod)? modifier) / (JI (freemod)? argument1))) /
  (((__LWinit (N o) ([ ])*))?) (((JIZA (freemod)? predicate) (guiza)?) /
  ((JIOZA (freemod)? sentence) (guiza)?) /
  ((JIOZA (freemod)? uttAx) (guiza)?) /
  ((JIZA (freemod)? modifier) (guiza)?) /
  (JIZA (freemod)? argument1 (guiza)?))) /
  (((__LWinit (N o) ([ ])*))?) ((JIZI (freemod)? predicate (guizi)?) /
  (JIOZI (freemod)? sentence (guizi)?) / (JIOZI (freemod)? uttAx (guizi)?) /
  (JIZI (freemod)? modifier (guizi)?) / (JIZI (freemod)? argument1 (guizi)?))) /
  (((__LWinit (N o) ([ ])*))?) ((JIZU (freemod)? predicate (guizu)?) /
  (JIOZU (freemod)? sentence (guizu)?) / (JIOZU (freemod)? uttAx (guizu)?) /
  (JIZU (freemod)? modifier (guizu)?) / (JIZU (freemod)? argument1 (guizu)?)))

argmod <- (argmod1 ((A1 (freemod)? argmod1))* gui?)

arg2 <- (arg1a freemod? ((argmod))*

arg3 <- (arg2 / (mex (freemod)? arg2))

indef1 <- (mex (freemod)? descpred)

indef2 <- (indef1 (guua)? ((argmod))*

indefinite <- indef2

arg4 <- ((arg3 / indefinite) ((ZE2 (freemod)? (arg3 / indefinite)))*

arg5 <- (arg4 / (KA (freemod)? argument1 (freemod)? KI (freemod)? argx))

argx <- (((N01 (freemod)?))* ((LAE) (freemod)?))* arg5)

arg7 <- (argx (freemod)? ((ACI (freemod)? argx))?)

arg8 <- (!(GE) (arg7 (freemod)? ((A1 (freemod)? arg7))*))

```

```
argument1 <- (((arg8 (freemod)? AGE (freemod)? argument) / arg8)
((GUU (freemod)? argmod))*)
```

```
argument <- (((NO1 (freemod?))* ((DIO (freemod?))* argument1)
```

The full construction of arguments is described in the reference grammar.

Note the ability to attach a final argument modifier at the very top level of an untagged argument (`argument1`), in addition to attaching such at very low levels. This is a quite distinct use of **guu**.

```
argumentA <- argument
```

```
argumentB <- argument
```

```
argumentC <- argument
```

```
argumentD <- argument
```

These classes are attached to the first, second, third, fourth and fifth non-case-tagged arguments in a `terms` sequence of modifiers and possibly case tagged arguments. They refer normally to second, third, fourth, etc. arguments of predicates.

```
argxx <- (&(((NO1 (freemod?))* DIO)) argument)
```

A case tagged argument.

```
term <- (argument / modifier)
```

```
modifiers <- (modifier (((freemod)? modifier))*)
```

```
modifiersx <- ((modifier / argxx) (((freemod)? (modifier / argxx))*))
```

```

terms<-((modifiersx? argumentA
(freemod? modifiersx)? argumentB?
(freemod? modifiersx)? argumentC?
(freemod? modifiersx)? argumentD?)/
modifiersx)

```

terms are arguments or modifiers.

modifiers is the class of sequences of modifiers.

modifiersx is the class of sequences of modifiers and case tagged arguments.

terms is the class of sequences of terms containing no more than four non case tagged arguments.

```

firstarg <- (argument1 ((modifiersx (freemod)?))?) ((GIO (freemod)? terms))? pre

```

```

terms <- (((modifiersx)? argumentA (((freemod)? modifiersx))?) (!(firstarg) arg

```

This is the different treatment of class `terms` in the alternative parser, enabling detection and rejection of untagged arguments which would start a sentence.

```

word <- (arg1a / indef2)

```

```

words1 <- (word ((ZEIA word))*

```

```

words2 <- (word ((ZEIO word))*

```

```

wordset1 <- ((words1)? LUA)

```

```

wordset2 <- ((words2)? LUO)

```

Internals of the construction of ordered and unordered lists.

```

termset1 <- ((terms) /
  (KA (freemod)? termset2 (freemod)? KI (freemod)? termset1))

termset1 <- (((modifiersx)* (!(firstarg) terms) / (KA (freemod)? termset2 (fre

termset2 <- (termset1 ((guu &(A1)))? ((A1 (freemod)? termset1 ((guu &(A1)))?))*

termset <- ((terms (freemod)? GO (freemod)? barepred) / termset2 / guu)

```

The class of terms after a predicate. A `termset1` is a `terms` possibly closed with `guu`, or a `termset2` forethought connected to a `termset1`. A `termset2` is an afterthought connected sequence of `termset1`'s.

A second form is given for rule `termset1`, used by the alternative parser.

A `termset` has the final optional attachment of a `go barepred` clause intended to modify the predicate to which the termset is attached. To use this construction, care must be taken to close the last argument in the `termset2` component so that the `go barepred` does not unintentionally attach to it. A `termset` can also be a solitary `guu`, which turns out to be quite useful.

```

barepred <- barepred <- (sentpred (freemod)? (((termset (guu)?) / (guu &(termse

markpred <- (PA1 (freemod)? barepred)

```

Basic sentence predicates with termsets optionally attached, before logical connection.

`barepred` adds the optional termset to a sentence predicate.

`markpred` adds a tense or `ga`.

An additional class `kekpred` was eliminated here 5/1/17 as part of the final elimination of distinctions between sentence and description predicates motivated by the already eliminated rule that metaphors could not have a forethought connected head modifier.

```

backpred1 <- (((N02 (freemod)?))* (barepred / markpred))

```



```

backpred <- (((backpred1 ((ACI (freemod)? backpred1))+
  (freemod)? (((termset (guu)?) / (guu &(termset))))?))
  (((ACI (freemod)? backpred))+ (freemod)? (((termset (guu)?)
  / (guu &(termset))))?))?) / backpred1)

predicate2 <- (!(GE) (((backpred ((A1 !(GE) (freemod)? backpred))+
  (freemod)? (((termset (guu)?) / (guu &(termset))))?))
  (((A1 (freemod)? predicate2))+ (freemod)?
  (((termset (guu)?) / (guu &(termset))))?))?) / backpred))

predicate1 <- ((predicate2 AGE (freemod)? predicate1) / predicate2)

```

Afterthought logical connection of main sentence predicates (with termsets attached), first with ACI connectives, then with A connectives (both of which are left grouping) then with the top level AGE connectives (and seldom used) which are right grouping.

The classes `backpred` and `predicate2` which implement ACI and A afterthought connections also implement the attachment of shared final termsets: recall that the predicates being linked with ACI or A connectives already contain termsets; the additional termsets in the rules are logically shared with the entire preceding logically connected predicate (at the appropriate level). It is necessary to close the final predicate being linked (solitary `guu` as a termset is useful for this) to ensure that a termset is really attached to the entire predicate and not just the last component.

The solution here for logically shared final termsets is not as general as the solution in `trial.85` but it is most unlikely that anyone will ever say anything (except maliciously) that nests the attachment of shared final termsets in a way it cannot handle. The `trial.85` solution is very elegant but relies on a left recursion in a way difficult to implement in a PEG.

Shared final termsets are not a feature of the right-grouping AGE connections. Notice that initial `ge` must be restricted in the `predicate2` formation to avoid ambiguity with the AGE connectives. This is an example of an actual ambiguity which went undetected in 1989 Loglan because the lexicography was not visible to the grammatical parser. This means that we do not have top level sentence predicates which are `ge`-initial, which is fine as there is no reason to use `ge` in top level sentence predicates.

```
identpred <- (((NO1 (freemod)?))* (BI (freemod)? termset guu?))
predicate <- (predicate1 / identpred)
```

The most general predicate class. We preserve the feature of the grammar that the identity predicates like **bi** are hard to logically link with other predicates. One *can* do it, with forethought connection. We think this is sound.

