Fall 2015 Report on the State of the Loglan Language (official and provisional)

M. Randall Holmes

February 9, 2016

version notes:

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 shortscope 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 necessarily 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.

1 Introduction

This document is intended to be a comprehensive report on the state of the Loglan language as of Fall 2015, for the Board, the Academy and the membership. I do attempt to distinguish between things officially approved and things which are still provisional, whether proposed by me or by others. 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

2 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.

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 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 my own 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 (Phrase 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 implementation; 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, although our specification can be supported at every step by remarks in NB3, and words from the dictionary do parse sensibly.

Having defined the syllable, we made the further decision to require that names be resolvable into syllables as well. 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 one borrowed predicate.

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.

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 notions of syllable and deduce stress patterns of predicates, 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 need to make this work. It was necessary to ensure that a comma marked pause was legal whereever 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 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.

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 pauses. 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).

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.

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.

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; 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 words) 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 allow closure of logical connectives of the classes A, CA, I, ICA with GU (whether or not they have PA suffixes): then an A PA situation which is not intended to be read as APA can be fixed either by writing A, PA (which is deprecated; breaking a word with an explicit pause is dangerous) or AGU PA (situations like APAGU PA can arise). Earlier versions of my solutions involved inconvenient pauses, which are no longer required.

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 explansions 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 indispensible (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 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. For example articles like **lemi** are actually words – putting a comma pause in will break the word and generally make the utterance ungrammatical. But in all previous versions one could write **le mi**. This is no longer the case.

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, unless they are phonetic pauses before a following vowel-initial or A word. 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 letteral 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, and also must end with explicit pauses. Acronyms are defused as a problem.

I have proposed to eliminate the vowel letterals of the form **afi**, **ama** in favor of **zia**, **ziama**. The old forms are still supported. I favor eliminating them because they are phonetically very eccentric.

strong quotation: There is a completely different strong quotation proposal in the reference grammar. The official one is not PEG or even BNF parsable. I became aware after the fact that my solution to strong quotation with **lie** in its usual applications was isomorphic to the solution for foreign names with **lao** provided by the previous Academy.

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 minds 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**. 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). 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 predunit 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.

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 acknowedged 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 donsu 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.

I have recently placed restrictions on freemods in and preceding termsets which should make it possible to use pauses to introduce shared termsets in some simple situations.

recognizing imperative sentences: It is an error in the existing grammars that a sentence like **Na la Ven, donsu 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 donsu 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 a raindrop now!".

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

3 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.

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 to be modified in accordance with any decision of *la Keugru*. 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).

4 Differences from earlier implementations of Loglan

I list some points about the approach I have taken to implementing the phonology/orthography, lexicology and grammar of Loglan. This section is intended only for readers with prior technical knowledge of the language.

NOTE: this section of the reference grammar as the same function as the catalogue of issues above in the report. I will probably eventually eliminate this section, but I might make points here not made above.

general intentions: My intentions have been conservative. My intention is that the language here be intelligible to a speaker of 1989 Loglan if there were such a being. I have made changes and clarifications mostly where I thought some clarification or repair was needed. There is no intention of radical redesign of the language, nor any intention of addition of major improvements at this time.

the perceived difficulty with LIP and trial.85: The official definition of the language was embodied in the trial.85 formal language, the LIP interactive parser. and the document Notebook 3 (NB3) as superseded by 1989 Loglan 1 and further decisions of the academy. There are serious difficulties with this. There is a fairly good description of the phonetics in the sources, but no adequate formal description of the lexicography. The definitions of classes of structure words are implicit in non-human-readable tables in LIP, which evidently have bugs in them. The fact that the lexicography (and the phonetics) are not actually integrated with the formal grammar means that there are limits to any provable unambiguity (and indeed the old definition has demonstrable ambiguities caused by interaction of word classes with the grammar). There are specific bad features of the old definition which I will bring up under other points.

PEG rather than BNF: I chose to write a phrase expression grammar (PEG) rather than a BNF grammar. Issues of ambiguity do not arise with a PEG, as it assigns priorities to alternative parses and picks the first one that works. The hazard analogous to ambiguity in a PEG implementation is "unintended parse", when priorities are set incorrectly. The PEG formalism is very expressive but does act a bit weirdly.

a unified approach to all levels: I wrote a single parser working from the level of letters upward, handling phonology/orthography, lexicography, and grammar all in the same parser, with no preprocessing. I thought initially that I would eventually write a different parser for Loglan phonetics, but in fact the initial PEG parser developed into a parser capable of handling a phonetic notation as well as traditional Loglan orthography.

phonetics: 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 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.

phonetic parsing: The development of parsing of predicates required rules that analyzed implicit stress on syllables (which in turn required me to define syllables). It turned out to be fairly easy to adjoin explicit notation for syllable breaks (using the hyphen instead of the close comma) and for stress and emphatic stress on syllables. I then made an effort to ensure that wherever an actual pause is permitted, a comma pause is legal to write (this was not true in LIP). This makes it possible to write in a "phonetic transcript" style in which all stresses of interest are shown (or all stresses and syllable breaks if one wants to be thorough) and all pauses in speech are represented by actual commas, which is handled by the parser as readily as traditional Loglan orthography (and mixed styles are also possible – phonetic notation of stress can be used for emphasis in writing).

A converse feature of the current parser (defending the concept of "word") is that one cannot write a space internal to a production of a grammatical class of words (this rules out **le mi hasfa**, which LIP does understand, because **lemi** is a word of class LE and that is the only way for the **mi** to make sense in that position).

lexicography: Certain large classes of words (the A connectives and relatives, the PA and NI words, the LE words) are very large and do not have adequate formal definitions in the previous sources. I made formal definitions of these word classes, which in many cases do not coincide with the definitions implicit in LIP (the latter are sometimes buggy). In almost all cases, there is no practical effect on what can be said in Loglan. One bad case is the separate immediately following point. The logically composite PA words as defined in NB3 involve an actual ambiguity (caused by permitted use of initial **noi**) which I repaired.

APA words and kin: Words like *efa* (and more crucially *irau*) present problems of actual ambiguity for the old definition of the language. We resolve this by allowing closure of logical connectives of the classes A, CA, I, ICA with GU. The APA words themselves might be dispensible, but they are common in the NB3 corpus (and the latest solution is much less awkward than earlier versions). The IPA words are not really dispensible in our view.

I entirely forbade KA and KI words to absorb following PA words as LIP seems to want to do.

We have arranged for words of classes PA and NI, which can be very long, to allow internal whitespace or explicit pauses under certain conditions.

acronyms: Acronyms were known to the previous Academy to be a problem. I have been guided in my handling of these by the principle that acronyms are a minor feature of the language and that it is therefore absurd to have to pause between letterals appearing as successive arguments to defend the acronym class. I changed acronyms to be names rather than predicates, which gives them the same boundary defenses as names (starting with name markers or pauses and ending with pauses). Acronyms appearing as dimension in NI words start with a new initial marker word and end with pauses. This leaves one free to use letterals freely for their most important grammatical function, as pronouns, without awkward pauses.

strong quotation: I have an entirely different strong quotation proposal. The original strong quotation is not implementable even in BNF.

serial names: Serial names like **la Djan Braon** were known to be an issue by the previous Academy. The strange solution had been adopted by the previous Academy of having the pause in **la Djan Braon** be of a different length than the other pauses in the language, to allow **la Djan Braon, blanu** to be read as a sentence and **la Djan Braon Blanu** to be read as the name "John Brown the Blue". My view is that serial names are a minor feature of the language which should not have leverage at such a fundamental level.

There is the further general issue of recognizing the beginnings of names. A name word begins with a pause (not necessarily shown as a comma) unless it is immediately preceded by a name marker word (**la** and **hoi** are examples). This causes names with phonetic copies of name markers in them to be problematic.

We have given a complete solution to both problems. Predicates in serial names must be preceded by **ci**. Name words containing name markers must be preceded in serial names by **ci** (this was already introduced by the previous Academy). Further, names following predicates must be preceded in serial names by **ci**. This allows correct resolution of serial names. All pauses are of the same length, though pauses at the ends of names followed by **ci** or further name words may be written simply as spaces. **La Djan Braon, blanu** is a sentence while **la Djan Braon ci Blanu** is a name.

The solution to the problem of recognizing the start of names is to allow names to begin only in very restricted contexts. The principal hazard in the old language definition was the presence of unmarked vocatives, which meant that a name could start almost anywhere, and so if names could include name markers then large sections of pause-free speech could accidentally be misparsed as names. The elimination of unmarked vocatives has been approved by the Academy. Names thus occur only after name markers (**ci** is a name marker), or after other name words (if they are free of false name markers), or after an explicit pause after a descriptive argument, as in **le blanu, Djan**, if it is name marker free. Complete details are in the grammar.

other issues: For other issues (some important) see the more complete catalogue of issues above in the Report.

5 Phonology and Orthography

5.1 Introduction

This section is about how Loglan is to be written (both letters and punctuation) and how it is to be spelled and pronounced.

5.2 Alphabet and Capitalization

The alphabet of Loglan has 23 letters **abcdefghijklmnoprstuvyz**, the Latin alphabet without the letters **qwx**. Removing **qwx** is the content of a proposal before the Academy. **aeiouy** are the *vowels*; **aeiou** are the *regular vowels*; the non-vowels are *consonants*; the consonants **mnlr** are *continuants* and may be used in a vocalic manner, in which situation they are referred to as *syllabic consonants*. The names of the consonants are formed by appending **-ai** or **-ei** to the letter (for uppercase or lowercase respectively). The names of the lower case regular vowels are formed by prepending **zi-** to the vowel; the names of the upper case regular vowels are formed by appending **-ma** to the name of the corresponding lower case form. The renaming of the vowels is a current proposal of mine. The question of what the name of **y** is is open in my mind: **ziy(ma)** is accepted by my parser at the moment, but is irregular in form (**liu ziy(ma)** parses only by special exception!) The language ought to have names for the foreign letters **qwx**: I have proposed **Haiu, heiu, Kaiu, keiu, Vaiu, veiu** as the names for **X, x, Q q, W, W.**

The capitalization convention of Loglan is that any uninterrupted sequences of letters (which may include syllable breaks and stress markers but not spaces or terminal punctuation) may have the initial letter either uppercase or lowercase, and must have all subsequent letters lowercase, with certain modifications (finalized 1/19/2016): the name of a letter may appear capitalized anywhere in a word, and lower case z may be followed by a capitalized vowel or a juncture may be followed by a capital letter anywhere in a word. The capitalization of letter names supports a convention with regard to possessives well-attested in Loglan text; the capitalization of letter names and of vowels after lower-case z supports internal capitalization in acronyms also well-attested in Loglan text; capitalization after junctures supports certain compound names like `la Beibi-Djein`.

5.3 A Note on Styles of Orthography

It was at one time a project to have a phonetic parser for Loglan. We would have thought of this as a separate gadget manipulating something like JCB's phonetic notation in NB3 or Loglan 1. In the event, this is not what happened. In the course of writing my parser, it became clear that a careful analysis of syllables and stresses was implicit in correctly parsing the standard Loglan orthography from the level of letters upward, and adding explicit devices for indicating phonetic features such as pauses (already of course denoted by commas, but not all pauses can be represented by commas under LIP), stress (not explicitly notated at all in the standard orthography) and syllable breaks (represented explicitly by "close-commas" in some contexts in JCB's notation; we have suppressed the close-comma and use hyphens for syllable breaks) seemed natural. We have produced a single parser which admits a continuum of styles ranging from the standard Loglan orthography to a style which I term "phonetic transcript" in which there are no spaces other than those which represent explicit pauses, and these are all marked with commas, and all stresses are shown explicitly (one could further show all syllable breaks explicitly, and for genuine phonetic transcript one should indicate explicitly how optional disyllables are being handled).

5.4 Punctuation

The comma **,** denotes a pause in speech. There are contexts in Loglan orthography where the presence of a mandatory pause can be deduced from the orthography though a comma is not present. It is a design goal in the parser, which I am not sure that I have achieved, to allow an explicit comma to be inserted in any place where a pause is allowed, and certainly whereever a pause is mandatory. A comma is always followed by a space. The use of a close comma to force a syllable break has been abandoned at least for now (if desired, it can be reintroduced as a variant of the syllable break **-** but I advise against it). Wherever a pause is intended, at least a space should be written. Spaces cannot occur in the middle of words in the latest version of the parser. One might think that one could pause whereever whitespace occurs, but there are some exceptions: for example, it is presumed that there is no pause after a name marker word unless an explicit comma is shown.

The period **.** denotes the termination of an utterance: other terminal punctuation marks **?!:;** are exactly equivalent as far as the parser is concerned. Terminal punctuation ought to be followed by at least two spaces. The parser now enforces at least one space or end of text after a terminal punctuation mark. I am considering some other punctuation, such as ellipses, dashes, and (silent!) parentheses and brackets. (ellipses and dashes have been added as freemods).

The hyphen **-** denotes a syllable break (it may **not** be pronounced as **y** as JCB proposed in Notebook 3). The apostrophe **'** marks a stressed syllable; it may be used instead of **-** (not in addition to the hyphen) to mark the syllable break after a stressed syllable, and it may be used after a final syllable before a non-letter or end of text to indicate that the final syllable is stressed.

The asterisk * may be used in two ways: in initial position, it marks utterances which are deprecated or error-infested as Loglan utterances [by its nature, this use is not recognized by the parser]. It may also be used with the same grammar as the apostrophe to denote emphatic stress.

It is important to notice that stress markers are placed not on the vowel but at the end of the stressed syllable.

Stress markers are always optional. There is no specific notation for a syllable which is not stressed, though it might be useful to add such a marker. (A note for those familiar with the language: of course, when we say that stress markers are optional, we should point out that indications of stress are not optional in some cases: the stress in a predicate word, if not signalled by an explicit stress marker, must be signalled by the end of the predicate word, indicated in this case by a space or terminal punctional mark. In the presence of an explicit stress, the end of the predicate word can of course be determined without additional punctuation.)

I suggest using the stress markers to indicate rhetorical stress in a way the parser can handle. I have been doing this myself.

Spaces between words not occurring after a comma can in most cases be omitted; in certain cases they indicate mandatory pauses which can always be explicitly marked (insert missing comma), or in the case of spaces after predicates may serve to mark penultimate stresses in the preceding word, and can be eliminated if the stress on the penultimate syllable of the preceding word is explicitly marked. Places where spaces are written are often but not always places where pauses are possible in reading the given text, but one certainly should not presume that a space indicates a pause. The aim is that any space where the pause is optional can be omitted and any space where it is mandatory can have a comma added. Spaces are now forbidden in the middle of words (with certain exceptions), and it is often but not always true that a place where a space is written is a place where one can pause. Whitespace or explicit comma pauses are permitted in the interior of PA words and NI words under certain conditions, and also after a borrowing affix in a complex predicate. Any place where whitespace *must* be written should be a place where an explicit pause can be inserted.

A specific style which should be possible to write for any Loglan utterance is the *phonetic transcript*, in which spaces appear only after commas and all stresses and syllable breaks are explicitly marked (or at least all stresses and syllable breaks of interest are marked; but if spaces are omitted stresses on predicates become mandatory).

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.

5.5 Pronunciation

Input about actual pronunciation of Loglan sounds from other members of the community is actively solicited.

5.5.1 Regular Vowels

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

a is pronounced as in *f***a***ther*
e is pronounced as in *b***e***t*
i is pronounced as in *mach***i***ne*
o is pronounced as in *l***o***st*
u is pronounced as **oo** in *p***oo***r*

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).

5.5.2 Grouping of Vowels

Two-letter diphthongs pronounced monosyllabically are as follows:

ai is English long **i** as in *p***i***ne*
ei is English long **a** as in *p***a***ne*
oi is as in English *b***oi***l*
ao is as **ow** in English *c***ow** (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.

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. [There *were* such contexts in the past, as I have only recently realized: 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].

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 mandatory pauses.

There are two different rules for grouping long strings of vowels. In a structure word (which will always be a compound attitudinal with an even number of vowels), the vowels are grouped in pairs and each pair is monosyllabic or disyllabic in a way compatible with the rules above. An odd length stream of vowels made of structure words will consist of a one-letter logical conjunction followed by a compound attitudinal.

In a name or predicate word, a long string of vowels without an explicitly given pause or syllable break is grouped using a priority scheme. If the first two letters of the stream make up a mandatory monosyllable, they are grouped together; otherwise, if the second and third letters make up a mandatory monosyllable, the first vowel is pronounced by itself and the second and third are grouped together; otherwise, if the first two letters make up an optional disyllable they may be grouped together or pronounced separately (the parser, absent an explicit syllable break, prefers to group them together), and otherwise the first letter is pronounced by itself; in any of these cases, repeat the process with the remaining stream of vowels until it is exhausted. This is a new proposal of mine superseding rather different rules given in earlier texts.

5.5.3 The Irregular Vowel

The irregular vowel **y** has as its standard pronunciation the "schwa" sound found in English *sof***a**. John Cowan suggests that we might sometimes want to give it the value of **oo** in English *l***oo***k*, a suggestion which I rather like. It is very important to note that the English or Russian tendency to convert the vowel in an unstressed syllable to this sound must be resisted.

5.5.4 The Consonant Sounds

The consonants **c** and **j** have pronunciations which are unusual in English.

c is pronounced as **sh** in **sh***oe*.

j is pronounced as **s** in *trea***s***ure*.

The consonant **g** always has the hard pronunciation in **g**et.

The consonant **h** usually has its usual English pronunciation but may also be pronounced as **ch** in Scottish *lo***ch** (the former pronunciation of **x** in Loglan), as for example if it appears at the end of a syllable. (This alternative pronunciation is part of the proposal to eliminate the foreign consonants).