```
subject <- (((modifiers (freemod)?))? ((argxx subject) /
(argument ((modifiersx (freemod)?)?)))
```

A grammatical subject is a sequence of terms which contains at least one argument, and no more than one un-case-tagged argument. We impose the restriction that such sequence of terms be used in certain contexts where NB3 actually supports such a restriction.

```
gasent1 <- (((NO1 (freemod)?))*
(PA1 (freemod)? barepred ((GA2 (freemod)? subject)?))
```

```
gasent2<-((NO1 freemod?)*
(PA1 freemod? sentpred modifiers?
(GA2 freemod? subject freemod?
GIO? freemod? terms?)))
```

```
gasent <- (gasent2 / gasent1)
```

The subject-deferred sentence construction. Either terms containing exactly one initial un-case-tagged argument or all terms may be deferred (marked with **ga**) as described in the reference grammar. If all terms are in the **ga** clause, the first one may optionally be separated from the sutori ones by **gio**. Notice that a tensed predicate by itself is a **gasent**, not an imperative (an observative): the final **ga** subject is optional in the class **gasent1**. This is a change from 1989 Loglan.

```
statement <- (gasent / (modifiers (freemod)? gasent) /
(subject (freemod)? ((GIO (freemod)? terms))?)? predicate))
```

```
statement <- (gasent / (subject (freemod)? (((GIO (freemod)? terms) / (GAA (fre
```

This is the basic statement class. One looks first for a **gasent**, then for a **gasent** with fronted modifiers (that all fronted terms in such a sentence are modifiers is not enforced in the 1989 Loglan grammar, but the intention is stated in NB3), then for an SVO sentence. We add the option of inserting additional un-case-tagged arguments before the predicate after the new marker **gio** (allowing SOV forms but requiring an explicit signal of this intention). Explicit marking of SOV sentences appears to be useful, because the overwhelming majority of such sentences in existing text appear to result from sentence construction errors.

The second form is the one used by the alternative parser, which forbids leading modifiers on a **gasent**; the particle **gaa** can be used in place of **gio** (and optionally without following terms) to signal that the last term before the subject should *not* be excluded from a termset which wants it (this allows the main verb to close the subject, as it did in previous versions).

```
keksent <- (((NO1 (freemod)?))*
((KA (freemod)? sentence (freemod)? KI (freemod)? uttA1) /
(KA sentence (freemod)? KI (freemod)? uttA1) /
(KA (freemod)? headterms (freemod)? sentence (freemod)?
KI (freemod)? uttA1)))
```

Forethought connected sentences. Note the extreme freedom of form of the last entry (class **uttA1**).

```
neghead <- (NO1 freemod? gap/NO2 PAUSE)
```

```
sen1 <- (neghead freemod?)* ((modifiers (freemod)? !(gasent) predicate) /
statement / predicate / keksent)
```

```
sen1 <- (((neghead (freemod)?))* (statement / predicate / keksent))
```

This sentence class, which we call “logical unit sentences”, includes imperatives which consist of a predicate with fronted modifiers, then statements, as above, then predicates (read as imperatives), then forethought connected sentences. Notice that tensed predicates with or without fronted modifiers are read as subject-deferred sentences (observatives) not as imperatives. Added the ability to negate sen1’s with sentence scope. Thus **neghead** is moved to this point. Restricted the possibility of closing a neghead with a pause to the situation where it makes no semantic difference, because the negation will negate the first term if it does not negate the entire sentence anyway (5/7)

The second form is used by the alternative parser, forbidding initial modifiers in imperatives.

```
sentence <- (sen1 ((ICA (freemod)? sen1))*)
```

A sentence is a logical unit sentence or a chain of logically linked logical unit sentences linked with ICA connectives.

```
headterms <- ((terms GI))+
```

```
uttAx <- (headterms (freemod)? sentence (giuo)?)
```

A list of terms may be fronted using **gi**: notice that these will be shared (as final arguments) by all components of the following sentence if it is not a logical unit sentence. The structure can be closed with a **giuo** for this reason.

We disagree with the 1989 Loglan assignment of positions to the initial terms: this is discussed in the reference grammar (and does not affect parses).

```
HUE0 <- ([Hh] UE)
```

```
invvoc<-(([ ]* HUE0 (CANCELPAUSE/([ ]* &C1)) name)/
(HUE !badspaces freemod? descpred guea? (((comma? CI comma?)/
(comma &nonamemarkers !AcronymicName)) name)?)/
(HUE !badspaces freemod? statement giuo?)/
(HUE !badspaces freemod? termset1 guu?)/
([ ]* HUE0 ((',' [ ]+)/([ ]* &V1)) name)/
(HUE stringnospacesclosedblock))
```

```
freemod<-((NOUI/(SOI freemod? descpred guea?)/
DIE/(NO1 DIE)/(KIE comma? utterance0 comma? KIU)/
(KIE2 comma? utterance0 comma? KIU2)/
invvoc/voc/CANCELPAUSE/(comma !(!namemarker Name)))/
JO/UI1/([ ]* '...' ([ ]* &letter)?)/([ ]* '--' ([ ]* &letter)?)) freemod?)
```

This is the ubiquitous class of free modifiers, discussed in the reference grammar. Notice that these are allowed in most medial positions in grammar rules, very seldom in initial position, occasionally in final position.

Notice that inverse vocatives, which have now been pulled out as a separate class, have usual features for a construction involving name markers. The `termset1` class is used when terms are used as an inverse vocative to allow closure with `guu`, which turns out to be important in existing text. Foreign names are specially guarded in inverse vocative constructions (notice the fancy class used).

12/2/17 use of parentheses with `kie` and `kiu` supported.

```
uttA <- ((A1 / mex) (freemod?))
```

```
uttA1 <- ((sen1 / uttAx / links /
linkargs / argmod / (modifiers (freemod)? keksent) /
terms / uttA/ NO1) (freemod)? (period?))
```

`uttA` is a class of fragmentary utterances which occur only as answers.

`uttA1` is a very large class of utterances including fragments of various kinds used as answers and also including the logical unit sentences and the sentences with fronted arguments (but not the general sentence class). Note the odd permission to use this class as the final component of a forethought connected sentence (above).

This is the lowest level utterance which can include terminal punctuation at the end (class `period`).

```
uttC <- ((neghead uttC) / uttA1)
```

`neghead` is an occurrence of **no** negating an entire utterance. This can be set off from the following utterance by **gu** or by a significant pause (one of the two very limited surviving instances of semantically significant pauses, and the only surviving instance of pause/GU equivalence in this grammar).

`uttC` is an optionally thus negated `uttA1`.

```
uttD <- ((sentence (period)? !(ICI) !(ICA)) /
  (uttC ((ICI (freemod)? uttD))*))
```

```
uttE <- (uttD ((ICA (freemod)? uttD))*)
```

The basic idea here is that an `uttD` is an `uttC` or a sequence of `uttC`'s linked with `ICI` connectives, and then an `uttE` is an `uttD` or a sequence of `uttD`'s linked with `ICA` connectives, but we made a slight modification: a sentence (with possible terminal punctuation) will be parsed as a single `uttD` rather than having two parses, as a sentence and as an `uttE`. This has no effect on the range of legal utterances but makes the parses of sentences as top level utterances more informative.

```
uttF <- (uttE ((I (freemod)? uttE))*)
```

```
utterance0 <- (!(GE) (!(PAUSE) freemod (period)? utterance0) /
  (!(PAUSE) freemod (period)?) / (uttF IGE utterance0) / uttF /
```

```
(I (freemod)? (uttF)?) / (I (freemod)? (period)?) /  
(ICA (freemod)? uttF)) ((&(I) utterance0))?)
```

```
utterance <- (!(GE) (!(PAUSE) freemod (period)? utterance) /  
(!(PAUSE) freemod (period)? ((&(I) utterance))? end) /  
(uttF IGE utterance) /  
(I (freemod)? (period)? ((&(I) utterance))? end) /  
(uttF ((&(I) utterance))? end) /  
(I (freemod)? uttF ((&(I) utterance))? end) /  
(ICA (freemod)? uttF ((&(I) utterance))? end)))
```

uttF is inhabited by uttD's and I (incl. IPA/IKOU) linked uttF's.
Changed uttF to flat grouping (interpreted as left grouping) 5/7.

utterance0 is inhabited by full utterances: this is the form which can
appear embedded in other utterances, in LI quotes or in KIE parentheses.
The top level class utterance is characterized by termination with class end.

12 Appendix: The Trial.85 Grammar

```
/* GRAMMAR 84 Loglan grammar as of Jun97 Trial.84
Apr99
0 conflicts
Copyright (C) 1982, 1984, 1986-1997 by The Loglan Institute, Inc.
```

Created in Jan-Feb 82 from JSP's Aug 81 grammar by SWL & JCB,
Modified in Mar 82, Dec 83, Mar 84, and Dec 86 - Jun 87 by JCB. and in 1987-99
Trial 84a (May 99) is a test for using GO in a different fashion
Trial.84 was created in Apr99 to incorporate an extension of GE to permit it to
group numbers e.g. to ge tecu mrenu, to add 'hi' which had been inadvertently
omitted from mex, and to replace feu in PA with fea, and to restore feu to UI.
zeu was added as an allolex of ze for ordered linking of arguments
Trial 83 was created in Jan 98 to modify the handling of vocatives as UI and
modify the use of HOI as an argument.
Trial.82 was created in Jun 97 to incorporate a number of modifications to the
A correction for when ge and go are combined with multiple predicates. The prev
A change to allow prenex quantifiers to extend over an afterthought compound se
A change in the relative binding strength of CI and ZE,(ZE1) to permit sentence
Addition of the new PA words, including the MIA subjunctive and the new trial s
Altering of the DA words to handle the new personal pronoun set.
Introduction of a MO lexeme, to allow MO to be both DA and NI, and likewise for
Putting NIRO into the PREDA lexeme.
Allowing NI+UI to be parsed separately as UI.
Changing ZE2 to parse as A4.

Trial 81 was created in Oct 94 to correct a typo in the utterance category. Th
Trial 80 was created in Dec 94, include luo and lou, mea, nuo, fuo, and juo. T
Trial 79 was created in Nov 93, allow JUE phrases without a corresponding JE ph
disallow GUU A TERMSET, adjust the word list lexeme to require pauses, and allo
Trial 78 was created in Feb 93 to eliminate TEI, split GUUs to allow for argmod
Trial 77a was created to test different BI and LA structures
Trial 77 was created in Jul 92 to add niu, change some other LWs, the parse of
Trial 76a was created in Jul 91 to add the possibility of multiple JIs to indef
Trial 76 is based on Trial 75. The changes are to accept terms and descpreds af

Trial 75 is based on Trial 74. The main change is to allow initial kekpreds in
Trial 74 is based on Trial 73. It moves kekpred to barepred, and makes other c
Trial 73 is based on Trial 72. It incorporates the Rice changes, including soi
Trial 72 is based on Trial 71. It has extended the vocative changes to bring a
Trial 71 is based on Trial 70, which was the original distribution version. It
Trial 70 is based on Trial 69 and incorporates all changes required by the issu
Trial 69 is based on Trial 68 and incorporates the changes of Nov 88
Trial 69b is a temporary version for testing modifications only. Yet to do is
Trial 69c is a test for separating ICA from I and making RA and NI essentially
Mar 89: Removed all free(head)mods from grammar proper in preparation for separ
Feb 89: Allowed for lao + gobbling of Linneans up to a comma. Allowed apostrop
Feb 89: Inserted new comma lexemes and separated pause(comma) from gu. These v
Feb 89: inserted ZE2 before argsign
Feb 89: Corrected to allow prenex modifiers to precede keksentences without GOI
Feb 89: Moved gu from termset1 to termset. Hoi redro nu herfa, nenkaa fails wi
3Nov88, Added a number of other missing words, and incorporated changes in disc
Added LIO DA and LE DA in Arg 1 and fixed for new case tags.
T68 is based on T67. freemods(freemods) are incorporated into the grammar elimi
T67 is based on T65&T66. Machine lexemes have been replaced by additional lexem
T65 was based on T64 and put the BUA Lexeme into PREDA by putting 'bua/bui'into

A new preparser, PP729, was built to go with this grammar.

In addition, M8 through M12 were renamed M7 through M11, in both grammar and pr

Later, on 2 June, it was noticed that the GI/GOI distinction is no longer neces

SOME DEFERRED PROBLEMS for study later or for solution before Turnover:-

- a) There are some problems with multiple negation of modifiers. Also of some ke
- b) Study how prenexes interact with sentence-kekking and term-advancing. It's n
- c) The preparser needs to be changed so that free mods before PAUSE gobble righ
- d) Check for redundant M-Lexemes; see comments to T33. Done.RAM
- e) Study why my efforts to negate arguments have all failed. Why? Negating modifiers, a very similar structure, was successful. Done. RAM
- f) <uttA => TAI | mex> are temporary allograms of this grameme only to permit s

PERMANENT COMMENT (do not erase this until SWL has dealt with it): T46x collaps
I've restored JIO to the lexicon and moved on. But I would recollapse JIO into
*/

%token A1

%{ /*: a1 zea */

%} /* used for A when connecting predicates */

%token A2

%{ /*: a2 */

%} /* used for A when connecting linkargs or modifiers */

%token A3

%{ /*: a3 */

%} /* used for A when connecting argmods */

%token A4

%{ /*: ha a e o u */

%} /* also CPDs anoi, apa, noanoi, etc. Used for all other A */

```
%token ACI
%{/*: */
%} /* recognized by CPD-lexer */
```

```
%token AGE
%{/*: */
%} /* recognized by CPD-lexer. */
```

```
%token BI
%{/*: bi bia bie cie cio */
%}
```

```
%token BAD
%{/*: */
%}
```

```
%token CA
%{/*: ca ce co cu */
%} /* also CPDs noca, canoi, nocanoi, etc. */
```

```
%token CI
%{/*: ci */
%}
```

```
%token CUI
%{/*: cui */
%}
```

```
%token DA
%{/*: ba be bo bu da de di do du mi tu mu ti ta tao tio tua mio miu muo muu to
%}
```

```
%token DIE
%{/*: die fie kae nue rie */
%}
```

```
%token DIO
%{/*: beu cau dio foa kao jui neu pou goa sau veu zua zue zui zuo zuu
```

```
lae lue */
%}
```

```
%token DJAN
%{/*: */
%} /* all C-final words found by lexer */
```

```
%{#define END 0
/*: . */
%}
```

```
%token FI
%{/*: fi */
%}
```

```
%token GA2
%{/*: ga */
%}
```

```
%token GE
%{/*: ge */
%}
```

```
%token GE2
%{/*: ge2 */
%}
```

```
%token GEU
%{/*: geu */
%}
```

```
%token GI
%{/*: gi goi */
%}
```

```
%token GO
%{/*: go */
%}
```

```
%token GU
%{/*: gu */
%}
```

```
%token GUE
%{/*: gue */
%}
```

```
%token GUI
%{/*: gui */
%}
```

```
%token GUO
%{/*: guo */
%}
```

```
%token GUU1
%{/*: guu */
%}
```

```
%token GUU2
%{/*: guu2 */
%}
```

```
%token HOI
%{/*: hoi */
%}
```

```
%token HU
%{/*: hu */
%} /* used only by CPD-lexer to find nahu-CPDs; otherwise with DA */
```

```
%token I
%{/*: i */
%} /* also CPDs ifa, inusoa, etc. */
```

```
%token ICA
```

```
%{/*: */
%} /* all eeskeks, recognized by lexer */

%token ICI
%{/*: */
%} /* ici & icaci-type words, all recognized by CPD-lexer */

%token IE
%{/*: ie */
%}

%token IGE
%{/*: */
%} /* ige & icage-type words, all recognized by CPD-lexer */

%token JE
%{/*: je */
%}

%token JI
%{/*: ji ja jie jae pe */
%}

%token JIO
%{/*: jio jao */
%}

%token JO
%{/*: jo */
%} /* also CPDs rajo, tojo, etc. */

%token JUE
%{/*: jue */
%}

%token KA1
%{/*: ka1 */
%} /* used for KA when connecting linkargs */
```

```
%token KA2
%{/ *: ka2 */
%} /* used for KA when connecting predicates */

%token KA3
%{/ *: ka ke ko ku */
%} /* also CPDs kanoi, nuku, nukunoi, kouki, nukouki,etc. For the rest */