The consonant **n** has its usual pronunciation in English except before **g** or **k**, where it is pronounced as **ng** in *so***ng** (which is also quite usual in English!). The latter sound appears only as a pronunciation of **n** in such contexts. I note that this might also apply to **nh** if **h** has the alternative pronunciation.

The other consonants all have their principal pronunciations as in English.

It is worth noting that **tc** is English **ch** as in **ch**in and **dj** is English **j** as in **j**udge.

5.5.5 Syllabic ("Vocalic") Consonants

No consonant ever appears doubled in Loglan, except the continuants **mnlr**. Where the continuants appear doubled, they are used syllabically (in effect, as vowels). We call these doubled consonants *syllabic pairs*. A syllabic pair is never adjacent to another occurrence of the same consonant. We require that a syllabic consonant (a continuant used as a vowel) must always be doubled: the main effect of this rule (which is suggested in Loglan 1) is that the spellings of some names must be changed.

5.5.6 Grouping of Consonants

Consonant clustering is governed by a number of rules.

There is a list of allowed initial pairs of consonants which may begin a syllable. An initial pair will not overlap with a syllabic consonant pair. A Loglan syllable will begin with a vowel or syllabic pair, or with a single consonant (not part of a syllabic pair), or with an initial pair (not overlapping a syllabic pair), or with an initial triple of consonants in which each of the two adjacent pairs of consonants is an initial pair.

The 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**

There is a list of pairs of consonants which may not occur adjacent to one another, even across a syllable boundary. These are called impermissble medial pairs (and the other pairs are called permissible medial pairs).

The 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 impermissble 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.

5.5.7 The Loglan Syllable

It is a curious feature of Loglan as presented heretofore that there is no precise definition of a permissible syllable, and in fact the places at which syllable junctures occur in JCB's phonetic productions are sometimes quite odd. We are much more precise about this, though we believe that we have not thereby in principle much modified the set of allowed words (or in practice modified it at all).

A Loglan syllable consists of three components, only one of which is mandatory. It begins with an optional initial consonant group, continues with a mandatory vowel group, and terminates with a final consonant group.

The initial consonant group is either null, a single consonant, an initial pair or an initial triple in which both adjacent pairs are permitted initial pairs. The initial consonant group cannot be or overlap a syllabic pair. This treatment of initial triples is stated in Notebook 3.

The vowel group is either a single vowel (**y** can occur), a mandatory or optional monosyllabic pair of regular vowels, or a syllabic pair of continuants. A syllabic pair of continuants cannot be adjacent to another occurrence of the same consonant.

The final consonant group is null or contains one or two consonants, neither of which can stand at the beginning of an impermissible medial pair or triple of consonants (possibly looking ahead into the next syllable). Further, if there are two consonants in the final group, they cannot consist of a non-continuant followed by a continuant (this is a new condition: such a combination would be forced to be pronounced as another syllable). The final consonant group cannot be or overlap a syllabic pair.

A syllable break immediately preceding a vowel must also follow a vowel (any syllable following a consonant group picks up at least one of those consonants).

The position of the juncture in a group of three or more consonants at the end of a syllable may be optional; in no case is it phonemic (there cannot be two distinct words which differ only in the placement of such a juncture). Forcing a syllable break between vowels may change one word to another (this will only happen in proper names).

Of course, further conditions are imposed on syllables depending on the kind of word in which they appear.

It is interesting to note the maximum degree of consonant clustering: CC-CCC is possible in a name or borrowing.

6 Phonetic Classification of Words

There are three main classes of words in Loglan, structure words, names and predicates. This section is concerned with the phonetic shape of these words.

6.1 Structure Words

Structure words (in Loglan, `cmapua`), also sometimes called little words, are mostly grammatical particles. There are some words which are phonetically structure words and semantically predicates, and some which are semantically names (subject to our proposal that acronyms be names rather than predicates).

On the phonetic level, structure words are built out of phonetic units of one of the shapes V, VV, CV, CVV, Cvv-V, where V denotes a regular vowel, C denotes a consonant and vv denotes a monosyllable (optional or mandatory). The units of the shape Cvv-V are currently not used or very little used [they occur in acronyms using the old vowel letters with the shape **afi**, and in a way inconsistent with the actual articulation of the acronym into letters, and may also occur in predicates formed with **zao** under a current proposa; recent proposals do add series of such words as names for foreign consonants and alternative abstraction constructors and closures]. The commonest cmapua consist of a single unit.

If any unit in a structure word is a VV, all units are VV's. These words are the compound attitudinals already mentioned. The compound attitudinals are the only vowel-initial words which do not have to be preceded by a pause.

A unit of the shape V may only appear in initial position, and must be preceded by an explicit (that is, comma-marked) pause. A V by itself falls under this rule. These words are logical or utterance connectives of certain classes, and in fact all logical or utterance connectives of these classes must be preceded by explicit pauses, though some of them have no phonetic reason to be so marked. This will be spelled out in more detail later.

Stress in a structure word is completely free – any, all or none of the syllables may be stressed. If the final syllable of a structure word is stressed and the following word is a predicate, there must be an explicit pause, comma-marked, between the two words. The phonetic cmapua which are semantically predicates are supposed to be penultimately stressed, like phonetic predicate words, but this is not currently enforced by my parser (it will be enforced when I get around to it!)

It is worth noting that the actual class of phonetic structure words has hardly any use in the grammar. Grammatically, this class is broken up into individual often very small classes each of which happens to satisfy its constraints. There is only one context in the PEG grammar, following the quotation article **liu**, where the general phonetic class of structure words is used. I believe that in LIP it may never have been used at all, since after **liu** LIP accepts only actual structure words, not phonetically acceptable ones; the latter approach is the one we take.

It is also worth noting that the phonetic units of structure words are not individually necessarily single syllables: a VV or CVV might be a disyllable, and a Cvv-V unit is definitely not a single syllable.

The vowel in a CV cmapua syllable cannot be followed directly by another vowel which would make a monosyllable. This defends the Cvv-V cmapua units. When a CV word is followed by a VV word without a pause, a syllable break must be used to indicate this. The pronunciation of the Cvv-V and the CV-VV alternatives is different: the change in orthography is also phonetically significant.

6.2 Names

A name must be made up of valid Loglan syllables and must end with a consonant. The final consonant must be followed by either a comma-marked pause [which is included in the name by the parser, unlike the following options], end of text, a terminal punctuation mark, or a space followed by another name word or the name marker word `ci` (in which case there is a pause in the pronunciation). [It can also be followed by a stress marker followed by any of these things, if it is finally stressed]. I disagree with Steve Rice's opinion in Loglan 3 that the comma after names can be omitted with experience; I believe it is important to reinforce it, and my parser requires it absolutely [and I have introduced more mandatory comma marking of pauses *before* names when necessary to firmly indicate where a name begins].

The requirement that names are formed of syllables is a new proposal. The original Loglan rule allowed any string of sounds ending in a consonant. In practice, nothing here has required changes in any Loglan name in use other than enforcing the rule that syllabic consonants must be doubled (which was actually suggested as an alternative by JCB in L1). Requiring that names be parsable into syllables has the virtue that false name markers can be restricted to occurrences of the name marker words inside a name such that the rest of the name is itself a phonetically valid name.

An acronymic name is not consonant final but must also have the following pause if not final, which must be explicitly comma-marked unless terminal punctuation, another name or **ci** follows.

We do not require that names be capitalized, but it is usual to capitalize them.

A name must be preceded either by a pause or by one of a class of name marker words. The name marker may optionally be followed by a pause. The class of name marker words is (currently) **la hoi hue ci gao liu** (recently enhanced to include **loi**, **loa**, **sia**, **sie**, **siu**). In orthography, the pauses mentioned here may or may not be comma-marked. Note that it is possible for a name to occur not preceded by a space, but only if it is preceded immediately by a name marker.

A vowel initial name must be preceded by a pause, which must be shown by at least a space and may or may not be comma-marked.

Names are the only consonant final words in Loglan. Thus the end of a name is always readily recognized.

A phonetic copy of a name marker word occurring in a name, such that the part of the name following the phonetic copy is a well-formed name, is called a *false name marker*. A false name marker presents a difficulty for the reader or auditor trying to determine the beginning of a name. The rule is that a name begins as early as possible. To make a false name marker into a true one, follow it with a space or comma marked pause. A further important rule is that a name containing a false name marker cannot occur after another name without being marked (we will see that the marker used is **ci**).

1/27/2016 when a name marker word is not followed by an explicit pause, and the text following parses as a name, it will be read as such. If the name marker word is followed by an explicit pause, the text following will be read as a name only after other alternatives are tried. This gives a much subtler solution to the false name marker issue. Examples are needed.

1/29/2016 I believe this problem is completely fixed (mod bugs: the PEG rule used is extremely tricky). If a name marker word is used and is not intended to be followed by a name word, a comma-marked pause or a space before a vowel (which phonetically must be a pause) must occur before the next break after a consonant. This can be handled by pausing right after the name marker word; it can also be handled by pausing later.

It is interesting to observe that the reforms in the last two paragraphs have the effect that we presume that a space after a name marker word is not a pause unless it is explicitly marked as such. There are examples where the parse changes if a comma is inserted.

6.2.1 Essay on why we believe we have solved the name boundary problem

From the beginning, it was easy to recognize the right boundary of a name, because a name is the only regular kind of word which ends in a consonant, and it is always followed by a pause (unless by end of speech or text). It is convenient to allow this pause not to be expressed by a comma in case it is followed by terminal punctuation or by another name in a serial name or by the marker word **ci** in a serial name.

The thing that was left in a much less satisfactory state by our Founders was the problem of identifying the left boundary of a name. A name must begin with a pause, unless it is preceded by one of the name marker words (these include **la**, but also **ci** because of its use in serial names and name-final descriptions, and **hoi** because of the vocative use of names (and so now **loi**, **loa**, **sia**, **sie**, **siu** because these are now allowed as vocative markers) and **hue** because of the use of names as inverse vocatives. The single word quotation operator **liu** and the letter formation operator **gao** are also name markers, because they can be followed by name words.

This created a left boundary problem in case a word contained a copy of a name marker word. The original proposal was that no name could contain a phonetic copy of a name marker word. But the name marker words represent very common strings of phonemes, especially **la**. This *would* have solved the problem, but users of the language rebelled. **Laplas** is clearly a name. The reason that this **would** be a solution is that the end of a name could always be recognized, and it would begin at the first appearance of a name marker word or a pause, reading backward.

Our solution allows free use of name marker words in names (at the cost of some restriction on how these names appear) and nonetheless allows one to recognize the left boundary of a word. First of all, unmarked occurrences of names are eliminated. Unmarked vocative uses of names are simply banned. A vocative always has a vocative marker, and all the vocative markers are name markers. Name final descriptions such as **la bilti, Djin** are a source of unmarked name words: I require an explicit comma, and if the name component contains a false name marker one must use **ci** to mark it (**la sadji ci Laplas**). In serial names, predunit components, names following predunit components, and name components containing false name markers must be marked with **ci**. (It is worth noting here that name-final descriptions are *not* serial names. The distinction is clearly drawn in the trial.85 grammar and is somewhat more marked in ours)

Note that a false name marker is defined more tightly in our grammar. We do impose the condition that names resolve into syllables, and so we can gain by stipulating that a phonetic occurrence of a name marker in a name word is only a problematic false name marker if what follows it is itself a well-formed name. **la** in **Laplas** is a false name marker, but **ci** in **Uacinton** is not.

An occurrence of a name marker not followed by a pause in speech and then followed by an unbroken string of sounds ending in a consonant followed by a pause or silence is always a name marker followed by a name word. Orthographically, an occurrence of a name marker which may be followed by a space but not a comma, then followed by a string of letters ending in a consonant followed by a comma, terminal punctuation, another name, or **ci** is always read as a name marker followed by a name word.

There are occurrences of name markers which are not followed by name words. The name markers are all words with other uses. If a name marker word is immediately followed by a pause in speech or an explicit comma in writing, my parser views what follows as a name word only as a last resort. Notice that this does mean that it is presumed that a mere space after a name word does not represent a pause in speech: adding a comma may change the parse!

An occurrence of a name marker word which is not marked with a comma may indeed be followed by something other than a name word. In speech, this is indicated by the fact that the next pause or silence is not after a consonant. The parser now enforces this condition. If the next break after a name word is a mere space and not followed by a vowel (so that one cannot tell if it is a pause in speech or not) and the next break after that which is not of this ambiguous kind is followed by a consonant (it does not have to be comma marked: the end of a foreign name can raise this error), an error is raised by the parser, because one has not made a definite pause orthographically to guard a possible name. When I re-parsed Leith's novel after installing this feature, I found that it **did** discover errors of this kind now and then. Generally they can be avoided without actual attention to this rule by style directives such as, "always pause explicitly at the end of a predicate name".

I have stated in this essay my reasons for believing that I have fully solved the problem of recognizing the beginnings of names. Note that the parser does not require explicit commas at the ends of alien text constructions (strong quotations, foreign names, foreign predicates and onomatopoeia) because these are better guarded syntactically. I do believe that we should **write** the pauses after names and the explicit pauses needed to guard the fronts of names and avoid unintended formation of long names. I think that the pauses around alien text and the pauses before vowels other than those before logical connectives can safely be left unmarked as they have fewer global effects.

A footnote: a particular correction which occurred often in the Visit was to inverse vocatives: the form **hue la Selis** falls victim to this rule because it cannot be told from **hue Laselis**. The need for this can be avoided if **hue Selis** will do instead; in **hue, la Selis, cutse** one needs to pause (one can also say **hue la, Selis, cutse**). And if you do address Laselis, the space in **hue Laselis** does **not** represent a pause in speech. **hue, laselis** parses as **hue la Selis**, because the pause after **hue** signals that a non-name parse is preferred.

It is worth noting that the rules on pauses after a name marker word apply only when the parser is actually reading it as a name marker word. **ci** has uses which are not recognized as name markers at all: there is no problem with **La Meris, bilti ci cluva je la Djan** because the parser does not even think of the **ci** as a potential name marker and is in no danger of reading a pseudo-name **Cluvajeladjan** in this context.

6.3 Predicates

Predicate words fall into two classes, borrowings from other languages and complexes. We describe the class of borrowings first, but we note that when a predicate word is parsed, one first attempts to parse it as a complex, and only after that does one attempt to parse it as a borrowing.

A predicate must resolve into Loglan syllables.

All predicates have penultimate stress, meaning that they are always stressed on the last syllable but one, ignoring syllables not containing regular vowels (only one such syllable may intervene between the stressed syllable and the last syllable), and usually only on that syllable (note for those familiar with the language: it is permitted to stress the final syllable of a nonfinal borrowing djifoa before the **y** hyphen; notice that this stress is in a different place than the stress would be in the same borrowing standing alone), so of course they have at least two syllables. This helps one to determine where a predicate word ends. All predicates are vowel-final (so they are not names). All predicates contain at least one occurrence of two adjacent consonants (so they are not structure words).

The rules governing the beginning of a predicate word are designed to prevent ambiguity between a predicate word and a structure word followed by a predicate word. If the word begins CC there is no difficulty. A predicate word cannot contain more than one consonant before the first CC junction, because then the first consonant plus the stream of following vowels could be peeled off as a structure word or words. So the general form of the beginning of a predicate is an optional single consonant, followed by an optional string of vowels, followed by CC, where the CC does not itself begin a valid predicate. The CC cannot be a permissible initial pair broken by a juncture.

A vowel initial predicate appearing in a noninitial position in a sentence must be preceded by a pause (which must be written at least as a space and can be comma marked).

A predicate cannot begin VCCV where the CC is a permissible initial pair. This could just be stated as an arbitrary stipulation, but it is worth recording the reasons. The problem is not to do with such predicates themselves, but with desirable longer complexes: a complex beginning CVV-CCV- with the CC an initial pair would have its initial CV fall off.

No predicate can be of the form CCVV or CCCVV. The CCVV predicates are not allowed so that the six letter forms CVC-CVV of complexes with the CC at the boundary initial will not have the initial CV fall off. The CCCVV forms were outlawed for technical reasons: they would enormously complicate the borrowing algorithm because they would create an entirely new kind of borrowing tail. Also the form **kastrua** for "beaver" (replacing the illegal **kahstrua** originally in the dictionary) was thought convenient: the space of CVCCCV borrowings opening up by forbidding the CCCVV forms is arguably more useful (and less consonant-heavy). An aesthetic point is that allowing either of these forms would create a context where the disyllable pronunciation of an optional monosyllable is forced, which otherwise does not happen.

A predicate cannot begin with a syllable whose vowel segment is a syllabic pair. A syllabic pair cannot follow a vowel in a predicate. A predicate will not contain two successive syllables with syllabic pair vowel segments. All of these constraints have to do with the **function** of allowing syllabic consonants in borrowings, which is to provide an additional device for modifying a proposed borrowing so that it is not a complex, by extending a Cc to a Ccc in a way that cannot happen in a complex (this may also fix a first CC with an initial pair so that preceding material will not fall off, which would not work if an initial syllable with a syllabic consonant were allowed). It is then easy to see that there is no reason for there to be such a cc following a vowel, nor for there to be two successive syllables with syllabic consonants in a borrowing, or indeed more than one of them. And allowing some of these things can cause problems. Another similar gluing strategy is adding **h** to a borrowing (final in a consonant group, as it must be) tactically to create a CC pair where none is present or to break up the shape of a borrowing which otherwise might be a complex.

The parser recognizes the end of a predicate either by noticing an explicit stress then counting syllables to the end of the word, or by seeing a space or punctuation ending the word and checking that the syllabification allows the syllable before the previous one (skipping a possible syllable without a regular vowel) to be stressed. This means that in the absence of an explicit stress, some spaces are mandatory which do not represent pauses in speech (though they always occur at points where it is permissible to pause, I believe), but rather signal the presence of a stress.