%token KOU
%{/ *: kou moi rau soa */
%} /* these are pa words separated out for the lexer */

%token KI
%{/ *: ki */
%} /* also the CPD kinoi */

%token KIE
%{/ *: kie */
%}

%token KIU
%{/ *: kiu */
%}

%token LAO
%{/ *: lao */
%}

%token LAU
%{/ *: lau lou */
%}

%token LE
%{/ *: le la lo lea leu loe lee laa */
%}

%token LEPO
```

```
%{/*: */
%} /* recognized by CPD-lexer*/
```

```
%token LI
%{/*: li */
%}
```

```
%token LIE
%{/*: lie */
%}
```

```
%token LIO
%{/*: lio */
%}
```

```
%token LIU
%{/*: liu lii niu */
%}
```

```
%token LU
%{/*: lu */
%}
```

```
%token LUA
%{/*: lua luo */
%}
```

```
%token SOI
%{/*: soi */
%}
```

```
%token MA
%{/*: ma si */
%} /* to recognize initial vowels in acronyms, NI otherwise */
```

```
%token ME
%{/*: me mea */
%}
```



```
%token MO
%{/*: mo */
%} /* to recognize MO as DA when not following a NI */
```

```
%token NI
%{/*: ho ni ne to te fo fe vo ve pi re ru sa se so su kua gie giu hi hie hiu
%} /* also CPDs neni, nenisei, iesu, ietoni, etc. */
```

```
%token NO1
%{/*: no1 */
%} /* used for NO + mod shown by PA */
```

```
%token NO2
%{/*: no2 */
%} /* used for NO + markpred shown by PO, ZO or PA1 */
```

```
%token NO3
%{/*: no3 */
%} /* used for NO + argument */
```

```
%token NO4
%{/*: no */
%} /*For all other no's*/
```

```
%token NOI
%{/*: noi */
%}
```

```
%token NU
%{/*: nu fu ju nuo fuo juo */
%} /* also CPDs nufu, nufuju, nuto (= nu), nute (=fu), nufo (=ju), nufe,
nuso, etc. */
```

```
%token PA1
%{/*: pa1 */
%} /* used for PA and GA when inflecting a predicate */
```

%token PA2

{/*: va vi vu pa na fa gia gua pia pua nia nua biu fea fia fua via vii viu c

} /* also CPDs pana, pazi, pacenoina, etc. For the rest of PAs*/

%token PAUSE

{/*: , # */

}

%token PO

{/*: po pu */

}

%token PREDA

{/*: he dua dui bua bui */

} /* all preda-forms words; also all pred-wds found by lexer, CPDs like rari, nenira, sutori, etc.; also acronyms like ebai, baicai, ebaicai, ebaiocai, haitosaiifo, etc., */

%token RA

{/*: ra ri ro */

}

%token HUE

{/*: hue */

}

%token SUE

{/*: sue sao */

}

%token TAI

{/*: gao */

}

/* forms like ama bai cai tai tei are recognized by the lexer; CPDs like baicai, ebaicai, ebaiocai, haitosaiifo, etc., belong to PREDA */

%token UI

```

%{/ *: ua ue ui uo uu oa oe oi ou ia ii io iu ea ei eo eu ae ai ao au bea b
%} /* also CPDs nahu, vihu, kouhu, duohu, nusoahu, etc. */

%token ZE2
%{/ *: ze zeu */
%} /* used for ZE + argsign */

%token ZI
%{/ *: zi za zu */
%} /* used by the preparser to recognize pazi-CPDs and acronymic PREDA's */

%token Z0
%{/ *: zo */
%} /* used by the preparser to recognize acronymic PREDA's; otherwise zo would

%start utterance

%{
#define YYDEBUG 1
%}

%%

err : error {yyerrok;}
;

guo : GUO {$$=NodeY1 ("guo", &$1);}
| GU {$$=NodeY1 ("guo", &$1);}
| err {$$=NodeA ("guo",1,LeafI (GUO,"0"));}
;

gui : GUI {$$=NodeY1 ("gui", &$1);}
| GU {$$=NodeY1 ("gui", &$1);}
| err {$$=NodeA ("gui",1, LeafI(GUI,"0"));}
;

gue : GUE {$$=NodeY1 ("gue", &$1);}
| GU {$$=NodeY1 ("gue", &$1);}

```

```

| err {$$=NodeA ("gue",1,LeafI (GUE,"0"));}
;

guu : GUU1 {$$=NodeY1 ("guu", &$1);}
| GU {$$=NodeY1 ("guu", &$1);}
| err {$$=NodeA ("guu",1,LeafI (GUU1,"0"));}
;

lua : LUA {$$=NodeY1 ("lua", &$1);}
| err {$$=NodeA ("lua",1,LeafI (LUA,"0"));}
;

geu : GEU {$$=NodeY1 ("geu", &$1);}
| err {$$=NodeA ("geu",1,LeafI(GEU,"0"));}
;

gap : PAUSE {$$=NodeY1 ("gap", &$1);}
| GU {$$=NodeY1 ("gap", &$1);}
| err {$$=NodeA("gap",1,LeafI(PAUSE,"0" ));}
;

juelink : JUE argument {$$=NodeY2 ("juelink", &$2);}
;

links1 : juelink {$$=NodeY1 ("links1", &$1);}
| juelink links1 gue {$$=NodeY3 ("links1", &$3);}
;

links : links1 {$$=NodeY1 ("links", &$1);}
| links A2 links1 {$$=NodeY ("links", 3, &$3);}
| KA1 links KI links1 {$$=NodeY ("links", 4, &$4);}
;

jelink : JE argument {$$=NodeY2 ("jelink", &$2);}
;

linkargs1 : jelink gue {$$=NodeY2 ("linkargs1", &$2);}
| jelink links gue {$$=NodeY3 ("linkargs1", &$3);}

```

```

;

linkargs : linkargs1 {$$=NodeY1 ("linkargs", &$1);}
| linkargs A2 linkargs1 {$$=NodeY ("linkargs",3, &$3);}
| KA1 linkargs KI linkargs1 {$$=NodeY ("linkargs", 4, &$4);}
;

predunit1 : PREDA {$$=NodeY1 ("predunit1", &$1);}
| SUE {$$=NodeY1 ("predunit1", &$1);}
| NU PREDA {$$=NodeY2 ("predunit1", &$2);}
| GE descpred geu {$$=NodeY3 ("predunit1", &$3);}
| NU GE despredE geu {$$=NodeY ("predunit1", 4, &$4);}
| ME argument gap {$$=NodeY3 ("predunit1", &$3);}
;

predunit3 : predunit2 {$$=NodeY1 ("predunit3", &$1);}
| predunit2 linkargs {$$=NodeY2 ("predunit3", &$2);}
;

predunit2 : predunit1 {$$=NodeY1 ("predunit2", &$1);}
| NO4 predunit2 {$$=NodeY2 ("predunit2", &$2);}
;

predunit : predunit3 {$$=NodeY1 ("predunit4", &$1);}
| PO predunit3 {$$=NodeY2 ("predunit4", &$2);}
;

despredA : predunit {$$=NodeY1 ("despredA", &$1);}
| kekpredunit {$$=NodeY1 ("despredA", &$1);}
| predunit CI despredA {$$=NodeY3 ("despredA", &$3);}
;

kekpredunit: NO4 kekpredunit {$$=NodeY2 ("kekpredunit:",&$2);}
| KA2 predicate KI predicate {$$=NodeY ("kekpredunit:",4,&$4);}
;

despredB : despredA {$$=NodeY1 ("despredB", &$1);}
| CUI despredC CA despredB {$$=NodeY ("despredB", 4, &$4);}

```