6.3.1 Borrowings

The additional features of a borrowing over and above the general features stated above are that it cannot contain any occurrence of **y** or of any of the disyllables **aa**, **ee**, **oo** which force stress on one of their components (nor of **ii**, **uu** as disyllables). Further, it cannot be a complex, but this is not enforced by the parser rule for borrowings directly, but by attempting to read any predicate as a complex first.

There are additional technical condition on explicit syllable breaks using **-'*** in borrowings whose motivation is described below. The situation CV-CC is not permitted where the final CC is an initial pair, and a CC-C may not have the two consonants separated by the juncture forming a permissible initial pair. At least the first of these placements of syllable breaks is not forbidden in speech, and indeed the parser will often choose the first one as default in the absence of explicit syllable breaks; this is an essentially orthographic rule.

6.3.2 Complexes

A complex is like a structure word in being composed of units which are not themselves syllables and whose interaction with syllabification can be tricky. These units are called *combining forms* officially; traditionally they have been called *affixes*, a deprecated usage. In Loglan they are called **djifoa**.

It is required that a syllable does not overlap with more than one djifoa. This is enforced by the restriction given above on junctures in borrowings: a string differing from a complex only by adding syllable breaks that violate djifoa boundaries will exhibit one of the excluded behaviors, and so will not parse as a borrowing. The first of the two excluded behaviors for borrowings does occur explicitly in complexes.

The djifoa are of the following basic forms (where C represents a consonant and V represents a regular vowel). Each of the djifoa of one of the three letter forms is either an abbreviation for a five letter form (this information is in the dictionary) or is associated with a structure word (also in the dictionary) [we associate the CVh djifoa with the CV structure word appearing as an initial segment; these are unassigned in the sources and do not appear in the dictionary]:

CVV: Note that an initial CVV djifoa cannot be followed by another `CV-`form due to the general rules of predicate formation. This is fixed by allowing an optional "hyphen" to be appended to the djifoa. This hyphen may be **r**, or it may be **n** if followed immediately by **r** initial in the next affix, or it may be **y**. Note that CVV djifoa where the VV is **aa**, **ee**, or **oo** can only occur in final or penultimate position among the djifoa making up a complex, as one of the syllables of such a djifoa must receive the main stress in the word (and if the VV is **ii** or **uu** it can only be pronounced disyllabically in final or penultimate position). The fact that a CVV appearing in final position where the VV is an optional monosyllable can be syllabified in two different ways may cause there to be two possible ways of stressing a complex (a borrowing may have longer final strings of vowels admitting many arrangements of syllable breaks and stresses).

CCV: A CCV never needs to be hyphenated in a regular complex, but it will require a **y** hyphen if followed by a borrowing djifoa, as any djifoa does.

CVC: Of course a CVC cannot appear in final position among the djifoa in a complex.

A CVC which is initial in a complex will be followed by a **y** hyphen if the next affix begins with a consonant and the CC juncture created would be an initial pair, unless the word is CVCCVV or CVCCCV, which do not need this form of hyphenation (this avoids a CV form falling off the front of the word). The **y** hyphen is a single syllable by itself; in terms of djifoa analysis, it is treated as part of the CVC djifoa. It should be noted that though this is related to the general rule that the part of a predicate beginning with the initial CC cannot be a predicate unless it is the whole predicate, it is not a mere case of this rule, as it is not necessarily the case that dropping the initial CV in such a predicate without the inserted **y** would produce a legal predicate.

This rule is why TLI Loglan no longer has the **slinkui** test. It is not in 1989 Loglan but it is explicitly approved by the academy (with further official modifications in 2013).

A **y** hyphen may also be appended to a CVC djifoa to prevent formation of an illegal medial pair or triple of consonants with the following affix.

CCVCV: In non-final position, the final V is replaced by **y**. If a syllable break is expressed, it is CCV-CV.

CVCCV: In non-final position, the final V is replaced by **y**. If a syllable break is expressed, it may be CV-CCV or CVC-CV – the former is of course allowed only if the CC is an initial pair.

borrowing djifoa: A borrowing djifoa is a complete borrowing plus hyphens; **y** is added before it if it is not initial [the parser views this **y** as appended to the previous djifoa, so in fact any djifoa may need to be hyphenated] and after it if it is not final. A borrowing by itself is not a djifoa. The following **y** hyphen is regarded as part of the djifoa. A non-final borrowing djifoa is optionally stressed on its final syllable before the **y** (which is not the same as the stress on the borrowing itself!); this stress must be expressed if the djifoa is followed by a final monosyllabic djifoa (because in this case the stress is the main penultimate stress on the whole predicate). It is permitted to pause after the subsequent **y** hyphen if the stress is expressed (and to write an explicit comma pause there; the parser does not accept a space in this context). Recall that **y** never appears in a borrowing; this makes it clear that borrowing djifoa can be resolved.

A complex is a word which satisfies the general conditions to be a predicate and resolves into djifoa (where any phonetic hyphens used are regarded as part of the preceding djifoa). A single five letter djifoa is a complex (a primitive predicate).

6.3.3 The zao construction

John Cowan has proposed this as an alternative to the use of borrowing affixes, and I quite like it as an option. A sequence of predicate words separated by the word **zao** is grammatically a predicate word. There is no grouping in this construction any more than there is in the basic predicate construction. It is permissible for one or more of the initial items in a sequence of words linked with **zao** to form a predicate to be CV or CVV affixes.

I see one use of this construction as the ability to paraphrase a complex whose structure may be unclear to someone.

I have a preliminary suggestion that all CVh affixes can be regarded as associated with the CV structure word with which they begin.

6.4 Essay: Moving syllable breaks in borrowings

This is a brief essay on the odd-looking parser rule `JunctureFix`. The motivation is to make it impossible to turn an illegal complex into a legal borrowing by moving a syllable break.

Where a CVC is followed by CVx, moving the juncture from CVC-Cvx to CVCC-V creates a syllable break of the form C-V, which never occurs in a predicate. Moving the juncture from CVC-CVx to CV-CCvx is only possible if the CC is initial, so we do not allow (in a borrowing) an explicit juncture after a vowel and before an initial pair. It must be noted that where no explicit juncture is given the parser may covertly choose this syllable break! This is a rule of orthography, not a rule of pronunciation.

Where a CVC is followed by CCV, moving from CVC-CCV to CV-CCCV is only possible if both consecutive pairs of consonants are initial, and the same restriction ruling out a vowel followed by a juncture followed by an initial pair works. Moving from CVC-CCV to CVCC-CV creates the other situation ruled ut by `JunctureFix`, a consonant followed by an initial pair broken by a juncture.

Where a CxV is followed by Cvx, moving the juncture would put it in front of a vowel.

Where a CxV is followed by CCV, moving the juncture would give a juncture before a vowel or the V-CC situation already ruled out.

Now there is a converse issue: does forbidding these situations create any difficulty with being able to place syllable breaks in borrowings?

Where V is followed by a juncture followed by CC, an initial pair, changing to VC-C is always possible no matter what follows.

Where CC-C occurs with the broken pair initial, changing to C-CC is always legal unless the situation is CC-CCC, which of course cannot occur in a complex anyway. We add the condition that what we forbid is CC-CV where the broken initial pair is followed by a vowel.

We allow the break in **i-gluu** where the second consonant of the initial pair is the first of a syllabic consonant pair, a situation which cannot occur in a complex.

7 Word Forms

This section deals with details of Loglan that are for the most part not manifest in the previous official formal grammar. The word class definitions are nowhere actually given formally; they are implicit in tables internal to the old interactive parser which are not human-readable and clearly have bugs.

Our program in desigining the PEG parser was to parse Loglan from the level of letters upward, and as a result we have had to mandate exact formal definitions for these word classes, which in some cases are clearly not exactly the same as those implicit in LIP. Generally our definitions are a bit more liberal, allowing more words. Details will be seen below.

Quotation constructions and other constructions which import foreign text are handled in this section. My implementation of strong quotation is a completely new proposal.

It is very important to articulate the concept of "word" formally. As JCB says in NB3, the defining characteristic of a word is that one cannot pause in the middle of it. He says this in the abstract, but then does not give us any formal definition of cmapua words: the definitions of word classes in LIP are part of the internals not expressed in the formal grammar we inherited. We are told that in Lojban there are no cmapua words with more than one syllable in this sense. This is not the case in TLI Loglan. Certain cmapua classes are genuinely classes of words, in that one cannot explicitly pause in the middle of a production of this class. The parser now forbids words to be written with spaces in them (so one cannot write **le na hasfa**: **lena** is a word). Earlier versions of my parser and LIP permit **le na hasfa** but note that they did not permit **le, na hasfa**.

It may very well be that with further work we could achieve the situation reported in Lojban where a stream of one-syllable cmapua is understood without reference to any pauses that may occur between syllables, so that there are no multi-syllable cmapua which are words in this sense. We further note that we do regard it as unfortunate when the placement of a pause in a stream of cmapua syllables materially affects meaning, though we observe some situations where this seems difficult to avoid (the classic `le, po` problem has now been resolved, though). We do not however regard the concept of "multi-syllable cmapua word" as alien to Loglan: JCB clearly envisaged there being such words.

7.1 Pauses

In 1989 Loglan, certain pauses were interpreted as GU and so had grammatical effects. This feature is no longer supported (though the grammar is structured in such a way that it could be turned back on, wholly or partially, for experimental purposes).

Pauses do not occur in the middles of words (with the exception of PA and NI words, and after borrowing affixes in complex predicates).

Pauses (expressed as commas or otherwise) are required in certain phonetic contexts as discussed above.

7.2 Structure words

We begin by considering the many classes of structure words.

7.2.1 Logical connectives for sentence components

There are numerous parallel classes of logical 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.

The basic series of connective roots is **a, e, o, u, 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 prefex **nu** may appear initially to a logical connective word of the basic series; it may only appear if followed by **u** or **no**. The root taken from **a, e, o, u, ha** (possibly with prefixed **no** and/or affixed **noi**, i.e., an A core) follows this. Finally a complete PA word (a tense in the broadest sense) with no internal pauses or spaces may follow as a suffix; finally, an optional **gu** may appear to close the word (for example, to prevent it from absorbing a following PA word). An A word may not be followed by whitespace then a PA word; an explicit comma pause must intervene. APA, IPA, CAPA, ICAPA words do not need to be followed by explicit pauses as in earlier versions of my parser.

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 because the related IPA words, which present much the same difficulties of termination, are clearly not dispensable without doing some violence to the corpus. The current solution seems to be quite neat; earlier approaches that I tried were rather baroque.

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 **nu** converts a logical connective to its converse. 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 `gu` 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. They are not preceded by pauses. The CA root forms are **ca, ce, co, cu, 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, 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; we will see if they are useful.

7.2.2 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, optionally followed by **gu**. The same issue exists for semantics of IPA words that is discussed above for APA wprds/ 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 connnective 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**.

The closure of logical connectives with **gu** is a new proposal here.

7.2.3 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** (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!

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

7.2.4 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 NI0 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).

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 ???; new proposal 10/17/2015),

sunoi: (less than; a prefix to a quantifier, by itself ???; new proposal 10/17/2015)

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 of numeral units consisting of a numeral (any word of class NI0 but this really makes sense only for the digits) 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 `moa` n means D followed by 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 NI0 words, with internal whitespace or explicit pauses permitted. It may optionally be preceded by a SA word.

RA: A RA word, which may optionally be prefixed by a SA word (this last option is a change from 1989 Loglan). 11/14/2015 update allows a RA word to be suffixed with **mo** and/or **moa** 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** with (resp.) **sarara**, **sinera** is an occasional correction needed in old texts.

A general quantifier word has a quite complex definition. It may optionally begin with **ie**. It is then followed by a quantifier core as described above. This may optionally be followed by an acronym which must start with the marker **mue**; 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.

The word **ie** used as a prefix here is an interrogative word meaning which. 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 avoid. The word "mex" (abbreviating mathematical expression) is used in the grammar section for quantifier words.

7.2.5 Letters, acronyms, and pronouns

A Loglan upper case consonant letter is C**ai**. A Loglan lower case consonant letter is C**ei**. A third series C**eo** is provided for lower case Greek letters. Further series C**aiu** and C**eiu** 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 **zi**V, and the upper case form is **zi**V**ma**. The old style forms V**fi** and V**ma** are currently supported in the parser but deprecated.

Other letter forms found in the sources are no longer supported.

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: (this [of situations]),

tio: (that [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 nth description back in the text not already bound to a pronoun will be bound to the nth letter on this stack when needed. It seems rather baroque but very simple cases can surely be used correctly.

The general class of pronoun words consists of letters or other pronouns, optionally suffixed with **ci** followed by a NI1 unit (usually a digit; not a multidigit numeral). It is very important to notice that for us a pronoun is a **single letter**, possibly suffixed with a numeral. 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.

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 is grammatically far more important than any use of sequences of letters as pronouns or acronyms.

Further letter words, which may be used as pronouns, but to which we may not attach numerical suffixes (? I may want to allow this), 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.

An acronym is a sequence of letter names (possibly abbreviated in the case of vowels to zV – not to just V – 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 component (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 noted above) 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).

We add as a footnote a remark on why we do not like the VCV letterals. If VCV letterals were used in acronyms, as in **la daiafi**, the analysis of this into phonetic cmapua units would have 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 such words one does not need to explicitly comma pause; they are treated in the same way as vowel-initial predicates. We note that we fixed a bug which caused the parser not to accept the VCV literals in acronyms.

7.2.6 Tense/location/relation operators

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

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

gua: (timeless habitual tense),

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

pua: (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 : (???I cannot find any indication of what this means, though there is a reference to it in text of trial.85),

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),

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: (???),

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),

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),

rui : (???),

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

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.

A compound PA word begins with an optional numeral or quantifier, followed by a string of PA roots or KOU cores taken from the lists above (recalling that KOU cores may include certain prefixes), optionally linked to further strings of PA roots/KOU cores by CA cores, then may optionally be closed with one of the qualifiers **za, zi, zu** (to see the effects of these qualifiers on tense and location operators, see the dictionary). Whitespace or explicit comma pauses may occur after PA roots, KOU cores, or CA cores in a compound PA word (but not immediately before a final ZI).

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 words can be used as prepositions (followed by an argument) or as tenses in the broadest sense (followed by a predicate). 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, it is generally illegal for it to be replaced by whitespace followed by a PA word (an explicit comma pause is forced): where a PA suffix is legal, it cannot be replaced by a following PA word without an explicit pause being indicated.

7.2.7 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.

vuza: 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 word (this is true in my grammar, and experiment confirms that this happens with LIP as well). In a word such as **fanazu**, what does the **zu** qualify? It seems most reasonable to suppose that in a word **pacenazu**, the **zu** qualifies both conjuncts. The current grammar does not allow logical conjunction of PA cores with different **zV** qualifiers to form words.

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

7.2.8 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 with property X...

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

la: The article for proper names.

These are the root words. Complex words of this class can be formed by following the root with an optional pronoun followed by an optional PA suffix. Spaces may not intervene between these components.

The name constructor **la** and complex articles derived from it appear in the list above but appear 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. These are forms to which I admit having paid very little attention.

7.2.9 Constructions involving alien text and related articles

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). It is also permissible to quote serial names: in this case the comma before the name is mandatory. **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.

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 of the various classes in this section. 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 pause may sometimes be required to terminate a quoted word where you want it terminated. **niu** may be used instead of **liu** to explicitly signal that a quoted word, though phonetically acceptable, is not a Loglan word. I have just installed the correction allowing **liu ziy** and **liu ziyma** to parse.

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. Whereever 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 termnology 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 **cii** before each subsequent block. The level of quotation (quotation, quotation of a quotation, etc) can be modified by suffixing a numeral directly to **lie** or **cii** (and it can change from block to block; where a change occurs the numeral must be explicitly given (the only case where **ne** is required)). This device was added because this technique of quotation would become really burdensome if nested quotations were made explicitly. This is an entirely new proposal. The original strong quotation method is not PEG parsable (it is not even BNF parsable) and I think has other weaknesses. The parser allows the use of **y** instead of **cii**, and I believe I prefer this usage, analogous to the use of **y** as a separator in foreign names with **lao**.

I add a comment here not yet implemented: I am considering giving **lie** the same grammar as **lao** (allow it to be followed by blocks of alien text separated by **y**). The fancy ideas about iterated quotation are less of an issue than I thought when I drafted these ideas: observe that **lie** quotes are legitimate Loglan utterances and so can be quoted with **li/lu**; complex situations can be handled by further noting that string concatenation can naturally be represented by **ze**. For the moment I am keeping my full proposal, which does support the limited style I am suggesting.

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

We support in the grammar without necessarily approving (also without necessarily disapproving; I know some Keugru members do not like them) the qualifiers **za** (text) and **zi** (sound) for quoted forms. The qualifier is to be affixed to the initial quotation article (after any suffixed numeral in the case of **lie**).

7.2.10 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**

also the new **guoa, guoe, guoi, guoo, guou**.

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)

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: (greaters),

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, but they have the same grammar as case tags. 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.

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.

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

7.2.11 Words which form free modifiers

The register markers indicate attitude toward the person addressed:

die: (dear),

fie: (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.

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.

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 :-)

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.

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**. [I have recently made these words additional vocative markers and thus 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 wiht **noi**.

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

7.2.12 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.

7.3 The Large Word Classes

There remain the large classes of predicate and name words.

7.3.1 Predicate words

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)** are all predicates semantically, though they are structure words phonetically. They form a grammatical class BI of identity predicates.

I propose adding to this class all the forms obtained by prefixing **nu**, giving converse operators (the parser allows this).

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! (some essay will be needed here).

The class PREDA of predicate words includes the last list.

The class PREDA includes quantity words suffixed with **ra, ri, ro** to form numerical predicates (cardinal, ordinal, quality ordinal, respectively). A predicate **tora** is a two place predicate, X is a two element subset of Y; **tora** is a two place predicate, X is the second term in series Y. I do not know what the quality ordinal predicates are supposed to be like (homework for me to do). The numerical predicates should be penultimately stressed just as ordinary predicates are [the grammar currently does not enforce this].