```
;  
  
despredC : despredB {$$=NodeY1 ("despredC", &$1);} | despredC despredB {$$=NodeY2 ("despredC", &$2);} ;  
  
despredD : despredB {$$=NodeY1 ("despredD", &$1);} | despredD CA despredB {$$=NodeY3 ("despredD", &$3);} ;  
  
despredE : despredD {$$=NodeY1 ("despredE", &$1);} | despredE despredD {$$=NodeY2 ("despredE", &$2);} ;  
  
descpred : despredE {$$=NodeY1 ("descpred", &$1);} | despredE GO descpred {$$=NodeY3 ("descpred", &$3);} ;  
  
senpred1 : predunit {$$=NodeY1 ("senpred1", &$1);} | predunit CI senpred1 {$$=NodeY3 ("senpred1", &$3);} ;  
  
senpred2 : senpred1 {$$=NodeY1 ("senpred2", &$1);} | CUI despredC CA despredB {$$=NodeY ("senpred2", 4, &$4);} ;  
  
senpred3 : senpred2 {$$=NodeY1 ("senpred3", &$1);} | senpred3 CA despredB {$$=NodeY3 ("senpred3", &$3);} ;  
  
senpred4 : senpred3 {$$=NodeY1 ("senpred4", &$1);} | senpred4 despredD {$$=NodeY2 ("senpred4", &$2);} ;  
  
sentpred : senpred4 {$$=NodeY1 ("sentpred", &$1);} | senpred4 GO barepred {$$=NodeY3 ("sentpred", &$3);} ;
```

```

mod1 : PA2 gap {$$=NodeY2 ("mod1", &$2);}
| PA2 argument gap {$$=NodeY3 ("mod1", &$3);}
;

mod : mod1 {$$=NodeY1 ("mod", &$1);}
| NO1 mod1 {$$=NodeY2 ("mod", &$2);}
;

kekmod : KA3 modifier KI mod {$$=NodeY ("kekmod", 4, &$4);}
| NO3 kekmod {$$=NodeY2 ("kekmod", &$2);}
;

modifier : mod {$$=NodeY1 ("modifier", &$1);}
| kekmod {$$=NodeY1 ("modifier", &$1);}
| modifier A2 mod {$$=NodeY ("modifier", 3, &$3);}
;

name : DJAN {$$=NodeY1 ("name", &$1);}
| name CI DJAN {$$=NodeY3 ("name", &$3);}
| name predunit {$$=NodeY2 ("name", &$2);}
| name DJAN {$$=NodeY2 ("name", &$2);}
;

mex : NI {$$=NodeY1 ("mex", &$1);}
| mex NI {$$=NodeY2 ("mex", &$2);}
;

descriptn : LE descpred {$$=NodeY2 ("descriptn", &$2);}
| LE mex descpred {$$=NodeY3 ("descriptn", &$3);}
| LE arg1 descpred {$$=NodeY3 ("descriptn", &$3);}
| LE mex arg1a {$$=NodeY3 ("descriptn", &$3);}
| GE2 mex descpred {$$=NodeY3 ("descriptn", &$3);}
;

voc : HOI descpred gap {$$=NodeY3 ("voc", &$3);}
| HOI argument gap {$$=NodeY3 ("voc", &$3);}
/* | HOI argument {$$=NodeY2 ("voc", &$2);}*/
| HOI gap {$$=NodeY2 ("voc", &$2);}

```

```

/* | name gap {$$=NodeY2 ("voc", &$2);}*/
;

arg1 : LIO mex gap {$$=NodeY3 ("arg1", &$3);}
| LIO descpred gap {$$=NodeY3 ("arg1", &$3);}
| LIO term gap {$$=NodeY3 ("arg1", &$3);}
| LE name gap {$$=NodeY3 ("arg1", &$3);}
| descriptn gap {$$=NodeY2 ("arg1", &$2);}
| descriptn name gap {$$=NodeY3 ("arg1", &$3);}
| LI utterance LU {$$=NodeY3 ("arg1", &$3);}
| LI LU {$$=NodeY2 ("arg1", &$2);}
| LIU {$$=LexLiu(&$1);}
| LIE {$$=LexLie(&$1);}
| LAO {$$=LexLao( &$1);}
| LEPO uttAx guo {$$=NodeY3 ("arg1",&$3);}
| LEPO sentence guo {$$=NodeY3 ("arg1",&$3);}
;

arg1a : DA {$$=NodeY1 ("arg1a", &$1);}
| TAI {$$=NodeY1 ("arg1a", &$1);}
| arg1 {$$=NodeY1 ("arg1a", &$1);}
| name gap {$$=NodeY2 ("arg1a", &$2);}
| GE2 arg1a {$$=NodeY2 ("arg1a", &$2);}
;

argmod1 : JI argument {$$=NodeY2 ("argmod1", &$2);}
| JI modifier {$$=NodeY2 ("argmod1", &$2);}
| JI predicate gui {$$=NodeY3 ("argmod1", &$3);}
| JIO uttAx gui {$$=NodeY3 ("argmod1", &$3);}
| JIO sentence gui {$$=NodeY3 ("argmod1", &$3);}
;

argmod : argmod1 {$$=NodeY1 ("argmod", &$1);}
| argmod A3 argmod1 gap {$$=NodeY ("argmod", 4, &$4);}
;

arg2 : arg1a {$$=NodeY1 ("arg2", &$1);}

```



```
| arg2 argmod gap {$$=NodeY3 ("arg2", &$3);}
;

arg3 : arg2 {$$=NodeY1 ("arg3", &$1);}
| mex arg2 {$$=NodeY2 ("arg3", &$2);}
;

indef1 : mex descpred {$$=NodeY2 ("indef1", &$2);}
;

indef2 : indef1 gap {$$=NodeY2 ("indef2", &$2);}
| indef2 argmod gap {$$=NodeY3 ("indefinite", &$3);}
;

indefinite : indef2 {$$=NodeY1 ("indefinite", &$1);}
;

arg4 : arg3 {$$=NodeY1 ("arg4", &$1);}
| indefinite {$$=NodeY1 ("arg4", &$1);}
/* | arg4 ZE2 arg3 {$$=NodeY3 ("arg4", &$3);}
| arg4 ZE2 indefinite {$$=NodeY3 ("arg4", &$3);} */
;

arg5 : arg4 {$$=NodeY1 ("arg5", &$1);}
| KA3 argument KI argx {$$=NodeY ("arg5", 4, &$4);}
;

arg6 : arg5 {$$=NodeY1 ("arg6", &$1);}
| DIO arg6 {$$=NodeY2 ("arg6", &$2);}
| IE arg6 {$$=NodeY2 ("arg6", &$2);}
;

argx : arg6 {$$=NodeY1 ("argx", &$1);}
| N03 argx {$$=NodeY2 ("argx", &$2);}
;

arg7 : argx {$$=NodeY1 ("arg7", &$1);}
| argx ACI arg7 {$$=NodeY3 ("arg7", &$3);}
```

```

;

arg8 : arg7 {$$=NodeY1 ("arg8", &$1);}
| arg8 A4 arg7 {$$=NodeY3 ("arg8", &$3);}
;

argument : arg8 {$$=NodeY1 ("argument", &$1);}
| arg8 AGE arg8 {$$=NodeY3 ("argument", &$3);}
| argument GUU2 argmod gap {$$=NodeY ("argument", 4, &$4);}
| LAU wordset {$$=NodeY2 ("argument", &$2);}
;

term : argument {$$=vocfind($1)?NodeY1 ("vocative", &$1):NodeY1 ("term", &$1);}
| modifier {$$=NodeY1 ("term", &$1);}
;

terms : term {$$=vocfind($1)?NodeY1 ("vocative", &$1):NodeY1 ("terms", &$1);}
| terms term {$$=NodeY2 ("terms", &$2);}
;

wordset : words lua {$$=NodeY2 ("wordset", &$2);}
| lua {$$=NodeY1 ("wordset", &$1);}
;

words : word {$$=NodeY1 ("words", &$1);}
| words word {$$=NodeY2 ("words", &$2);}
;

word : arg1a gap {$$=NodeY2 ("word", &$2);}
| NI gap {$$=NodeY2 ("word", &$2);}
| UI gap {$$=NodeY2 ("word", &$2);}
| PA2 gap {$$=NodeY2 ("word", &$2);}
| DIO gap {$$=NodeY2 ("word", &$2);}
| predunit1 gap {$$=NodeY2 ("word", &$2);}
| indef2 {$$=NodeY1 ("word", &$1);}
;