When a numerical predicate is modified by a quantifier, the little word **ge** must intervene to avoid the modifier from eating up the numerical predicate (NI words can include spaces or even pauses). **ne ge tori**, not **ne tori** or even **ne, tori**. I have had to make this change in old texts.

And of course the class PREDA includes the predicate words in the phonetic sense of the first section.

7.3.2 Borrowing predicates

A section which is owed.

7.3.3 Making complex predicates

A section which is owed.

7.3.4 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 statement 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.

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.

8.1 Sentences and Utterances

This corresponds to the last part of the trial.85 or loglanagenda 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) 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. The set of terms is even more usually a single argument (the subject of the sentence) but more than one argument may be followed by the predicate. The initial list of terms must include at least one argument, or the sentence will be understood as an imperative. **Da de blanu** has the same meaning as **da blanu de**, "X is bluer than Y".

Another alternative is the gasent, a VOS 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 gasent, 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 contain either exactly one argument or all the arguments in the sentence. 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 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 a raindrop!". When the final **ga**+ terms is omitted, a missing **ga ba** is understood.

As explained below (class sen1), an unmarked predicate (which may include following terms) possibly preceded by one or more terms not including any arguments is an imperative sentence: **Donsu ta mi**; "give that to me"; **Na la Ven, donsu ta mi**: "At nine, give that to me". Marking the predicate with a tense makes it a declarative sentence with an indefinite subject: **Fazi donsu ta mi**, "Someone will shortly give that to me". This is to be understood as **Fazi donsu ta mi ga ba**. The sentence **Fazi donsu 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 gasent.

NOTE TO SELF: the freemod after the initial no in gasent might profitably not be allowed to be a pause, to avoid notional ambiguity with neghead. This might apply in other places. I'm not sure there is any semantic ambiguity.

All three of these forms are options in the grammar rule "statement": it looks for a gasent first, then a gasent with initial modifiers, and only then for a sensible SVO sentence. Notice the reordering of what is going on by the hearer when **di fa donsu de...** (which sounds as if Z will give Y to ...) is completed **di fa donsu de ga da**: this is why we actually wish to forbid such sentences.

Underlying this is the form $Px_1, x_2, \ldots, x_n$ of a 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 $x_1Px_2, \ldots, x_n$. In what I regard as a much more dubious decision, this was grouped $x_1[Px_2, \ldots, x_n]$, 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 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 form is a keksent – I will defer describing this for a moment.

A sen1 is a statement, solitary predicate without a tense marker (an imperative sentence, as noted above), or keksent.

A sentence is a sen1 followed by zero or more logical connectives of class ICA followed by a sen1: **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 logically connected sentence in Loglan there will be connectives between each pair of adjacent sentences.

An 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 logically connected subsentences if indeed the sentence is logically connected (which is why an ability to close the sequence is needed). Head terms can also 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. Thee is a more thorough proposal along these lines in the list of proposals below in the Report.

A keksent is an optional negative **no**, followed by a word like **ke**, followed by a sentence or uttAx, followed by **ki** followed, surprisingly, by the very general 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".

NOTE TO SELF: the actual way that keksent is written in the grammar seems to have strange options in it, and I dont know why uttAx is inlined rather than being mentioned.

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 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".

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**. NOTE TO SELF: should NODIE be a separate class which NO1 is guarded against?

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.

inverse vocative: hue followed by a name (including foreign names, 1/10/2016), a statement, a descriptive predicate (with optional following name: **hue bilti** works, and so does **hue bilti, Djin**) or an argument list, indicating who is speaking. 11/14/2015, 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.

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".

cancelpause: A sequence of the form [**comma**] **cuu** or [**comma**] **y** [**comma**] (the second form is borrowed from **la Sorme Lengu**). The intention is to allow an unintentional pause which might otherwise be construed as significant (equivalent to **gu**) to be cancelled harmlessly. It could also be used to support pausing for effect. Obsolete though not yet deleted, as pause/GU equivalence is no longer supported. It may still have uses, as after a name marker if one did not intend to pause.

pause, ellipsis, hyphen A comma pause, not significant. Ellipsis . . . and hyphen `--` can also appear as freemods, possibly preceded by spaces, possibly followed by spaces up to a letter (update of 1/10/2016).

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 uttA class utterance is a connective, the word **ie** (how many?), or a number. These can only occur as utterances as answers to questions.

An uttA1, a very general class of utterances already mentioned in the keksent class above, may be a sen1 class sentence, an uttAx class sentence **no** by itself, a tightly bound argument list beginning with **je** or **jue**, an argument modifier (subordinate clause, class argmod), a list of terms followed by a keksent (NOTE TO SELF: this I do not understand; it is there in trial.85), a list of arguments, or an uttA. This may be terminated with a period or other final punctuation.

My best guess is that in the terms followed by the keksent, the initial arguments or modifiers distribute over all logically/causally connected components of the keksent. It might be intended that the initial terms be a modifier, but the rule is there with general terms. NOTE: I intend to change the initial terms to `modifiers`.

I think the uttA1 class, which includes a lot of utterance fragments, often serves to provide a form for answers to questions. But it does include the sen1 and uttAx classes of complete sentences so it can include quite general utterances. And it enters into the makeup of the keksent class. Some of the forms it permits for keksents 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.

NOTE: I would like to have 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.

An uttC is an uttA1 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!); an initial **no** will otherwise be absorbed into some shorter structure at the beginning of the utterance.

An uttD is 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 TO SELF: there is a sort of ambiguity here. The class sentence is formed by linking sen1 sentences with ICA connectives, as in **Mi blanu, ica tu redro**. But this utterance standing by itself will be read as an uttD. Food for thought, though there is no true ambiguity I suppose: both utterances mean the same thing.

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

ige construction: An uttE 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.

8.2 Predicates

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

8.2.1 The basic building blocks of predicates: predunit classes

The class 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.

2. a conversion (or reflexive) operator followed by **ge** followed by a despredE (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 despredE closed off optionally by **geu** or **cue**.

5. An abstraction forming word like **po** followed by an uttAx sentence 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].

9. a predicate word (see above in the lexicography section).

Any of the above forms of predunit1 may include free modifiers in all medial positions and terminally.

A predunit2 is formed by affixing one or more **no**'s (possibly followed by free modifiers) to the front of a predunit1.

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

A predunit is either a predunit3 or a predunit3 preceded by a short-scope PO operator such as **poi**. The predunit 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 Cmalo Hasfa** (John the Small House), but not *__la Djan ci Blanu Tu__ or *__la Djan ci Cmalo Hasfa__. It is important to notice that modification of one predicate by another can occur in a predunit only inside a **ge...(geu)** block.

A 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 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".

8.2.2 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 predunits and kekpredunits 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 big (thing)". 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 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 despredE class is of special note as being the sort of predicate which can be enclosed in **ge...(geu)** to form a predunit1

A descpred, the top level class of predicates used in descriptions, is either a despredE or a despredE followed by **go** followed by a descpred, 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).

8.2.3 Sentence predicates, first pass

The basic sentence predicates belong to classes senpred1-4 exactly analogous to despredA,B,D,E, with the following differences:

1. a senpred1 contains no kekpredunit components (query: why do we not allow non-initial kekpredunit components?)

2. a senpred2 is a senpred1 or a despredB beginning with **cui**.

3. a senpred3 is either a senpred2 or a senpred2 linked by a CA word to a despredD.

4. a senpred4 is a senpred3 or a senpred3 followed by a chain of despredD's. Free modifiers or pauses may appear between components of this chain.

The motivation of these rules is that we do not want kekpredunit-initial chains of modifiers and we do allow pauses between items in a chain of modifiers. The use of description predicates as components does rule out some pauses that might be harmless.

A sentpred is a senpred4 or a senpred4 followed by **go** followed by a barepred. This is more general than a descpred because a barepred may have a list of arguments attached (loosely rather than with JE/JUE).

8.2.4 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 an argument, or list of arguments, or a structure built by logical connection of simpler termsets.

A kekpred is a kekpredunit followed by a chain of despredD's with optional intervening free modifiers. (I am a little puzzled by this: it appears that as of trial.85 we are allowed to have kekked head modifiers in sentence predicates, with precautions to prevent their falling apart: notice that a kekpred is a description predicate in form. I have now looked at trial.85: note 75 characterizes the desired behavior, which my parser does indeed replicate; my objection is that it appears that the trial.85 grammar is demonstrably ambiguous as written once this change is made; I can exhibit distinct parses of the same utterance; this is not a problem for my grammar or indeed for my understanding of the intention here, as the PEG as written appears to do exactly the intended thing as described in note 75; compare the parses of **da ka redro ki blanu hasfa** and **da ka redro ki blanu gu hasfa**).

NOTE: I do wonder if this change allowing kekked head modifiers means that the sentpred/descpred distinction should be seriously rethought. It is still important not to allow pauses in metaphor strings in descriptions, but that reluctance could also be eliminated by not allowing certain already deprecated sentence forms. This note is not saying that the grammar doesn't work, but that it might be seriously simplified.

A barepred is a sentpred or kekpred followed by an optional termset (which may have a non-pause free modifier before it). This is the class which can appear after **go** in a sentpred.

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

A backpred1 consists of one or more **no**'s of negation (with following optional free modifiers) followed by a barepred or markpred. 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 barepred 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 (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). It is important to notice that before 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 solitary **guu** as a termset is actually quite useful). 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) or a structure build by linking general predicate2’s with A afterthought connectives and adding a termset (shared by all the logically linked predicate2’s). 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 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.

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

8.3 Clauses, arguments and term lists

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

8.3.1 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** are the "name markers". A name word must be preceded either by a pause or by a name marker.

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**
2. `ci` (possibly preceded by a free modifier) followed optionally by a pause followed by a predunit; 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 predunit).

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 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.

8.3.2 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.

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

A vocative is either **hoi** followed by an optional comma pause followed by a serial name followed by an optional gap, or **hoi** followed by an optional free modifier followed by a descpred (with optional following name, as in **hoi bilti, Djin**) followed by an optional gap, or **hoi** followed by an optional free modifier followed by an argument followed by an optional gap, or **hoi** followed by alien text (construed as a foreign name, added 1/10/2016). 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 added **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 new vocative markers 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 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 descpred. 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 descpred. Freemods are allowed in both medial positions. **Le to mrenu** is a simple example.

3. A LE word followed by an optional freemod followed by an arg1 (a particular flavor of argument to be revealed) followed by a descpred. **Le le hasfa, mrenu** is an example. This is a sort of possessive.

4. A LE word followed by a mex followed by an arg1a (another subtle flavor of argument; a difference to be noted is that a pronoun is an arg1a but not an arg1) [with freemods insertable in medial positions]. An example is **Le to le mrenu** We can have **Le to mi**, whereas we cannot have **Le mi hasfa** under the previous heading (though we do have **Lemi hasfa**, with the right meaning, because **lemi** is a LE word).

5. **ge** followed by a mex followed by a descpred. I need to firm up my understanding of what this case is for.

An 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 an uttAx, 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 these cases are now included in a separate subclass `abstractn`.

4. **lio** followed by either a descpred 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 preferred to descriptn, unless there is a comma marked pause after the **la**, in which case the parser attempts non-name readings first. As above, **la Has'faran** is a name.

7. a descriptn, optionally followed by a non-pause freemod, 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, all closed optionally by a gap.

   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 in forms of address: **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 a 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 arg1a of arguments consists of the following kinds (tested for in this order):

1. a DA pronoun

2. a TAI letteral pronoun

3. an arg1

4. **ge** followed by an optional freemod followed by an arg1a. I need to understand the use of prefixing **ge** here.

An arg1a of any of these shapes may further include a following non-pause freemod.

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

We now introduce argument modifiers (subordinate clauses).

An argmod1 (atomc 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 uttAx

3. a JI word followed by a modifier (a relative clause)

4. a JI word followed by an argument

Any of these forms are optionally closed by GUI (the word **gui**, optionally attended by commas before and/or after, or a gap).

A construction of the class argmod is a series of argmod1's linked by A logical connectives (NOTE: the gap in the argmod class definition seems to be redundant).

An argument of class arg2 is an arg1a or an arg1a followed by one or more argmods. NOTE: Again, the optional gap in the arg2 rule may be redundant. NOTE: Should there be a medial freemod in the arg2 rule? I think there may be a reason not to have one. NOTE: look carefully at closure of argmods, arg2's.

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 descpred (e.g., **to mrenu**).

An indefinite is an indef1 followed by an argmod. 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 a positional or semantic case tag, **lae** or **lue**, or the operator **ie** (interrogative which)). 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 cmapua **ge**.

An argument is a set or list form (to be discussed later [very possibly not at all in this draft]; this construction is one I have done little analysis of) or 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). NOTE: I am very suspicious of the fact that explicit set and list forms are not allowed to enter into any complex argument constructions. I expect that when I do pay attention to them I will entirely change their role in the grammar, much as I did with (PO sentence) predicates. I suspect that AGE connectives should group to the right as AGE predicate connectives certainly do.

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 is a case of arg1]. I have an overall impression that closures of constructions involving argmods would benefit from an overhaul.

8.3.3 Explicit set and ordered list forms (omitted for now)

Omitted for the moment. These require analysis; they are not first-class citizens of the language in the grammar as presented in trial.85.

8.3.4 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. 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.

8.3.5 Terms, term lists, and termsets (including linkargs)

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). A construction of class terms is the same except that medial freemods may not be pauses.

A termset1 is a construction of class terms optionally closed with GUU (either **guu** optionally flanked with explicit commas on one or both sides or a gap) or a forethought construction: a KA word followed by a termset2 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.

A termset is one of the following:

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

2. a termset2

3. GUU = the word **guu** (optionally flanked by pauses on one or both sides) or a gap. This null case is actually quite useful where logically shared termsets are in play.

Termset is an important class, having been introduced earlier as a black box internal feature of the predicate classes. The modification of termset to use terms instead of terms (and forbidding pause as a freemod between a predicate and the attached termset) is designed to allow tests of pause/GU equivalence.

We now consider the tightly bound lists (culminating in 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 (NOTE: the gue in this rule might be problematic, forcing double closures in some situations).

A 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. Recall that the class linkargs enters into the construction of predunits.

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 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.

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 ¡- JE term from jelink ¡- JE argument

(leaving out freemods for clarity)

This is clearly grammatically harmless and allows much finer use of modifiers (PA clauses).

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 cmapua. 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 donsu ti mi** is no longer a deprecated imperative, but instead is to be understood as **Fazi donsu ti mi (ga ba)**, "Someone is about to give this to me". **Vi le hasfa fazi donsu 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 could be enforced by grammar (I provide a demonstration parser in which this is achieved) or by a style directive, leaving the peculiar sentences in which retrospective reassignment of arguments happen as an unlikely option. I also provide a parser which has this behavior.

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.

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/foregn 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 (in fact, other grammar changes elsewhere make such pauses illegal), 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.

A particular point to be examined is the treatment of free modifiers, with its implicit partial implementation of pause/GU equivalence. The Academy may wich to abandon the possibility of pause/GU equivalence, which would be straightforwardly managed by removing pause as a case of GU and converting all occurrences of `(|PAUSE freemod)?` to `freemod?`. Some language on this point which may need to be articulated as a separate proposal should be added.

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 unilikely "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 ny 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.

This proposal is fully implemented in my parser. 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.

10 Appendix: Annotated Provisional PEG Grammar

This is the grammar of 1/30/2016: enough changes had been made that I thought piecemeal editing wasn't suitable. However, the inserted comments are mostly the original ones and surely need updates. There is a good deal of clutter in the grammar (and in the remarks) that should be cleaned out.

```
lowercase <- (!([qwx]) [a-z])

uppercase <- (!([QWX]) [A-Z])

letter <- (!([QWXqwx]) [A-Za-z])
```

The Loglan alphabet. **q, w, x** are excluded.

```
juncture <- (([-] &(letter)) / ([\'*] !(juncture)))

stress <- ([\'*] !(juncture))

juncture2 <- (((!stress juncture &(letter)) / (stress !((([ ])* Predicate))
((', ' ([ ])* &(Predicate)))?)) !(juncture))
```

Syllable breaks (including stress markers). The hyphen must be followed by a letter. The stress markers may not be followed by another juncture. The form `juncture2` is used to enforce the rule that a stress at the end of a cmapua and preceding a predicate must be followed by a comma pause.

```
Lowercase <- (lowercase / (juncture (letter)?))
```

A lowercase letter or a juncture (possibly followed by a capital letter) ; what can occur in a word after the initial letter which might be capitalized.

```
Letter <- (letter / juncture)
```

An upper-case letter or a juncture.

```
comma <- ([,] ([ ])+ &(letter))

comma2 <- (([,])? ([ ])+ &(letter))

end <- (([ ]* '#' [ ]+ utterance)/
     ([ ]+ !.)/!.)

period <- (([!.:;?] (&end / (([ ])+ &(letter))))
 ((&(HUE) freemod (period)?))?)
```

Punctuation. A comma must be followed by whitespace followed by a letter. The class `period` is inhabited by terminal punctuation, all equivalent for the grammar so far, which may be followed either by the end of text or whitespace followed by a letter. An element of class `period` may have an appended inverse vocative (optionally followed by another `period`: installed 12/21/2015 to support Alex Leith's usage in his novel.