termset1 : terms guu {$$=NodeY2 ("termset1", &$2);}

```

```

;

termset2 : termset1 {$$=NodeY1 ("termset2", &$1);}
| termset2 A4 termset1 {$$=NodeY3 ("termset2", &$3);}
| KA3 termset2 KI termset1{$$=NodeY ("termset2",4,&$4);}
;

termset : termset2 {$$=NodeY1 ("termset", &$1);}
| terms GO barepred {$$=NodeY3("termset1", &$3);}
| guu {$$=NodeY1 ("termset", &$1);}
;

barepred : sentpred termset {$$=NodeY2 ("barepred", &$2);}
| kekpred termset {$$=NodeY2 ("barepred", &$2);}
;

markpred : PA1 barepred {$$=NodeY2("markpred", &$2);}
| PO gap sentence gap {$$=NodeY ("markpred", 4, &$4);}
| NO4 markpred {$$=NodeY2 ("markpred", &$2);}
;

backpred1 : barepred {$$=NodeY1 ("backpred1", &$1);}
| markpred {$$=NodeY1 ("backpred1", &$1);}
/* | kekpred {$$=NodeY1 ("backpred1", &$1);}*/
| NO2 backpred1 {$$=NodeY2 ("backpred1", &$2);}
;

backpred : backpred1 {$$=NodeY1 ("backpred", &$1);}
| backpred1 ACI backpred{$$=NodeY ("backpred", 3, &$3);}
;

bareekpred: barefront A1 backpred {$$=NodeY ("bareekpred", 3, &$3);}
;

barefront : barepred {$$=NodeY1 ("barefront", &$1);}
| bareekpred termset {$$=NodeY2 ("barefront", &$2);}
;

```

```
markekpred: markfront A1 backpred {$$=NodeY ("markekpred", 3, &$3);}
;
```

```
markfront : markpred {$$=NodeY1 ("markfront", &$1);}
| markekpred termset {$$=NodeY2 ("markfront", &$2);}
;
```

```
predicate2: barefront {$$=NodeY1 ("predicate2", &$1);}
| markfront {$$=NodeY1 ("predicate2", &$1);}
| NO2 predicate2 {$$=NodeY2 ("predicate2", &$2);}
;
```

```
predicate1: predicate2 {$$=NodeY1 ("predicate1", &$1);}
| predicate2 AGE predicate1 {$$=NodeY ("predicate1", 3, &$3);}
;
```

```
identpred : BI termset {$$=NodeY2 ("identpred", &$2);}
| NO4 identpred {$$=NodeY2 ("identpred", &$2);}
;
```

```
kekpred : kekpredunit {$$=NodeY1 ("kekpred", &$1);}
| kekpred despredD {$$=NodeY2 ("kekpred", &$2);}
;
```

```
predicate : predicate1 {$$=NodeY1 ("predicate", &$1);}
| identpred {$$=NodeY1 ("predicate", &$1);}
;
```

```
gasent : PA1 barepred GA2 terms {$$=NodeY ("gasent", 4, &$4);}
| NO2 gasent {$$=NodeY2 ("gasent", &$2);}
;
```

```
statement : gasent {$$=NodeY1 ("statement", &$1);}
| terms gasent {$$=NodeY2 ("statement", &$2);}
| terms predicate {$$=NodeY2 ("statement", &$2);}
;
```

```
keksent : KA3 sentence KI uttA1 {$$=NodeY ("keksent", 4, &$4);}
;
```

```
| KA3 gap sentence KI uttA1 {$$=NodeY ("keksent", 5, &$5);}
| KA3 headterms sentence KI uttA1{$$=NodeY ("keksent", 5, &$5);}
| NO3 keksent {$$=NodeY2 ("keksent", &$2);}
;
```

```
sen1 : predicate {$$=NodeY1 ("sen1", &$1);}
| statement {$$=NodeY1 ("sen1", &$1);}
| keksent {$$=NodeY1 ("sen1", &$1);}
;
```

```
sentence : sen1 {$$=NodeY1 ("sentence", &$1);}
| sentence ICA sen1 {$$=NodeY3 ("sentence", &$3);}
;
```

```
headterms : terms GI {$$=NodeY2 ("headterms", &$2);}
| headterms terms GI{$$=NodeY3 ("headterms", &$3);}
;
```

```
uttA : A4 {$$=NodeY1 ("uttA", &$1);}
| IE {$$=NodeY1 ("uttA", &$1);}
| mex {$$=NodeY1 ("uttA", &$1);}
;
```

```
uttAx : headterms sentence gap {$$=NodeY3 ("uttAx", &$3);}
;
```

```
uttA1 : uttA {$$=NodeY1 ("uttA1", &$1);}
| uttAx {$$=NodeY1 ("uttA1", &$1);}
| NO4 {$$=NodeY1 ("uttA1", &$1);}
| terms {$$=NodeY1 ("uttA1", &$1);}
| links {$$=NodeY1 ("uttA1", &$1);}
| linkargs {$$=NodeY1 ("uttA1", &$1);}
| sen1 {$$=NodeY1 ("uttA1", &$1);}
| argmod {$$=NodeY1 ("uttA1", &$1);}
| terms keksent {$$=NodeY2 ("uttA1", &$2);}
;
```

```
freemod : UI {$$=NodeY1 ("freemod", &$1);}
| SOI descpred gap {$$=NodeY3 ("freemod", &$3);}
| DIE {$$=NodeY1 ("freemod", &$1);}
| NO4 DIE {$$=NodeY2 ("freemod", &$2);}
| KIE utterance KIU {$$=NodeY3 ("freemod", &$3);}
| HUE statement gap {$$=NodeY3 ("freemod", &$3);}
| HUE terms gap {$$=NodeY3 ("freemod", &$3);}
| voc {$$=NodeY1 ("freemod", &$1);}
| JO {$$=NodeY1 ("freemod", &$1);}
;

neghead : NO4 gap {$$=NodeY2 ("neghead", &$2);}
;

uttC : uttA1 {$$=NodeY1 ("uttC", &$1);}
| neghead uttC {$$=NodeY2 ("uttC", &$2);}
;

uttD : uttC {$$=NodeY1 ("uttD", &$1);}
| uttC ICI uttD {$$=NodeY3 ("uttD", &$3);}
;

uttE : uttD {$$=NodeY1 ("uttE", &$1);}
| uttE ICA uttD {$$=NodeY3 ("uttE", &$3);}
;

uttF : uttE {$$=NodeY1 ("uttF", &$1);}
| uttF I uttE {$$=NodeY3 ("uttF", &$3);}
;

utterance : I {$$=NodeY1 ("utterance", &$1);}
| freemod {$$=NodeY1 ("utterance", &$1);}
| uttF {$$=NodeY1 ("utterance", &$1);}
| I uttF {$$=NodeY2 ("utterance", &$2);}
| ICA uttF {$$=NodeY2 ("utterance", &$2);}
| uttE IGE utterance {$$=NodeY3 ("utterance", &$3);}
;

%%
```

13 Appendix: old version notes for this document

The text of the reference grammar needs to be reconciled with the new parser on a few points, also listed as cleanup points.

The reference grammar itself will continue to be edited for readability!

9/10/2016 More work on commenting the latest PEG grammar. I need to modify the main text of the reference grammar, since the version with SOV marking and case tags only at the top level of arguments is now the main provisional parser.

9/8/2016 More work on the new round of comments on the latest PEG grammar.

9/4/2016 I have incorporated the text of the test parser PEG grammar as another appendix and I am rewriting comments on it afresh.

The test parser contains upgrades to SOV sentence marking and forbids logical combination of tagged arguments. It combines my two proposals for SOV marking; comments will be updated. It also incorporates a massive retooling of the phonetics of pauses, which does not reflect any change in the English articulation of the grammar, but did correct some unnoticed errors in the Visit parse. The basic idea is that the pauses before logical connectives are no longer incorporated into the previous word, but are treated as freemods, and any **comma** is readable as a freemod which is not followed by a name word (the last detail is required to handle **la blanu, Djan** phrases correctly).

The parse errors which were missed in the Visit were occasions where an APA or IPA connective was followed by a VV attitudinal: the connective must be closed with FI because the pause is phonetically mandated. A conceptual bug caused the more complicated earlier phonology functions to miss this.

8/30/2016 perfected the second test parser so that it can recognize afterthought connected arguments which are fully explicitly case tagged. Also corrected an error in the rule **arg7**. I note for the reader of parses that the second test parser renames **oneargument** to **subject**, because that is what

this class is, in both of its uses.

8/29/2016 specs for a second test parser are included in the PEG appendix. This one also restricts the formation of SOV sentences but in a different way: where more than one argument appears before a predicate in a statement, no more than one of them can fail to be explicitly case tagged. This version should parse the Visit as intended, detecting the parse errors which we want to detect, without any need for innovations in sentence construction.

8/27/2016 I have just posted a test version of the parser which draws a formal distinction between SVO and SOV(O) sentences not drawn up to this point. In NB3, JCB remarks that the terms before the predicate in the SVO case of the class `statement` rule should contain exactly one argument, but that the parser does not enforce this, and subsequently forms with more than one argument before the predicate have been cited as useful to implement SOV word order. I have already required that the initial terms contain at least one argument (if not, the sentence is captured as an imperative in rule `sen1`). I now propose changing the form of this case to

`oneargument (GIO terms)? predicate.`

The effect is to require that if there is more than one argument among the terms before the predicate, the particle **gio** must appear between the first two arguments. There may be latitude about where it occurs due to modifiers.

I have parsed the Visit using this test grammar (this is the parse actually posted at the moment). In the entire text there is just one sentence to which I had to add a **gio**. In all other cases, various formation errors in sentences had created incorrect sentences with unintended parses in the old SOV form. I had already noticed this phenomenon in some cases while working through the Visit. The problem is that a formation error which breaks a subject into two arguments or which causes an argument which has become detached from previous text to be left in front of the current sentence being parsed will lead to an unintended parse rather than an error. Several of the mistakes newly corrected in the Visit look to me like things which might happen quite often in text composed by English speakers and ought to be captured by the parser.

I do not think it is an undue hardship to say **da gio de di donsu** rather than **da de di donsu**, particularly since this SOV speech pattern does not seem to be common.

It is worth repeating that this test parser, though it appears to add an innovation to the language, has as its main effect detection of actual errors in existing text which does not use this feature at all. Almost all occurrences of the old SOV form are the result of mistakes in Loglan sentence construction, and the main usefulness of the new rule is found in its forbidding the old SOV form, not in its permitting the new one.

I am not going to make this my main version until I have thought about it a little and documented it in the reference grammar.

The reference grammar and the PEG appendix now contain descriptions of the new proposal in the test parser.

8/24/2016 Proofreading.

8/23/2016 Adding new Fall 2016 blurb.

7/30/2016: minor edits and proofreading. Moved the now huge swath of old version notes to an appendix.

7/23/2016 style change highlighting the list of social lubricant words and grammar correction of `junction` to `junction2` in classes A0 and TAI0, fixing a failure to require pauses between stress-final `cmapua` and predicates.

After that, substantial editing of most of the document.

Two significant grammar upgrades – not that they probably affect existing text. The case `terms keksent` in `uttA1` is changed to `modifiers keksent`: I am morally certain that this is the intention.

The class `uttD` is redefined as `uttD<-((sentence? period !ICI !ICA)/(uttC (ICI freemod? uttD)*))`, on which I will comment below. The idea is that utterances of class `uttE` which parse as of class `sentence` really should be recognized as sentences, which they are not under previous versions of my grammar or under `trial.85`.

7/16 a minor correction and updated running headers

7/9/2016: Numerous small edits and removals of outdated remarks.

I have introduced English terminology for frequently mentioned grammar classes to make the grammar section more readable. In some parts of the grammar, the use of the PEG class names seems reasonable, as these classes are only ever mentioned in the context of their being used to build the next class up. I also moved the set and list constructions to the right place in the grammar.

I have added the new English grammatical terminology to the PEG grammar comments. I believe that the classes which have been given English names are probably those (or most of those) which should be assigned Loglan predicates.

6/26/2016: I tracked down and added the definitions of the missing PA words, which are items from non-adopted subjunctive proposals that we may or may not really want. There are articles on the Loglan site which give fuller explanations of these words.

3/18/2016: a very minor typographical revision.

2/27/2016: more cleanup re the same point.

2/26/2016 The new implementation of the possessive is now described in the text of the reference grammar. Every now and then something in **le some lengu** is just a good idea, as in this case.

2/21/2016 An experiment. The `descriptn` class is upgraded in a way which makes it possible to eliminate the inflections in the LE word class (so **lemi**, **levi**, **lemina** no longer have to be words). It is rather delicate, though it has some possible advantages. This is updated in the PEG grammar appendix: new language in the reference grammar sections pending.

In parses, you will notice that former words **lemi**, **levi** come apart into separate words. Existing parses should look normal apart from this.

The idea is that **lemina hasfa** can now really be parsed as **le mi na hasfa**. In any of the LE initial clauses of `descriptn`, LE may optionally be followed by an `arg1a` then a PA2 tense, then by the expected rest of the construction. An interesting point is that the possessive `arg2` can be closed more efficiently by using a tense. A weak point which had to be corrected is that the `arg1a`'s used must exclude anything that starts with a `mex` (because `mex` plays an essential role in class `descriptn`; this was very evident when I tried to use more general classes of argument in this construction), and an obscure case of `arg1` does start with a quantifier, namely, certain cases of `abstractn`. This was detected by parsing the Visit: an actual parse failure due to this rare case showed up. In **Le ri po zbuma ga bilti**, the parser will not consider **ri po zbuma** as a possessive component, but will correctly parse it as **Le ri (po zbuma) ga bilti** (the few explosions are beautiful). But one can say **Le ge ri po zbuma ga pu bilti** (the few explosions' qualities of beauty), by guarding **ri po zbuma** with **ge**.

2/19/2016 A subtle change in articulation of PA "words": pauses next to CA0 are always permitted, even internally to APA words and to PA used prepositionally (pauses between PA units are not allowed in these contexts). Where PA "words" are used as tenses or as modifiers without an argument, a pause between PA units does not break them (unless one looks forward and sees that one is about to read a preposition: **Mi hijra pa ce na fa vi la Djan** articulates as **Mi hijra (pa ce na fa) (vi la Djan)**; the break

between **fa** and **vi** here must be an actual pause, as play with phonetic parses will reveal). This does not mean that the semantics are indifferent to breaks: a hard break of a PA word used as a modifier with no argument, using **gu** instead of a pause, may change the semantics (this can clearly happen with location words).

2/18/2016 A vanishingly small improvement to the treatment of APA words, making it possible to link PA words without arguments used as modifiers with A connectives without confusing them with legacy APA words: **Mi hijra pa, e na, e fa** parses, because the parser will not complain about **e na**, as it would otherwise (fearing ambiguity with **ena**,), because it knows that an APA word will not appear before a further A connective (or other logical or sentence connective).

2/17/2016: slight tweak defending legacy APA words from ambiguities. Also, changed `mod1` so that the PA component contains no pauses: **Mi hijra na vi la Djan** parses **Mi hijra na (vi la Djan)** rather than **Mi hijra (na vi la Djan)**. The latter approach **could** be interpreted to give the right semantics, but the new one is clearer as to the likely intended meaning.