The silence/change of voice marker `#`, which we take from JCB's notation, is not really a piece of Loglan punctuation. It can never be quoted as part of Loglan text.

```
B <- [Bb]

C <- [Cc]

D <- [Dd]

F <- [Ff]

G <- [Gg]
```

```
H <- [Hh]

J <- [Jj]

K <- [Kk]

L <- [Ll]

M <- [Mm]

N <- [Nn]

P <- [Pp]

R <- [Rr]

S <- [Ss]

T <- [Tt]

V <- [Vv]

Z <- [Zz]
```

Consonant letters irrespective of capitalization.

```
a <- ([Aa] !Mono (juncture2)?)

e <- ([Ee] !Mono (juncture2)?)

i <- ([Ii] !Mono (juncture2)?)

o <- ([Oo] !Mono (juncture2)?)

u <- ([Uu] !Mono (juncture2)?)
```

```
AA <- ([Aa] (juncture)? [a] (juncture2)?)

AE <- ([Aa] (juncture)? [e] (juncture2)?)

AI <- ([Aa] [i] (juncture2)?)

AO <- ([Aa] [o] (juncture2)?)

AU <- ([Aa] (juncture)? [u] (juncture2)?)

EA <- ([Ee] (juncture)? [a] (juncture2)?)

EE <- ([Ee] (juncture)? [e] (juncture2)?)

EI <- ([Ee] [i] (juncture2)?)

EO <- ([Ee] (juncture)? [o] (juncture2)?)

EU <- ([Ee] (juncture)? [u] (juncture2)?)

IA <- ([Ii] (juncture)? [a] (juncture2)?)

IE <- ([Ii] (juncture)? [e] (juncture2)?)

II <- ([Ii] (juncture)? [i] (juncture2)?)

IO <- ([Ii] (juncture)? [o] (juncture2)?)

IU <- ([Ii] (juncture)? [u] (juncture2)?)

OA <- ([Oo] (juncture)? [a] (juncture2)?)

OE <- ([Oo] (juncture)? [e] (juncture2)?)

OI <- ([Oo] [i] (juncture2)?)

OO <- ([Oo] (juncture)? [o] (juncture2)?)
```

```
OU <- ([Oo] (juncture)? [u] (juncture2)?)

UA <- ([Uu] (juncture)? [a] (juncture2)?)

UE <- ([Uu] (juncture)? [e] (juncture2)?)

UI <- ([Uu] (juncture)? [i] (juncture2)?)

UO <- ([Uu] (juncture)? [o] (juncture2)?)

UU <- ([Uu] (juncture)? [u] (juncture2)?)
```

Vowels and vowel groups independent of capitalization and presence of medial or final junctures. Note that `juncture2` is used here in final position, because these classes are used to build cmapua syllables.

The vowel in a CV cmapua syllable cannot be followed directly by another vowel which would make a monosyllable. This defends the Cvv-V cmapua units. When a CV word is followed by a VV word without a pause, a syllable break must be used to indicate this.

```
V1 <- [AEIOUYaeiouy]

V2 <- [AEIOUaeiou]
```

Classes of vowels. `V1` includes the irregular vowel **y**; `V2` is the class of regular vowels.

```
C1 <- (!(V1) letter)
```

The class of consonant letters.

```
Mono <- (([Aa] [o]) / (V2 [i]) / ([Ii] V2) / ([Uu] V2))
```

```
EMono <- (([Aa] [o]) / ([AEOaeo] [i]))
```

Classes of monosyllabic vowel pairs. `Mono` is the class of pairs which can be monosyllabic; `EMono` is the class of pairs which must be monosyllabic.

```
NextVowels <- (EMono / (V2 &(EMono)) / Mono / V2)
```

This class defines the next syllable to be extracted from a long chain of vowels in a name or borrowing. Exclusive monosyllables are chosen first, then single vowels which are followed by exclusive monosyllables, then optional monosyllables, then the next vowel.

```
BrokenMono <- (([a] juncture [o]) / ([aeo] juncture [i]))
```

This is an exclusive monosyllable forcibly broken by a juncture.

```
CVVSyll <- (C1 Mono)
```

A CVV syllable.

```
LWunit <- (((CVVSyll (juncture)? V2) /
 (C1 !(BrokenMono) V2 (juncture)? V2) /
 ([Zz] 'iy' (juncture)? ('ma')?) /
 (C1 V2)) (juncture2)?)
```

Phonetic units of a multisyllable cmapua according to the definition in NB3. The names **ziy** and **ziyma** of the letter **y** are included as irregular permissible units.

```
LW1 <- (((V2 V2) / (C1 !(BrokenMono) V2 (juncture)? V2) /
 (C1 V2)) (juncture)?)
```

The structural units of cmapua, VV, CVV independent of syllablehood, CV, with possibility of medial and final appearance of junctures.

```
caprule <- ((uppercase / lowercase) ((('z' V1) /
 lowercase / (juncture (caprule)?) / TAIO))* !(letter))
```

A capitalized sequence of letters and junctures. A letteral may appear independently capitalized in such a sequence: this supports usages like `leSai` found in Leith and other Loglan texts. It does allow odd capitalizations of phonetic occurrences of letterals in other contexts. Similarly, a vowel may appear capitalized after lower-case z to support internal capitalization in acronyms, and any letter may appear capitalized after a juncture to support natural capitalization in certain compound names (and possibly other compound contexts).

```
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')
```

The consonant pairs which are allowed to appear in initial position in a syllable.

```
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] (juncture)? [r]) / ([Gg] (juncture)? [l]) /
 ([Gg] (juncture)? [r]) / ([Jj] (juncture)? [m]) /
 ([Kk] (juncture)? [l]) / ([Kk] (juncture)? [r]) /
 ([Mm] (juncture)? [r]) / ([Pp] (juncture)? [l]) /
([Pp] (juncture)? [r]) / ([Ss] (juncture)? [k]) /
([Ss] (juncture)? [l]) / ([Ss] (juncture)? [m]) /
 ([Ss] (juncture)? [n]) / ([Ss] (juncture)? [p]) /
 ([Ss] (juncture)? [r]) / ([Ss] (juncture)? [t]) /
 ([Tt] (juncture)? [c]) / ([Tt] (juncture)? [r]) /
([Tt] (juncture)? [s]) / ([Vv] (juncture)? [l]) /
([Vv] (juncture)? [r]) / ([Zz] (juncture)? [b]) /
 ([Zz] (juncture)? [v]) / ([Zz] (juncture)? [l]) /
 ([Ss] (juncture)? [v]))
```

A pair of consonants which is either an initial pair or an initial pair separated by a juncture.

```
NonmedialCC <- (([b] (juncture)? [b]) /
 ([c] (juncture)? [c]) / ([d] (juncture)? [d]) /
 ([f] (juncture)? [f]) / ([g] (juncture)? [g]) /
 ([h] (juncture)? [h]) / ([j] (juncture)? [j]) /
 ([k] (juncture)? [k]) / ([l] (juncture)? [l]) /
 ([m] (juncture)? [m]) / ([n] (juncture)? [n]) /
([p] (juncture)? [p]) / ([q] (juncture)? [q]) /
 ([r] (juncture)? [r]) / ([s] (juncture)? [s]) /
 ([t] (juncture)? [t]) / ([v] (juncture)? [v]) /
 ([z] (juncture)? [z]) / ([h] (juncture)? C1) /
 ([cjsz] (juncture)? [cjsz]) / ([f] (juncture)? [v]) /
 ([k] (juncture)? [g]) / ([p] (juncture)? [b]) /
 ([t] (juncture)? [d]) / ([fkpt] (juncture)? [jz]) /
 ([b] (juncture)? [j]) / ([s] (juncture)? [b]))
```

Pairs of consonants which are not permitted (with or without a syllable break).

```
NonjointCCC <- (([c] (juncture)? [d] (juncture)? [z]) /
 ([c] (juncture)? [v] (juncture)? [l]) /
 ([n] (juncture)? [d] (juncture)? [j]) /
 ([n] (juncture)? [d] (juncture)? [z]) /
 ([d] (juncture)? [c] (juncture)? [m]) /
 ([d] (juncture)? [c] (juncture)? [t]) /
 ([d] (juncture)? [t] (juncture)? [s]) /
 ([p] (juncture)? [d] (juncture)? [z]) /
([g] (juncture)? [t] (juncture)? [s]) /
 ([g] (juncture)? [z] (juncture)? [b]) /
([s] (juncture)? [v] (juncture)? [l]) /
 ([j] (juncture)? [d] (juncture)? [j]) /
([j] (juncture)? [t] (juncture)? [c]) /
([j] (juncture)? [t] (juncture)? [s]) /
([j] (juncture)? [v] (juncture)? [r]) /
([t] (juncture)? [v] (juncture)? [l]) /
 ([k] (juncture)? [d] (juncture)? [z]) /
([v] (juncture)? [t] (juncture)? [s]) /
 ([m] (juncture)? [z] (juncture)? [b]))
```

Triples of consonants forbidden, with or without a syllable break.

```
RepeatedVowel <- (([Aa] (juncture)? [a]) /
 ([Ee] (juncture)? [e]) / ([Oo] (juncture)? [o]) /
([Ii] juncture [i]) / ([Uu] juncture [u]))
```

Doubled vowel disyllables, notable because one of the vowels must be stressed and one unstressed. In the cases of **i** and **u** the juncture is not optional for this class, as the letter pairs admit a monosyllabic pronunciation.

```
RepeatedVocalic <- (([Mm] [m]) /
([Nn] [n]) / ([Ll] [l]) / ([Rr] [r]))

Syllabic <- [lmnr]
```

Syllabic consonants. Where a consonant is used syllabically, it must be doubled. Individual potentially syllabic consonants are used below to define pairs of consonants forbidden in syllable-final position.

```
Nonsyllabic <- (!(Syllabic) C1)

Badfinalpair <- (Nonsyllabic !('mr') !(RepeatedVocalic)
Syllabic !((V2 / [y] / RepeatedVocalic)))
```

A bad final pair is a nonsyllabic consonant followed by a syllabic consonant but not by **mr** or a syllabic pair, the whole not being followed by a vowel or syllabic pair, and the two consonants not separated by a juncture. This amounts to a nonsyllabic consonant followed by a syllabic consonant in syllable-final position. The reason it would be bad is that it could really only be pronounced as another syllable.

```
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)) !(juncture))
```

Permitted initial sequences of consonants in syllables. The general idea is that one has a sequence of one to three consonants, any adjacent pair in which is an initial pair, and not sharing a consonant with a syllabic consonant pair. The first version, which appears in syllables in predicates, may not be followed by **y**. Of course the initial consonants in a syllable are not followed by a juncture.

```
VowelSegment <- ((NextVowels !(RepeatedVocalic)) /
 !(C1 RepeatedVocalic) RepeatedVocalic)

VowelSegment2 <- (NextVowels / !(C1 RepeatedVocalic) RepeatedVocalic)
```

The vowel segment component of a syllable. The first version, which appears in borrowed predicates, is either the next vowel syllable chosen according to the priority scheme given above or a syllabic consonant pair, and if it is a vowel or vowel pair it cannot be followed by a syllabic pair. In the second version, found in names, the restriction on being followed by a syllabic pair does not apply. A syllabic pair will not be followed by an occurrence of the same consonant.

```
Syllable <- (((FirstConsonants)? !(RepeatedVowel)
!((&(Mono) V2 RepeatedVowel)) VowelSegment
 !(Badfinalpair) (FinalConsonant)? (FinalConsonant)?)
(juncture)?)
```

The general definition of a syllable, as found in a borrowed predicate. A syllable consists of an optional `FirstConsonants` followed by a vowel segment followed by zero one or two final consonants (defined below). There are side conditions: the vowel segment cannot be the first vowel in a repeated vowel pair, nor can it be a vowel pair followed by another occurrence of the second vowel in the pair, and if there are two final consonants, they cannot make up a `Badfinalpair` (because such a pair is impossible to pronounce except as another syllable: this rule is almost unique among my phonetic rules in not being found in some form in the original material–but also, it is nowhere violated except in names where we correct it by doubling the continuant).

The strictures on repeated vowels enforce the rule that the repeated vowel pairs which force stress cannot occur in borrowed predicates.

```
JunctureFix <- (((FirstConsonants)? V2 juncture &(InitialCC)
(!(C1) RepeatedVocalic)) /
 ((FirstConsonants)? VowelSegment C1 !(InitialCC) &(MaybeInitialCC V2)))
```

This class defines conditions on junctures which are excluded in certain contexts. The first clause describes a situation in which one has a syllable with one vowel ending in a juncture and followed by an initial consonant pair which is not the first two consonants of a consonant followed by a syllabic pair. In this situation, the juncture should be moved between the two consonants. In the second case we have a syllable followed by a consonant followed by an initial consonant pair broken by a juncture; in this case the juncture should be moved to the head of the consonant pair.

The rule `JunctureFix` is a technical device to prevent the parser from allowing borrowings which are illegal variants of complexes formed simply by moving syllable breaks. A vowel-final syllable may not be followed by an initial consonant pair in a borrowing– the juncture must be placed to separate the two consonants (unless the second consonant of the pair starts a syllabic consonant pair – the juncture `i-gllu` is permitted). A syllable in a borrowing may not end with a consonant followed by an initial consonant pair broken by a juncture. The reason for this is that one of these situations must occur in a variant of a complex formed by moving a syllable break in a way which fails to respect the boundaries of djifoa, and any such situation can be fixed by moving a juncture in a legal way. The point of this rule is to prevent complexes with illegally placed syllable breaks from parsing as borrowings. Explicit syllable breaks of the kinds excluded do occur in complexes, and such syllable breaks may likely occur in speech in borrowings with no harm done; this is basically a rule of orthography rather than phonetics.

Recently noted that the second sequence should only be forbidden when followed by a vowel; a four consonant sequence will not occur in a borrowing!

```
SyllableFinal1 <- ((FirstConsonants)? !(RepeatedVocalic)
 VowelSegment !(stress)  juncture? (&(Syllable) / &([y]) / !(Letter)))

SyllableFinal2 <- ((FirstConsonants)? !(RepeatedVocalic)
 VowelSegment (juncture)? (&([y]) / !(Letter)))

SyllableFinal2a <- ((FirstConsonants)? !(RepeatedVocalic)
 VowelSegment (juncture)? (&([y]) / !(Letter)))

SyllableFinal2b <- ((FirstConsonants)? !(RepeatedVocalic)
VowelSegment stress (&([y]) / !(Letter)))
```

The first two classes describe syllables which can be identified as (possibly) final in a borrowed predicate word because they are vowel-final, not explicitly stressed and followed by another syllable, `y` or a non-Letter, or definitely final because followed by `y` or a non-Letter.

The second two classes give the same descriptions modified for the case of a syllable (possibly) final in a borrowing affix. Syllables final in a borrowing affix are always followed by `y` and may be explicitly stressed: the classes given are not incorrect but could be simplified. Recall that a Letter is a letter or juncture.

```
StressedSyllable <- (((FirstConsonants)? !(RepeatedVowel)
!((&(Mono) V2 RepeatedVowel)) VowelSegment
!(Badfinalpair) (FinalConsonant)? (FinalConsonant)?) stress)
```

This is an explicitly stressed syllable appropriate for a borrowed predicate.

```
FinalConsonant <- (!RepeatedVocalic !(NonmedialCC) !(NonjointCCC) !(Syllable)
 C1 !((juncture V2)))
```

This is the class defining each of the two optional final consonants of a syllable in a borrowed predicate. They may not start a forbidden pair or triple of consonants. They may not start a syllable. They may not be followed by a juncture then a vowel. Both of these last two measures enforce the idea that syllables start as soon as possible. They will not start syllabic pairs.

```
Syllable2 <- (((FirstConsonants2)? (VowelSegment2 /
 [y]) !(Badfinalpair) (FinalConsonant2)? (FinalConsonant2)?)
 (juncture)?)

FinalConsonant2 <- (!RepeatedVocalic !(NonmedialCC) !(NonjointCCC)
 !(Syllable2)  C1 !((juncture V2)))
```

These classes define syllables in names and the final consonant class in names. The main difference is that the vowel segment may be **y** and various context restrictions on the vowel segments in borrowed predicates are lifted: the vowel segment may overlap a repeated vowel pair and may be followed by a syllabic pair. The rules governing the final consonant are the same, except that the kind of syllable it cannot start is the one appropriate to names.

The preceding block of rules defines the Loglan syllable, in two versions, one for borrowed predicate words and one for names, with some variations and subclasses specified.

The most general form of the Loglan syllable is the one which can appear in names. It consists of an optional permitted sequence of initial consonants, followed by a vowel segment or `y`, followed by one or two optional final consonants, which may not make up a pair consisting of a non-continuant followed by a continuant, followed by an optional juncture. The rules governing the final consonants are that a final consonant may not begin a forbidden sequence of two or three consonants from the lists above taken from NB3 (whether this continues into the next syllable or not, and ignoring junctures), it may not begin a syllable (junctures are placed as far to the left as possible if not explicitly given) and it may not be followed by an explicit juncture followed by a vowel (a syllable will always incorporate at least one of a preceding block of consonants).

In borrowed predicate words, the additional conditions are imposed that the vowel segment may not be replaced by `y` and the last vowel in the vowel segment may not start a repeated vowel sequence which forces a stress.

A class of stressed syllables appropriate in borrowings is given.

Classes of syllables are given which can be identified as (possibly) final in a borrowed predicate word because they are vowel-final, not explicitly stressed and followed by another syllable, `y` or a non-letter, or definitely final because followed by `y` or a non-letter. Syllables final in a borrowing affix are always followed by `y` and may be explicitly stressed: the classes given are not incorrect but could be simplified.

Note use of VowelSegment2 in Syllable2 so that names may contain vowels followed by syllabic consonants, ruled out now in predicates.

```
Name <- (([ ])* &(((uppercase / lowercase)
 ((!((C1 (stress)? !(Letter))) Lowercase))* C1 (stress)?
 !(Letter) (&end / comma / &(period) / &(Name) /
 &(CI)))) ((Syllable2)+
(&end / comma / &(period) / &(Name) / &(CI))))
```

This is the definition of the class of name words. A name word must resolve into syllables of class `Syllable2`. It must follow the capitalization conventions. Its final letter must be a consonant, which may be followed optionally by an explicit stress mark, which may further be followed by a `comma` (included in it). If it does not end with a `comma`, it must be followed by one of the following items not included in it: end of text, a `period`, another `Name` (noting that a name includes initial whitespace), or the cmapua **ci**.

```
CCSyllableB <- (((FirstConsonants)?
 RepeatedVocalic !(Badfinalpair)
 (FinalConsonant)? (FinalConsonant)?) (juncture)?)
```

This is the class of syllables in borrowings with syllabic pairs as their vowel segments.

```
BorrowingTail <- ((!(JunctureFix) !(CCSyllableB)
 StressedSyllable ((!(StressedSyllable) CCSyllableB))?
 !(StressedSyllable) SyllableFinal1) /
(!(CCSyllableB) !(JunctureFix) Syllable
 ((!(StressedSyllable) CCSyllableB))?
 !(StressedSyllable) SyllableFinal2))
```

A borrowing tail is a final segment of a borrowed predicate, consisting of a stressed syllable, not containing a syllabic pair, followed optionally by an unstressed syllable with a syllabic pair as its vowel segment, followed by a required unstressed final syllable of one of the kinds above.

This is implemented either as an explicitly stressed syllable followed optionally by an unstressed syllabic consonant pair syllable, followed by a `SyllableFinal1`, which is merely possibly final (vowel final, not explicitly stressed, followed by another syllable or a **y** or a non-Letter), or as a mere syllable followed by an optional syllabic pair syllable followed by a `SyllableFinal2` (which is known to be final because followed by **y** or a non-Letter). Recall that the class of Letters includes junctures.

The leading syllable in a borrowing tail will not be in the `JunctureFix` class.

```
PreBorrowing <- (((!(BorrowingTail) !(StressedSyllable) !(JunctureFix)
 !((CCSyllableB CCSyllableB)) Syllable))* !(CCSyllableB) BorrowingTail)
```

A pre-borrowing is a sequence of syllables none of which are explicitly stressed or satisfy the `JunctureFix` condition or are initial in a borrowing tail, followed by a borrowing tail. It also may not contain two successive syllables with syllabic consonants as vowel segments. This class is used in the algorithm for finding the left end of a borrowing, below.

```
HasCCPair <- ((((C1)? ((V2 (!stress juncture)?))+ !(Borrowing)stress?))?
 !((!(InitialCC) MaybeInitialCC)) C1 (juncture)? C1)
```

This class captures the condition on a pre-borrowing that it must contain a CC pair. A word of this class begins with an optional prefix which is either a consonant followed by one or more vowels or just one or more vowels. If the prefix is present and unstressed, what follows it cannot be a borrowing on its own [in a phonetic transcript, dropping a finally stressed initial CV^n might leave a residue which could be read as a predicate, by reading further than intended!]. The rest of the word must begin with two consonants, which may be separated by a juncture, but may not be separated by a juncture if they would otherwise form an initial pair.

```
CVCBreak <- (C1 V2 (juncture)? &(MaybeInitialCC)
 C1 (juncture)? &((PreComplex / ComplexTail)))
```

A word of this class begins with CVCC where the last C is initial in a pre-complex (see below) or complex tail, and the CC pair is initial (possibly broken by a juncture). Such words may not be predicates. The problem is that the initial CV could fall off as a little word.

```
CCVV <- ((&(BorrowingTail) C1 C1 (C1)? V2 stress !Mono V2) /
 (&(BorrowingTail) C1 C1 (C1)? V2
(juncture)? V2 (!(Letter) / ((juncture)? [y]))))
```

This is a word which would otherwise qualify as a borrowing of the form CCVV or CCCVV (these forms are excluded: banning them allows some slightly larger complexes to be formed without difficulties).

```
Borrowing <- (!(CVCBreak) !(CCVV) &(HasCCPair)
 !((V2 (juncture)? MaybeInitialCC V2)) !(CCSyllableB)
(((!(BorrowingTail) !(StressedSyllable)!(CCSyllableB CCSyllableB)
 !(JunctureFix) Syllable))* !(CCSyllableB) BorrowingTail))
```

A borrowed predicate cannot be of the classes `CVCBreak` or `CCVV`. It must be of class `HasCCPair`. It cannot begin VCCV where the CC is an initial pair (even if broken by a juncture). It cannot begin with a syllabic pair syllable. It then consists of a series of syllables which are not explicitly stressed, do not satisfy the `JunctureFix` condition, and are not initial in a borrowing tail, followed by a borrowing tail. [The condition that the borrowing tail not begin with a syllabic pair syllable appears to be redundant.]

We now give the formal definition of a borrowing. A borrowing is a pre-borrowing (look at the end of the rule). In addition, it either begins with a pair of consonants or has a (C)V^n (meaning an optional consonant followed by a sequence of vowels) followed by a pair of consonants which does not start a shorter borrowing itself. This rule both makes sure that there is a pair of adjacent consonants and prevents a cmapua from falling off the front of a borrowed predicate.

It does not begin with a CVCC-sequence where the CC is an initial consonant pair and peeling the CVC off would give a pre-complex (see below). Nor can it begin with VCCV where the CC is an initial pair. The first syllable of a borrowing does not contain a syllabic consonant. A borrowing may not contain two successive syllables with syllabic consonants.

There are no CCVV or CCCVV borrowings.

Nothing in this definition is new, except for technical details of placement of syllable breaks, which do not contradict anything in the sources (in any significant way: JCB did write some very odd junctures).

The CC found in HasCCPair cannot be a permissible initial broken by a juncture.

```
PreBorrowingAffix <- ((((!(StressedSyllable) !(SyllableFinal2a) !(JunctureFix)
 !((CCSyllableB CCSyllableB)) Syllable))+ SyllableFinal2a)
 (juncture)? [y] !stress juncture? (([ ,] ([ ])*))?)
```

This is a preliminary definition of borrowing affix; this resolves into syllables, not stressed, not in `JunctureFix`, followed by a possibly stressed final syllable, followed by **y** and possibly an explicit pause. We permit stress on the final syllable of a borrowing affix; note that the placement is not the same as in the borrowing it is derived from. It also may not contain two successive syllables with syllabic consonants. This condition makes sense, and imposing it on borrowings and borrowing affixes prevents a kind of ill-formed borrowing affix from being accepted which would otherwise be hard to detect.

```
BorrowingAffix <- (!(CVCBreak) !(CCVV) &(HasCCPair)
 !((V2 (juncture)? MaybeInitialCC V2)) !(CCSyllableB)
(((!(StressedSyllable) !(SyllableFinal2a)
 !(CCSyllableB CCSyllableB) !(JunctureFix) Syllable))+
SyllableFinal2a) (juncture)? [y] !stress juncture? (comma)?)
```

This class adds the restrictions that pick out the actual borrowing affixes from the wider class just defined. They are the same as the additional formal restrictions on borrowings.

A borrowing affix is exactly a borrowing followed by `y` (a ruling in Appendix H) but it differs from a borrowing in having its stress, if any, placed finally. These are not new conditions; they are found in our sources. Separate but very similarly motivated rules are needed because of the different treatment of the stress.

```
StressedBorrowingAffix <- (!(CVCBreak) !(CCVV) &(HasCCPair)
 !((V2 (juncture)? MaybeInitialCC V2)) !(CCSyllableB)
(((!(StressedSyllable) !(SyllableFinal2a)
 !(CCSyllableB CCSyllableB)!(JunctureFix) Syllable))*
 SyllableFinal2b) (juncture)? [y] !stress juncture? !([,]))
```

This is an explicitly stressed borrowing affix.

We now begin the definition of the class of complex predicates. These are peculiar in being built from djifoa ("affixes") rather than syllables. In defining possible syllable breaks in complexes, I followed the rule that a syllable in a complex predicate may not cross the boundary of a djifoa.

```
yhyphen <- ((juncture)? [y] !(stress) juncture? !([y]) &(letter))
```

This is a **y**-hyphen, used to fix sound conflicts at djifoa boundaries. It is unstressed, may be preceded and/or followed by actual juncture hyphens and must be followed by a letter after any juncture hyphen.

A y-hyphen is a buffering syllable which may be placed between djifoa to fix pronunciation problems. It is regarded as part of the preceding djifoa for parsing purposes. It is not stressed and will be followed (after an optional hyphen) by a letter other than `y`.

```
CV <- ((C1 V2 !(V2)) !(stress) juncture?)
```

This defines the final unstressed CV syllable of a five letter djifoa. It will not be followed by a vowel or stress marker. It may be followed by a juncture.

```
Cfinal <- ((C1 yhyphen) /
(!(NonmedialCC) !(NonjointCCC)
 C1 !(((juncture)? V2))))
```

This is the final consonant of a CVC djifoa. It is either a consonant followed by a **yhyphen** or a consonant not beginning a forbidden medial pair or triple of consonants and not followed by a vowel (even with an intervening juncture).

```
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])))
```

This is a class of phonetic hyphens used to fix sound conflicts at boundaries between djifoa. This will be an **r** not followed by another **r** or by a vowel (including across junctures), an **n** followed by an **r**, or an unstressed **y**; it will be followed by a letter, possibly with intervening juncture, which will not be a **y**.

The class `noyhyphen` excludes the **y** phonetic hyphens, and does not absorb a following juncture.

```
StressedSyllable2 <- (((FirstConsonants)?
VowelSegment !(Badfinalpair)
 (FinalConsonant)? (FinalConsonant)?) stress (yhyphen)?)
```

This is an explicitly stressed syllable of a kind that can occur in a complex predicate. Repeated vowels are not restricted.

```
CVVStressed <- (((C1 &(RepeatedVowel)
 V2 !(stress) (juncture)? !(RepeatedVowel)
 V2 (noyhyphen)?) (juncture)? (yhyphen)?) /
 (C1 !(BrokenMono) V2 !(stress) juncture
 V2 (noyhyphen)? stress (yhyphen)?) /
 (C1 !(Mono) V2 V2 (noyhyphen)? stress (yhyphen)?))

CVVStressed2 <-
 (C1 Mono (noyhyphen)? stress (yhyphen)?)
```

The first class is the class of all finally stressed two-syllable CVV dijifoa. The first sort, with a repeated vowel, may be qualified as possibly finally stressed. The class BrokenMono applies to optional monosyllables which are forced to be two syllables by an explicit juncture. One has CVV djifoa here with optional hyphen in the middle but in any case certain to be two syllables, and either with explicit stress at the end or repeated vowels of the sort which force a stress.

The second class is the explicitly stressed CVV monosyllable

```
CVV <- (!((C1 V2 stress V2 (hyphen)? stress))
 ((C1 !(BrokenMono) V2 (juncture)? !(RepeatedVowel)
 V2 (noyhyphen)?) (juncture)? (yhyphen)?))
```

This is a general CVV djifoa. It does not have two stressed syllables. It is not followed by another vowel the same as its final vowel.

```
CVVFinal1 <- (C1 !(BrokenMono) V2 stress V2 !stress juncture?)

CVVFinal2 <- (((C1 !(Mono) V2 V2) /
 (C1 !(BrokenMono) V2 juncture V2)) !(Letter))

CVVFinal5 <- (((C1 !(Mono) V2 V2) /
 (C1 !(BrokenMono) V2 juncture V2)) &(((juncture)? [y])))

CVVFinal3 <- (C1 Mono !(stress)  juncture?)

CVVFinal4 <- (C1 Mono !(Letter))
```

These are various forms of CVV djifoa which must be or can be final in a complex. The first one is a disyllable with an explicit stress on the first syllable. The second is either a mandatory disyllable or a disyllable formed by an explicit juncture, not followed by a letter or juncture. The third (oddly labelled 5) is the same as the previous but followed by **y** (with a possible intervening juncture). The fourth (oddly labelled 3) is an unstressed possible monosyllable (this one is a merely possible final syllable; the others must be final). The fifth (oddly labelled 4) is a possible monosyllable not followed by a letter or juncture.

```
CVC <- ((C1 V2 Cfinal) (juncture)?)

CVCStressed <- (C1 V2 !(NonmedialCC)
 !(NonjointCCC) C1 stress (yhyphen)?)
```

The first class is the class of CVC djifoa. The second is the class of stressed CVC djifoa, possibly followed by a yhyphen which would be included.

```
CCV <- (InitialCC !(RepeatedVowel) V2 (yhyphen)? (juncture)?)

CCVStressed <- (InitialCC !(RepeatedVowel)
 V2 stress (yhyphen)?)

CCVFinal1 <- (InitialCC !(RepeatedVowel) V2 !(stress)  juncture?)

CCVFinal2 <- (InitialCC V2 !(Letter))
```

These are forms of CCV djifoa. A CCV djifoa begins with an initial pair, is not followed by another copy of the same vowel, may be followed by and include a yhypen and/or a juncture. The second form is an explicitly stressed form satisfying the same conditions. The last two are possibly final forms: the first is simply unstressed, the second (which must be final) is not followed by a letter or juncture.

Laying to rest a silly remark made in older documents, a CCV djifoa may require a **y** hyphen, but only when standing before a borrowing djifoa.

```
CCVCVMedial <- (InitialCC V2 juncture?  C1 [y] !stress juncture? &(letter))

CCVCVMedialStressed <- (CCV stress C1 [y] !stress juncture? &(letter))

CCVCVFinal1 <- (InitialCC V2 stress CV)

CCVCVFinal2 <- (InitialCC V2 (juncture)? CV !(Letter))
```

These are the CCVCV five letter djifoa. The first two are the medial four-letter djifoa, with the final vowel replaced with **y**. These must be followed by a letter, possibly with an intervening juncture. The last two are five-letter final forms, one explicitly stressed and one not followed by a letter or juncture.

```
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?)

CVCCVFinal1b <- (C1 V2 !(NonmedialCC) C1 stress CV)

CVCCVFinal2 <- (C1 V2 ((juncture &(InitialCC)))?
 !(NonmedialCC) C1 (juncture)? CV !(Letter))
```

These are the CVCCV five letter djifoa. The first two are the medial forms. The optional juncture between the syllables can be placed before the two consonants if they are an initial pair, and in any case between. The last are final five-letter forms, either explicitly stressed (in the two possible ways) or not followed by a letter or juncture.

```
GenericFinal <- (CVVFinal3 / CVVFinal4 / CCVFinal1 / CCVFinal2)

GenericTerminalFinal <- (CVVFinal4 / CCVFinal2)
```

These are final or possibly final monosyllables. The second one includes only those which are not followed by a letter or juncture and so must be final.

```
Affix1 <- (CCVCVMedial / CVCCVMedial / CCV / CVV / CVC)
```

These are the non-borrowing djifoa.

```
Peelable <- (&(PreBorrowingAffix) !(CVVFinal1)
 !(CVVFinal5) Affix1 (!(Affix1) / &((&(PreBorrowingAffix)
 !(CVVFinal1) !(CVVFinal5) Affix1
!(PreBorrowingAffix) !(Affix1))) / Peelable))
```

The peelable djifoa at the beginning of a borrowing affix are a string of Affix1's each of which is initial in a pre-borrowing-affix, none of which are final in a borrowing affix or followed by an Affix1 final in a borrowing affix. Any peelable djifoa in this sense are actually fake: if we see no Affix1 or a peelable Affix1 we are at the start of a borrowing affix.

The details are horrible: the upshot is that if one has a Peelable initial Affix1 or no initial Affix1, the apparent borrowing affix which follows is not resolvable into djifoa. Something which looks like a borrowing affix, has a front Affix1 but does not have a peelable front Affix1 is in fact resolvable into Affix1's and not a borrowing affix at all. Same remarks for Peelable2 below.

I shall try to narrate this. A `Peelable` starts with an `Affix1` (one of the non-borrowing medial djifoa). It cannot be a `CVVFinal1` or a `CVVFinal5` (it will not by hypothesis be final in the borrowing affix in which it is initial at all; these are two syllable forms which could be the entire pre-borrowing affix minus the final **y**). It is initial in a `PreBorrowingAffix`. It is followed either (1) by something not an `Affix1` (so we can see immediately that the `PreBorrowingAffix` in which it is immersed does not resolve into djifoa) or (2) another `Affix1`, similarly starting a `PreBorrowingAffix` and not a `CVVFinal1` or `CVVFinal5` which is followed either by a non-`Affix1` or a non-`PreBorrowingAffix`, so once again we can see that the pre-borrowing affix in which our original djifoa was initial does not resolve into djifoa, or (3) an object of class `Peelable` (which is part of the `Peelable` string we read, unlike the objects in (1) and (2)), from which we can keep peeling initial affixes until we get behaviour (1) or (2) and see that the original pre-borrowing affix does not resolve into djifoa. In all cases, what we have is a pseudo-djifoa.

It is a very important point in the justification of `Peelable` that two successive `Affix1`'s which start pre-borrowing affixes will agree on where their respective pre-borrowing affixes end: because such an djifoa cannot include or be immediately followed by a **y** and a pre-borrowing affix ends with a **y** – as long as the first `Affix1` doesn't have two syllables with initial stress, so it can just be closed off with a **y** itself, which is why the two **CVV** classes are ruled out.

```
FiveLetterFinal <- (CCVCVFinal1 / CCVCVFinal2 /
CVCCVFinal1a / CVCCVFinal1b / CVCCVFinal2)

Peelable2 <- (&(PreBorrowing) !(CVVFinal1) !(CVVFinal2) !(CVVFinal5)
 !(FiveLetterFinal) Affix1 !(FiveLetterFinal) (!(Affix1) /
&((&(PreBorrowing) !(FiveLetterFinal) !(CVVFinal1) !(CVVFinal2) !(CVVFinal5)
 Affix1 !(PreBorrowing) !(FiveLetterFinal) !(Affix1))) / Peelable2))
```

As Peelable, but for borrowings proper.