2/16/2016 I think the complete solution to APA words (requiring corrections to the Visit parse, alas). Any APA word (or CAPA, ICAPA, IPA) must be closed with either **fi** (preferred) or a full comma pause (deprecated, preserved to make it possible to parse legacy text with a little punctuation: though we think IKOU[`pause`] words might survive, as a pause after such words seems perfectly reasonable). The Visit parse has been corrected, which was time consuming but not awful.

Also added fine tuning for vowel initial names: there is always a pause between them and a name marker word, so the proper behavior is enforced (non-name readings are tried first) – unless CANCELPAUSE is inserted. Some vowel initial names which resolve into other things can only be uttered with CANCELPAUSE. I **thought** there might still be a use for CANCELPAUSE!

2/15/2016 Tiny exception to the previous: the negative attitudinals NO UI can be pronounced without internal pause. This does not extend to negating a VV compound; just a single VV word may be negated pauselessly. This is widely attested in existing text, and even if NO UI is written with a space, NO, UI with a comma will work quite differently; when NO UI is written one actually **presumes** that no comma is present.

2/14/2016 The phonetics of boundaries between vowel-final and vowel-initial words are so annoying that I finessed the entire issue by (invisibly

to ordinary orthography) imposing pauses (which do not need to be comma marked, but do need to be at least a space) before VV attitudinals (allowing compound VV attitudinals; one needs to pause before the first in a sequence of VV's not before each of them).

2/13/2016 Enough work done on the grammar that I set up the appendix with the fresh PEG code. I added alternative forms **(po/pu/zo)z(a/i/u)** and closures **guo(zi/za/zu)** to my proposal re the GUO GUO problem due to objections about words like **poia** being misconstrued as **po ia**.

2/11/2016 A solution to unordered lists (finite sets given by enumerating their elements) and ordered lists is given. It is different from the old one but uses the original phonetic material. I'm going to defer discussing it, but enough is said in the grammar that the enterprising can figure it out.

2/11/2016 There is now only one List of Issues.

2/11/2016 Finally got the syllabification of borrowings perfectly tuned so that a legal borrowing cannot be created by placing an explicit syllable break in a pre-complex in a way which violates djifoa boundaries, and the default syllabification of borrowings actually satisfies the test conditions if made explicit. Details in classes **SyllableA**, **SyllableB** and **JunctureFix**, and there is a detailed essay on moving syllable breaks in borrowings (see table of contents) which motivates the details, in detail. I changed the syllabification of names back to the original (end a syllable as early as possible) since that is also the strategy for borrowings except in one crucial case.

2/10/2016 Got a correct understanding of the VCCV rule, and realized that I need to ban the five letter forms as borrowing affixes for similar reasons (and besides, they certainly are not borrowings, and have their own affix forms!)

2/10/2016 A Useful Remark: it may **look**, because of my flurry of remarks since the New Year, as if I have been making lots of changes to my provisional grammar. This is not really the case: most of the issues mentioned are internal bug fixes which would not require any consultation with la Keugru even if the grammar as it stands were official. The grammar is quite stable, where parsing of ordinary utterances written in ordinary Loglan style is concerned, and the fact that the Visit parses well suggests that I have not moved far from the original situation. The Proposal regarding additional long scope abstraction operators is serious and new. The 2/9/2016 bug fix and following readjustment of syllabification is major but is a bug fix (the stated policy in the text should have led to the new behavior all along). It should only affect text with explicit syllable and stress marking re parsabil-

ity, though the actual parse produced for a word may be different [and there are few or no users who know how to see the word parses]. The 1/29/2016 adjustment requiring additional explicit pauses before names enforces something that one must do in speech anyway. Most of this activity is debugging to get phonetic parsing to work correctly, now that I am testing it extensively (it started with debugging of quite standard parsing where needed in the Leith text).

The other activity which I have been doing extensively though I have not reported on it as often is reading and editing of the text of the reference grammar. I have been trying to remove anachronisms (which can lead to inconsistencies between different parts of the text) and improve readability.

2/9/2016 technical fix to syllable formation. The default articulation of a borrowing into syllables is now allowed as an explicit articulation by `JunctureFix`. I made the same change in default placement of syllable breaks in the `Syllable2` class which appears in names, though there is no urgent need for that other than maintenance of some parallelism. Basically, it will go a little further into a final stream of consonants than it did before. Also subtle fixes to final CCV and CVV djifoa to assist recognizing ends of complexes with explicit stress. Predicates cannot be immediately followed by `y` for subtle reasons having to do with borrowing affixes.

2/9/2016 major bug fix. The class `JunctureFix` was broken because of a typo, so I didn't notice that `HasCCPair` was defined in a way which contradicted its assumptions about syllable breaks in borrowings, which were also wrong in other ways. You live and learn. Corrected (?) versions up. Required some further fiddling with other classes.

2/9/2016 I have been editing the text of the reference grammar diligently. I note the addition of essays on borrowing predicates and making complexes. In recent days I have also added an essay on the solution of the false name marker problem, an essay on the odd auxiliary rules for placement of explicit stresses in borrowings, and (in the PEG section) an essay on the rather complicated looking PEG rule for recognizing apparent initial affixes in a borrowing as being nothing of the sort.

2/8/2016 corrected a minor bug in `CCVCVMedial` and partially enforced penultimate stress on numerical predicates.

2/7/2016 change to notation for stresses and syllable breaks in the PEG. Corrected some subclass definitions of djifoa which might malfunction in phonetic parses. This fix involves enough change of text that it may have new bugs, but it seems to work correctly on existing bits of phonetic parsing.

2/7/2016 Fixed **ne**, **tori** problem. Quantifier predicates will not contain spaces or comma pauses and will not be consumed by preceding quantifiers. Solved trailing spaces issue by a change to the **end** class. Technical change to **JunctureFix**. Forbade silliness with syllabic consonants which I thought was already forbidden.

2/6/2016 Implemented JCB's silence/change of voice marker **#**. This is a rather subtle move. Note that **#** is not a quotable or parenthesizable piece of Loglan punctuation (the class **utterance0** of utterances which can be quoted or parenthesized is not affected by this move). The use of it is to mark changes of voice in batch processed texts without having to introduce a line break.

2/5/2016 Definitely I need to do phonetic parser testing. I found not one but two bugs which cooperated to break a phonetic parse which I took from an example in L1.

2/5/2016 Finished a major editing pass through the document. I tried to remove anachronisms, notably any references to pause/GU equivalence except as a thing of the past.

2/5/2016 fixed a fine phonetic point only in play because we are now using some Cvv-V cmapua: when a CV cmapua is followed by a VV word in a way which would make a monosyllable, at least a syllable break must be indicated.

2/3/2016: fix to class StressedSyllable2 – it should **not** assume a final consonant! A tiny fix: corrected the parser to accept VCV letterals in acronyms.

2/3/2016: added an essay (look in the table of contents) detailing the reasons why I believe I have *solved* the problem of recognizing the left boundary of a name. Added official Proposals re new letter names and new abstraction operators and closures. Documented the new abstraction operators and closures in the reference grammar. The changes to vocatives and inverse vocatives should be stated in more detail.

2/3/2016 minor fix to CCVV. Don't forget that 2/1 changes, which are important, are only documented in the PEG appendix so far.

2/1/2016 major upgrade and Proposal re closures of abstract descriptions and predicates.

1/31/2016 rationalized shortslope abstractors

1/31/2016 updates to grammar of vocatives and inverse vocatives. The PEG appendix has the rules entirely replaced with fresh versions but the comments are old and some may be out of date. Not all updates are nec-

essarily in the reference grammar yet. I edited the comments in the PEG appendix.

1/29/2016: what I think is the final or near-final fix to the name marker problem. One *must* pause somewhere before the end of the first name after a name marker which does not actually stand before a name word. It's a very subtle piece of PEG programming!

1/27/2016: new series of foreign letters Caiu, Ceiu, supporting names for QWX. Refinement of the false name marker solution: a name marker word followed by an explicit pause will be followed by a name as a last resort (after trying other alternatives).

1/21/2016 multipart foreign name vocatives and inverse vocatives; LI LU quotes of names.

1/19/2016 overhaul of capitalization. CCV djifoa can be quoted with **liu**.

1/18/2016 forbade CCCVV predicates for technical reasons

1/17/2016 Further revisions in support of Leith parsing.

1/10/2016 Happy New Year! More modifications, mostly fixes and extensions suggested by parsing Leith novel.

12/20/2015 A number of changes to support parses of Leith novel.

11/15/2015 Added comments to the text supporting the grammar changes dated 11/14/2015. All of these in one way or another I regard as open to question, so the comments are labelled with the date (and the questions are raised there).

11/14/2015. Corrected to deal with a number of minor changes to the grammar made in the course of parsing the First Visit to Loglandia.

10/22/2015. Corrected to deal with rationalization of PA and NI classes and elimination of pause/GU equivalence.