The narrative here is the same as above under `Peelable`: the difference is that more final forms are specificially excluded. Of course, the `Affix1` found here will not be final in the pre-borrowing in which it is initial. There is a situation in which a djifoa can appear `Peelable2` when it is actually not in intent: this is if it is the stressed syllable in a two-djifoa predicate and the following djifoa can be read as starting a non-complex borrowing. It appears that this does no harm, however, since this will successfully be recognized as a `ComplexTail` and so will be read as a complex. An attempt to read it as a borrowing would give the same initial segment. It would be nice to fix this bug, though. I am not sure whether simply requiring that a `Peelable2` be unstressed would work, and I do believe that this bug actually cannot mess up the parse of any predicate.

```
Affix <- ((!(Peelable) !(Peelable2) Affix1) / BorrowingAffix)

Affix2 <- (!(StressedSyllable2) !(CVVStressed) Affix)
```

Since we can exclude fake djifoa read from the fronts of borrowing affixes or borrowings, we can read Affixes (djifoa). Affix2 excludes stressed non-borrowing djifoa.

```
ComplexTail <- ((Affix GenericTerminalFinal) /
 (!((!(Peelable) Affix1)) StressedBorrowingAffix GenericFinal) /
(CCVCVMedialStressed 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]))))
```

This catalogues the djifoa or pairs of djifoa which are guaranteed to terminate a complex, either because they explicitly end or because they are explicitly stressed. I don't believe I need to reproduce the catalogue of situations in English; I think they are readable from the rule.

I should check this – I discovered that the explicitly stressed CVV monosyllables had been overlooked, which broke some phonetic parses.

```
Primitive <- (CCVCVFinal1 / CCVCVFinal2 /
CVCCVFinal1a / CVCCVFinal1b / CVCCVFinal2)
```

This class is actually redundant (primitives turn out to be complexes), but it is nice to have the parser label the primitive five letter forms.

```
PreComplex <- (ComplexTail / ((!((CVCStressed /
CCVStressed / CVVStressed / ComplexTail /
 StressedSyllable2)) Affix) PreComplex))
```

This is a stream of djifoa none of which have a misplaced stress or start a ComplexTail, followed by a ComplexTail.

```
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 complex is a pre-complex satisfying initial conditions: one of these is that a CV or CVV cannot fall off the front and leave a predicate; the other forbids CVCC- initial predicates with more than six letters with the CC initial.

The issue about CV or CVV falling off the front is tricky in a phonetic transcript; one wants to make sure it doesn't make a predicate by reading further. If the CV or CVV being dropped is the stressed syllable, this provides no evidence against its being a predicate, unless the next syllable is a CVV.

```
Predicate <- ((&(caprule) ((Primitive / Complex / Borrowing)
 ((([ ])* Z AO (', ')? ([ ])* Predicate))?)) /
 (C1 V2 (V2)? ([ ])* Z AO (comma)? ([ ])* Predicate))
```

The full definition of a predicate word. Note that a borrowing is whatever is not a primitive or complex. Many though not all primitives and complexes would also parse as borrowings, and various tricky errors in complex predicate formation have manifested invisibly to ordinary use by parsing complexes as borrowings due to a bug. The option of building complex predicates from predicate words using `zao` is supported. It is worth noting that I do not require a pause after `zao` when it is followed by a vowel-initial borrowing, but the option of inserting a pause is supported.

A consonant initial cmapua can be affixed with ZAO to the front of a predicate.

It is important to note that there is no grouping with ZAO and there is no distinction between constructions with ZAO and the usual complexes, or indeed between either of these and mixed forms. Of course cmapua can be affixed to the front of a complex with zao which happen not to have corresponding djifoa.

```
__LWinit <- (([ ])* !(Predicate) &(caprule))
```

This is a marker for the beginning of a freestanding cmapua (structure word). Whitespace will be scanned over initially. The class excludes fake unit cmapua which are actually the beginning of predicates. The capitalization convention is enforced.

```
Oddvowel <- ((juncture)?
(((V2 (juncture)? V2 (juncture)?))* V2) (juncture)?)
```

This is a device for ensuring that CV cmapua are not mistaken for initial segments of CVV cmapua. Under normal circumstances, a cmapua unit will not be followed by an odd number of vowels without a pause (junctures are ignored). The reason for this is that a stream of vowels which is not in initial position in a block of letters will be a string of VV attitudinals (UI words). So a little word will not be followed by a specimen of class Oddvowel (a stream of vowels of odd length).

```
__LWbreak <- (!(Oddvowel) !((!((([ ])* Predicate))
 !((&(nonamemarkers) Name)) (A / ICI / ICA / IGE / I)))
 ((comma &((!((&(nonamemarkers) Name)) (V2 / A)))))?)
```

This class is used to mark the ends of cmapua and some other words. It enforces the condition that the word is not followed without a space by an odd number of vowels. Most of its business is to ensure that vowel initial words which are not predicates (which will be members of certain classes of logical and utterance connectives, which are listed, will not follow a word without being preceded by an explicit pause. Further, the class allows an explicit pause to follow the word (included in the word for parsing purposes) before any vowel-initial or A word (not all A words are vowel initial, but the non-vowel-initial ones (such as `noa`) must nonetheless be preceded by pauses). Checks have been added to ensure that pauses before vowel initial names are treated correctly.

```
CANCELPAUSE <- (comma (('y' comma) / (C UU __LWbreak)))

PAUSE <- ((!(CANCELPAUSE) comma
 !((A / ICI / ICA / IGE / I)) !((&(V2) Predicate)))
 !((([ ])* &(nonamemarkers) Name)))
```

PAUSE contains those pauses which are not mandated by other phonetic conditions. The role of this class is greatly reduced by the abandonment of PAUSE/GU equivalence, under which these were the pauses which could have semantic functions, but it still has some limited use.

CANCELPAUSE is a device for indicating that a pause was made in error. The uses envisaged for it had to do with PAUSE/GU equivalence but there may be situations where one wants to cancel a pause nonetheless, so I am leaving it here.

```
TAI0 <- (!Predicate (((V1 juncture?
 !Predicate !Name M a juncture?)/
(V1 juncture? !Predicate !Name F i juncture?)/
(C1 AI u?)/(C1 EI u?)/(C1 EO)/
(Z i juncture? V1 juncture? (M a)? juncture?))
(!Oddvowel/&(![ ] TAI0))))
```

Atomic letter words. The deprecated V-initial names for vowels are retained for the moment. V-`ma` and V-`fi` are the old upper- and lowercase vowel names. The new ones are `zi`-V-`ma` and `zi`-V. The name for `y`, `ziy` is supported though phonetically irregular. C1-`ai` and C1-`ei` name upper and lower case consonants. C1-`eo` names Greek letters.

I have added C-aiu and C-eiu letters. Haiu is X, Kaiu is Q, Vaiu is W (change to e for lower case). This supplies the need for names of these non-Loglan letters, which are common in mathematics.

The principal function of these words is as pronouns!

This appears in this odd location for phonetic reasons: the classes A and I need to exclude it. Its proper place in the development is below just before class DA0.

I installed a completely unprincipled fix to save acronyms with the legacy VCV letterals which also allows a large space of multiletter pronouns: a TAI0 followed by a string of VCV letterals is still a TAI0. This creates no problems but is utterly unprincipled!

Fix installed to allow use of VCV letterals in acronyms.

```
NOI <- (!(Predicate) N OI !(Oddvowel))

A0 <- (!(Predicate) ((([AEOUaeou] !([AEIOUaeiou])) /
 (H a)) (juncture)? !(Oddvowel)))

A <- (__LWinit !(TAI0) (((N u) &((u / (N o)))))?
((N o))? A0 (NOI)? !((([ ])+ PA)) (PANOPAUSES)?
((!(Predicate) G u))? !(Oddvowel))

A1 <- (A __LWbreak)
```

These three classes define the main series of logical connectives. A0 contains the atomic logical connectives, the words `a e o u` amd the interrogative connective `ha`.

The class A of compound logical connectives is quite complex. One may prefix a compound logical connective with `no`. A `no` or `u` initial connective may further be prefixed with `nu`. The core of an A connective is an instance of A0. It can then be suffixed with `noi` and further suffixed with a PA class tense/location/relation word. The word can further be closed with **gu**.

The class A is not closed up with `LWbreak` because it can serve as an initial component of other classes. Other rules in the grammar (such as `LWbreak` above) ensure that one must explicitly pause before any word or word component of class A.

A1 is inhabited by top level A words.

The class A needs to check that it is not the beginning of one of the words `Vma` or `Vfi`, legacy vowel letterals. We do encourage not using these words, using `ziV` and `ziVma` instead.

An A word cannot be followed by a PA word unless the word is closed with GU or an explicit comma pause.

```
ACI <- (A !(Predicate) C i __LWbreak)

AGE <- (A !(Predicate) G e __LWbreak)
```

These are two other series of logical connectives. The ACI connectives bind more tightly. The AGE connectives bind more loosely. See the grammar section for details. A modification to the grammar was required to ensure that an A connective followed by GE cannot be confused with AGE: the solution is that classes of predicates and arguments linked by A1 are forbidden to start with GE (which is deprecated as a matter of style already): this is a new grammatical rule though, and causes certain examples given (and deprecated) by JCB not to parse at all.

```
CA0 <- ((((N o))? !(Predicate)
((C a) / (C e) / (C o) / (C u) / (Z e) / (C i H a)) !(Oddvowel)) (NOI)?)

CA1 <- (!(Predicate) (((N u) &(((C u) /
 (N o)))))? ((N o))? CA0 !((([ ])+ PA))
 (PANOPAUSES)? ((!(Predicate) G u))? !(Oddvowel))

CA <- (__LWinit &(caprule) CA1 __LWbreak)

ZE2 <- (__LWinit (Z e) __LWbreak)
```

An auxiliary series of logical connectives (used to link predicates internally to metaphors and in the construction of some other classes). Their structure is closely analogous to that of the main series; no pauses are needed.

Includes NO prefix and NOI suffix in core CA connective CA0. I should remove the optional NO in CA1.

The rule given here allows more CA words than LIP seems to permit.

ZE2 signals a variant use of ZE to connect arguments. I am contemplating adding words to this class to give a different solution to the problem of representing sets and lists by enumeration.

```
I <- (__LWinit !(TAI0) i !((([ ])+ PA)) (PANOPAUSES)?
 ((!(Predicate) G u))? __LWbreak)

ICA <- (__LWinit !(Predicate) i ((!(Predicate) H a) / CA1) __LWbreak)

ICI <- (__LWinit i (CA1)? !(Predicate) C i __LWbreak)

IGE <- (__LWinit i (CA1)? !(Predicate) G e __LWbreak)
```

Sentence and utterance level connectives with different levels of precedence. See the final rules of the grammar for the details.

An explicit pause is required if an I class word not closed with GU is followed by a PA word and an IPA word is not intended: a bare space between an I word and a following PA word is not permitted.

The class I needs to check that it is not the beginning of a legacy vowel letteral `ima` or `ifi`.

```
KA0 <- (!(Predicate) (((K a) / (K e) / (K o) /
 (K u) / (K i H a)) !(Oddvowel)))

KOU <- (!(Predicate) (((K OU) / (M OI) / (R AU) / (S OA)) !(Oddvowel)))

KOU1 <- (((N u N o) / (N u) / (N o)) KOU)

KA <- (__LWinit &(caprule) (((((N u) &((K u))))? KA0) /
 ((KOU1 / KOU) K i)) (NOI)? __LWbreak)

KI <- (__LWinit (K i) (NOI)? __LWbreak)
```

The forethought connectives. The causal word classes KOU and KOU1, which are more naturally associated with the PA words, are needed for the causal series.

```
BadNIStress <-  ((C1 V2 V2? stress
(M a)? (M OA)? NI RA)/
(C1 V2 stress V2 (M a)? (M OA)? NI RA))

NI0 <- !BadNIStress (!(Predicate) (((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)) !(Oddvowel)))

SA <- !BadNIStress(!(Predicate) ((S a) / (S i) / (S u)) (NOI)? !(Oddvowel))

RA <- !BadNIStress(!(Predicate) (((R a) / (R i) / (R o)) !(Oddvowel)))

NI1 <- ((NI0 (!BadNIStress M a)? (!BadNIStress M OA NI0*)? !Oddvowel)
     (comma2 !(NI RA) &NI)?)

RA1 <- ((RA (!BadNIStress M a)? (!BadNIStress M OA NI0*)? !Oddvowel)
      (comma2 !(NI RA) &NI)?)

IE1 <- (__LWinit IE __LWbreak)

NI2 <- ((((SA)? ((NI1)+ / RA1)) / SA)
 (NOI)? ((CA0 (((SA)? ((NI1)+ / RA1)) / SA) (NOI)?))*)
```

```
NI <- (__LWinit (IE1)? NI2 ((&((M UE)) Acronym
 (comma / &end / &(period)) !((C u))))? ((C u))? !(Oddvowel))

mex <- (__LWinit NI __LWbreak)
```

SA is a class of quantifiers with special functions as prefixes (which can be negated with **-noi**).

RA is a subset of the quantifier words which do double duty as suffixes creating predicates from quantity words.

guarded from being initial in a predicate, as many word component forms are.

The rule NI1 supports notation for powers of ten (though it also allows other odd things). An atomic quantifier word (in intention a numeral but the grammar does not require this) can be followed by an optional `ma`, then an optional `moa` which may have a numeral attached. The `ma` multiplies by 100; `moa` multiplies by 1000; `moa` followed by n multiplies by 1000^n (so `nomoato` is one million). This is new. The shape of `moa` was changed to avert confusion with the pronoun `mo`.

Notice that this change ensures that the use of `ma` to multiply by one hundred and the use of `ma` to capitalize a letter are distinguishable as what occurs just before them will make it clear what is intended. `ma` and `moa` are not members of NI0.

The rule NI2 describes quantifier cores.

IE1 is the interrogative "which", also used in argument constructions.

NI is the class of mathematical expressions. mex is a freestanding word class, NI can be a component of other classes. There is an optional initial `ie` (which?); this is followed by a quantifier core; one may optionally append an acronym followed by an explicit pause as a dimension, and there is a final option of appending `cu`.

As a "word" class NI is unusual in freely permitting spaces and comma marked pauses in the interior of its instances.

It is worth noting that SARA forms are not predicates but NI words. L1 **sara** is a predicate: for us this is **sarara**.

The NI class has been fixed so that it does not eat a following numerical predicate even though it may contain spaces or comma pauses: **ne, tori** does work. The fix also has the effect that numerical predicates do not contain spaces or comma pauses.

BadNIStress is a device for (almost) forcing penultimate stress on numerical predicates. There is freedom in where to place the stress if the last syllable(s) are **ma**, **moa** – it could be on these syllables or the last one before them.

```
CI <- (__LWinit (C i) __LWbreak)
```

a sort of verbal hyphen used in several constructions.

```
Acronym <- (([ ])* &(caprule) ((M UE) / TAI0 /
 ([Zz] V2 !(V2))) ((NI1 / TAI0 /
 ([Zz] V2 (!(V2) / ([Zz] &(V2))))))+)
```

There is now a single acronym class. An acronym is a sequence of letter names (possibly abbreviated in the case of vowels, and number names, beginning either with **mue** or a letter (possibly abbreviated) and having more than one component (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 (dimensions must begin with **mue**) and to form acronymic names (no longer acronymic predicates).

```
TAI <- (__LWinit (TAI0 / ((G AO) !(badspaces) !(V2) ([ ])*
 (Name / Predicate / (C1 V2 V2 (!(Oddvowel) /
&(TAI0))) / (C1 V2 (!(Oddvowel) / &(TAI0)))))) __LWbreak)
```

This is the full class of letterals, including John's proposal that allows construction of letters using `gao` followed by an arbitrary word (name, predicate or consonant-initial unit cmapua).

```
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)) !(Oddvowel))
```

Atomic pronouns of various kinds. They appear in this section because letterals are also pronouns.

```
DA1 <- (!(Predicate) ((TAI0 / DA0) ((C i !([ ]) NI0))? !(Oddvowel)))

DA <- (__LWinit DA1 __LWbreak)
```

Pronouns. These can be suffixed with mathematical expressions using `ci`. Restriction to single digit indices imposed to ensure that one does not need to pause to recognize the end of such a word. Please note that letteral pronouns are **single letters**, possibly with a numerical subscript. There are no multi-letter variables and thus there is no need to pause between letteral pronouns which appear in sequence as arguments in a sentence. It does appear as if this is what JCB assumed in NB3, though it is not what LIP does. It is far more important to treat sequences of pronouns sensibly than to support use of multi-letter variables. This is a change from LIP but not I think from the actual intentions of the founder.

There was danger of a need to pause after an acronym to avoid it absorbing a letteral: it is actually now always required that one pause after an acronym.

Here begins the definition of tense/location/relation operators. The component classes KOU1 and KOU were referenced early because of their role in defining forethought causal connectives.

```
PA0 <- (!(Predicate) (((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 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) / (M OU) / (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) !(Oddvowel)))

PANOPAUSES <- (!(Predicate)
 (((!(PA0) NI))? ((KOU1 / PA0))+
((CA0 ((KOU1 / PA0))+))* (ZI)? !(Oddvowel)))

PA <- (!(Predicate) (((!(PA0) NI))?
 (((KOU1 / PA0) ((comma2 &((((CA0 (comma2)?))? PA))))?))+
 (((CA0 (comma2)?) (((KOU1 / PA0)
 ((comma2 &((((CA0 (comma2)?))? PA))))?))+))* (ZI)? !(Oddvowel)))
```

I am skeptical of **pau**, ago, but implemented it because it appears in the Visit.

A complex PA word begins with an optional numeral or quantifier, followed by a core PA0 or KOU1 word, which may be linked by CA0 connectives to further possibly negated PA0/KOU1 words, then may optionally close with a ZI qualifier or be followed by another PA word. Pauses are permitted in medial positions (not immediately before ZI).

The version without pauses is used in APA and IPA words

```
PA2 <- ((__LWinit PA __LWbreak) (freemod)?)
```

A PA word as a preposition, so to speak.

```
GA <- (__LWinit (G a) __LWbreak)

PA1 <- (((PA2 / GA) __LWbreak) (freemod)?)
```

A PA word as a predicate marker (a tense, so to speak). GA is the purely grammatical predicate marker with no semantic freight.

```
ZI <- (!(Predicate) ((Z i) / (Z a) / (Z u)))
```

qualifying suffixes for tense and location words.
guarded against being initial in a predicate.

```
LE <- (__LWinit ((L EA) / (L EU) / (L OE) /
 (L EE) / (L AA) / (L e) / (L o) /
((L a) !(badspaces))) ((DA1 / TAIO))?
 !((([ ])+ PA)) (PA)? __LWbreak)
```

The commonest class of argument constructors. These can be adorned with an optional pronoun followed by an optional tense/location/relation operator.

```
LEFORPO <- (__LWinit ((L e) / (L o) / NI2) __LWbreak)
```

These are the possible LE components of a LEPO phrase (an abstraction descriptor). The LE and PO components are separate words, and can be separated by commas or not as one pleases. Note that this includes some NI words.

```
LIO <- (__LWinit (L IO) __LWbreak)
```

The numeral article.

```
LAU <- (__LWinit ((L AU) / (L OU)) __LWbreak)

LUA <- (__LWinit ((L UA) / (L UO)) __LWbreak)
```

Left and right boundary markers for explicit sets and lists above (these are currently not documented, and their grammar needs serious repair).

Below, the beginning of quotation constructions.

```
LI1 <- (L i)

LU1 <- (L u)

Quotemod <- (((Z a) / (Z i)) !(Oddvowel))
```

affixes (in the true sense) which can qualify the sense of a quotation (as text or sound). At least one member of *la Keugru* strongly deprecates this device.

```
LI <- ((__LWinit LI1 !(V2)
 (Quotemod)? ((([,])? ([ ])+))? utterance0
 (', ')? __LWinit LU1 __LWbreak) /
 (__LWinit LI1 !(V2) (Quotemod)? comma name
 (comma)? __LWinit LU1 __LWbreak))
```

Above is the construction for quoting a Loglan utterance, `li` utterance `lu`. The convention of enclosing the quoted text in explicit pauses is allowed but not required. It is also possible to quote a name with this construction (a comma before the name is mandatory in this case).

```
stringnospaces <- (([,])? (([ ])+ ((!([ ,]) !(period) .))+)
 ((([,])? ([ ])+ &(letter)) / &(period) / &end))

stringnospacesclosed <- (([,])? (([ ])+
 ((!([ ,]) !(period) .))+) (([,] ([ ])+) / &(period) / &end))

stringnospacesclosedblock <- ((stringnospaces
 ((!(([y] stringnospacesclosed)) [y] stringnospaces))*
 ([y] stringnospacesclosed)) / stringnospacesclosed)
```

a block of text beginning with whitespace or an explicit pause and ending with whitespace, an explicit pause, 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 further classes build comma-closed alien text or sequences of alien text blocks separated by **y** with the final one comma closed.

```
LAO1 <- (L AO)

LAO <- (([ ])* (LAO1 stringnospaces (([y] stringnospaces))*))
```

JCB advertises this as the Linnaean biological name construction, but it is actually a perfectly general device for constructing foreign names in general (Steve Rice noted this in L3). A name is `lao` followed by one or more stringnospaces blocks. If there is more than one block, they must be separated by `y` bounded with pauses. JCB does not require writing the `y` but for the moment I do. One must pause at the end of a LAO construction though one need not write a comma.

I propose that if one wants to construct names by look rather than sound, one should use `lao`.

```
LIE1 <- (L IE)

CII1 <- ((C II) / [y])

LIE <- (([ ])* LIE1 ((!([ ]) NI0))?
 (Quotemod)? stringnospaces
((CII1 ((!([ ]) NI0))? stringnospaces))*)
```

This is my strong quotation construction, a new proposal, quite different from and intended to completely replace the L1 strong quotation proposal. The format is very similar to LAO (accidentally, I was not aware of the latest LAO specification). The format is `lie` (quoted text), with quoted text broken by `cii` where breaks occur. Quoted text may not contain commas (commas like whitespace to be replaced by `cii` (`y` also permitted)) or terminal punctuation. The initial `lie` may be qualified with Quotemod and/or with a numeral to indicate nested quotation. `cii` may also be numeral qualified to indicate level of nesting of quotation. Essay promised with all details.

```
LW <- (&(caprule) (((!(Predicate) V2 V2))+ /
((!(Predicate) (V2)? ((!(Predicate) LWunit))+) / V2)))

LIU0 <- ((L IU) / (N IU))

LIU1 <- (__LWinit ((LIU0 !(badspaces) !(V2)
 (Quotemod)? ((([,])? ([ ])+))?
 (Name / (Predicate (comma)?) /
(CCV (comma)?) / (LW (([,] ([ ])+ !([,])) / &(period) /
 &end / &((([ ])* Predicate)))))) / (L II (Quotemod)? TAI __LWbreak)))
```

This is the single word quotation construction. Oddly, this is the only place where the rule LW implementing the NB3 definition of compound cmapua is used, and so it appears here. A single name, predicate, or compound cmapua word may be quoted. `liu` (compound cmapua) must be comma terminated except when followed by a predicate or terminal punctuation. LIP appears to quote only actual compound cmapua (which is the basis of my claim that LIP makes no use of the compound cmapua phonetic algorithm anywhere). Mine will quote any phonetically possible cmapua. One can use `niu` to indicate that the quoted word is not actually a Loglan word.

CCV djifoa may be quoted with **liu**.

`lii` (name of a letter) gives an actual name for a letter (letterals being pronouns when unadorned).

It is wise to guard **liu** quotations with a comma whether this is formally required or not. Note that **liu** is a name marker, so text without explicit pauses leading up to a name word may lead to parse problems.

```
SUE <- (__LWinit ((S UE) / (S AO)) stringnospaces)
```

This handles two quite different functions with the same grammar (not quotation constructions but similar in handling non Loglan text). `sao` (foreign word) constructs a predicate synonymous with the foreign word. `sue` (transcribed sound) constructs an onomatopoeic predicate meaning to make the given sound.

```
CUI <- (__LWinit (C UI) __LWbreak)
```

left marker for metaphor grouping with connectives

```
GA2 <- (__LWinit (G a) __LWbreak)
```

The specific use of **ga** to mark the first argument in a gasent

```
GE <- (__LWinit (G e) __LWbreak)
```

general purpose left marker, several uses

```
GEU <- (__LWinit ((C UE) / (G EU)) __LWbreak)
```

The right marker for metaphor grouping. The L1 form is still supported.

```
GI <- (__LWinit ((G i) / (G OI)) __LWbreak)
```

Marker at the end of fronted argument and modifier lists (used for building object-first sentences; also useful for putting something before the predicate in an imperative).

```
GO <- (__LWinit (G o) __LWbreak)
```

Marker used for putting modifiers in metaphors after the modified.

```
GU <- (__LWinit (G u) __LWbreak)

GUI <- (__LWinit (G UI) __LWbreak)

GUO <- (__LWinit (G UO) __LWbreak)

GUOA <- (__LWinit (G UO  a) __LWbreak)

GUOE <- (__LWinit (G UO  e) __LWbreak)

GUOI <- (__LWinit (G UO  i) __LWbreak)

GUOO <- (__LWinit (G UO  o) __LWbreak)

GUOU <- (__LWinit (G UO  u) __LWbreak)
```

```
GUU <- (__LWinit (G UU) __LWbreak)

GUE <- (__LWinit (G UE) __LWbreak)
```

The word **gu** is the general purpose right closure. The other words close specific constructions. **gui** closes subordinate clauses. **guo** closes abstraction predicates and descriptions. **guu** closes termsets (lists of arguments and modifiers after predicates). **gue** closes the tightly bound argument lists built with **je** and **jue**.

```
JE <- (__LWinit (J e) __LWbreak)

JUE <- (__LWinit (J UE) __LWbreak)
```

The word **je** is used to tightly attach an argument or modifier to a predicate; **jue** is used to attach second and further arguments and modifiers in a list similarly tightly.

```
JI <- (__LWinit ((J IE) / (J AE) / (P e) /
(J i) / (J a) / (N u J i)) __LWbreak)

JIO <- (__LWinit ((J IO) / (J AO)) __LWbreak)
```

Markers for subordinate clauses.

```
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) / (L AE) / (L UE)) __LWbreak)

DIO2 <- (__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)) __LWbreak)
```

Case tags, and the words **lae** and **lue** of indirection which have similar grammar. Class DIO2 is not used yet, but I do have ideas about situations where identifying true case tags as a class would be necessary.

```
ME <- (__LWinit ((M EA) / (M e)) __LWbreak)
```

me converts arguments to predicates. I believe **mea** is the result of a misunderstanding by JCB, and **me** properly understood does its work as well.

```
NUO <- (((N UO) / (F UO) / (J UO) / (N u) / (F u) / (J u)) !(Oddvowel))

NU <- (__LWinit ((NUO !((([ ])+ (NIO / RA)))
((NIO / RA))? (freemod)?))+ __LWbreak)
```

Conversion operators, identifying arguments or changing the order of arguments in predicates.

```
PO1 <- (__LWinit ((P o) / (P u) / (Z o)) !(Oddvowel))

PO1A <- (__LWinit ((P OI  a) / (P UI  a) / (Z OI  a)) !(Oddvowel))

PO1E <- (__LWinit ((P OI  e) / (P UI  e) / (Z OI  e)) !(Oddvowel))

PO1I <- (__LWinit ((P OI  i) / (P UI  i) / (Z OI  i)) !(Oddvowel))

PO1O <- (__LWinit ((P OI  o) / (P UI  o) / (Z OI  o)) !(Oddvowel))

PO1U <- (__LWinit ((P OI  u) / (P UI  u) / (Z OI  u)) !(Oddvowel))

POSHORT1 <- (__LWinit ((P OI) / (P UI) / (Z OI)) !(Oddvowel))

PO <- (__LWinit PO1 __LWbreak)


POA <- (__LWinit PO1A __LWbreak)

POE <- (__LWinit PO1E __LWbreak)

POI <- (__LWinit PO1E __LWbreak)

POO <- (__LWinit PO1O __LWbreak)

POU <- (__LWinit PO1U __LWbreak)
```

```
POSHORT <- (__LWinit POSHORT1 __LWbreak)
```

short-scope and long-scope abstraction forming cmapua.

```
DIE <- (__LWinit ((D IE) / (F IE) / (K AE) / (N UE) / (R IE)) __LWbreak)
```

Indicators of attitude toward the person addressed (social register).

```
HOI <- (__LWinit ((H OI) / (L OI) /
 (L OA) / (S IA) / (S IE) / (S IU)) __LWbreak)
```

The vocative marker and other words which can open a vocative.

```
JO <- (__LWinit ((NIO / RA))? (J o) __LWbreak)
```

The right marker for the scare quote construction. The numerical suffix indicates its scope.

```
KIE <- (__LWinit (K IE) __LWbreak)

KIU <- (__LWinit (K IU) __LWbreak)
```

the verbal parentheses used to insert utterances as parenthetical free modifiers.

```
SOI <- (__LWinit (S OI) __LWbreak)
```

The constructor for spoken emoticons.

```
UI0 <- ((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)) !(Oddvowel))

NOUI <- ((__LWinit (N o) ([ ])* !(Predicate)
 UI0 __LWbreak) / (__LWinit UI0 NOI __LWbreak))

UI1 <- (__LWinit (UI0 / (NI F i)) __LWbreak)
```

Attitudinal constructions. The atomic cmapua of class UI can be negated, either with a **no** prefix or a **noi** suffix (the latter I believe is a small proposal). In additional the numerical discursives with **fi** are attitudinals.

Note that certain UI words are also HOI words.

```
HUE <- (__LWinit (H UE) __LWbreak)
```

The inverse vocative marker.

```
NO1 <- (__LWinit !(KOU1) !(NOUI) (N o)
!((__LWinit KOU)) !((([ ])* (JIO / JI))) __LWbreak)
```

This is the class of occurrences of **no** as a separate word, not absorbed by a following attitudinal, **kou** word, or subordinate clause marker.

```
AcronymicName <- (Acronym (&end / ',' / &(period) / &(Name) / &(CI)))

DJAN <- (Name / AcronymicName)
```

The full construction of name words. Acronyms are treated as a separate class of names. `DJAN` contains all name words. `Name` contains the consonant final name words from the phonetics section.

```
BI <- (__LWinit ((N u))? ((B IA) / (B IE) /
 (C IE) / (C IO) / (B IA) / (B [i])) __LWbreak)
```

The special class of predicates of which **bi**, the “identity” predicate, is the best known example. Converses formed with **nu** of these predicates (all of them are binary) are allowed (a proposal).

```
LWPREDA <- (((H e) / (D UA) / (D UI) / (B UA) / (B UI)) !(Oddvowel))
```

predicates realized as cmapua: the question predicate and various “predicate pronouns”.

```
PREDA <- (([ ])* &(caprule) (Predicate / LWPREDA /
 (!([ ]) NI RA)) !((!((&(nonamemarkers) Name))
 (A / ICI / ICA / IGE / I))) ((',' ([ ])+
&((!((&(nonamemarkers) Name)) (V2 / A)))))? (freemod)?)
```

Words which are understood as non-identity predicates. These are designed to absorb mandatory pauses which follow them before vowel-initial words, and to recognize when such mandatory pauses have been omitted.

```
guo <- ((PAUSE)? (GUO / 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)?)

gui <- ((PAUSE)? (GUI / GU) (freemod)?)

gue <- ((PAUSE)? (GUE / GU) (freemod)?)

guu <- ((PAUSE)? (GUU / GU) (freemod)?)
```

These classes realize the various special right closure markers. Each one can be expressed in its literal form, as **gu** or as a pause. Any pauses next to it are absorbed into it.

```
lua <- LUA
```

closures for explicit set and list constructions. Not really supported.

```
geu <- GEU
```

right closure in metaphors

```
gap <- ((PAUSE)? GU (freemod)?)

gap2 <- gap

guu1 <- gap
```

The grammar implementation of the general **gu** right closure: **gu** (absorbing pauses). pause/GU equivalence has been abandoned.

```
juelink <- (JUE (freemod)? term)

links1 <- (juelink (((freemod)? juelink))* (gue)?)

links <- ((links1 / (KA (freemod)? links (freemod)?
 KI (freemod)? links1)) (((freemod)? A1 (freemod)? links1))*)

jelink <- (JE (freemod)? term)

linkargs1 <- (jelink (freemod)? (links)? (gue)?)

linkargs <- ((linkargs1 / (KA (freemod)? linkargs (freemod)?
 KI (freemod)? linkargs1)) (((freemod)? A1 (freemod)? linkargs1))*)
```

Tightly bound lists of arguments and modifiers. The ability to include sentence modifiers in these lists is a proposal. In any such list, the argument directly after the predicate is linked with **je**; second and subsequent items in these lists are linked with **jue**. Such lists (and individual items linked with **je** or **jue**) can be linked with forethought and afterthought logical connectives. They can be right closed using **gue**.

```
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 (com
 abstractpred / (ME (freemod)? argument (gap2)?) / PREDA) (freemod)?)
```

Predicate units which are in a sense “atomic”. Some of these are quite complex structures which are in effect parenthesized. Each item in this rule is described in the list in the section on the basic building blocks of predicates in the reference grammar.

The PO sentence and PO uttAx forms have been moved to this category from a much different position in the trial.85 grammar, which allowed such forms a very limited role in the grammar. They have been moved into a separate class `abstractpred` to allow use of many bracketing forms, solving the closure problem (I hope).

```
predunit2 <- (((NO1 (freemod)?))* predunit1)
```

A `predunit2` is a possibly multiply negated predunit1. `NO2` captures occurrences of the word **no** not captured by `predunit1` instances.

```
NO2 <- (!(predunit2) NO1)

predunit3 <- ((predunit2 (freemod)? linkargs) / predunit2)

predunit <- (((POSHORT (freemod)?))? predunit3)
```

A `predunit3` is obtained from a `predunit2` by attaching JE/JUE linked arguments. A `predunit` is then either a `predunit2` or a `predunit2` preceded by a short-scope abstraction operator. These are the sort of predicates which can occur as units in serial names. Notice that these include no metaphorical modifications or logical connections, except possibly internally to parenthesized items in class `predunit1`.

```
kekpredunit <- (((NO1 (freemod)?))*
KA (freemod)? predicate (freemod)? KI (freemod)? predicate)
```

A `kekpredunit` is a possibly multiply negated forethought logically connected general predicate.

```
despredA <- ((predunit /
 kekpredunit) (((freemod)? CI (freemod)? (predunit / kekpredunit)))*)
```

A `despredA` is either a predunit or a kekpredunit or a chain of these linked with `ci`, the most tightly binding form of metaphorical modification.

```
despredB <- ((!(PREDA) CUI (freemod)?
 despredC (freemod)? CA (freemod)? despredB) / despredA)

despredC <- (despredB (((freemod)? despredB))*)
```

`despredB` and `despredC` interlock. A `despredA` is an item of either class. A `despredC` is just a chain of one or more `despredB`'s. A `despredB`, if it is not simply a `despredA`, begins with **cui**, followed by a `despredC` (that is, by one or more `despredB`'s), followed by a tt CA connective, followed by a final `despredB`. The purpose here is to enable leftward extension of the scope of a `CA` connective to an occurrence of **cui**. The simple juxtapositions here are left-grouped metaphorical modifications.

```
despredD <- (despredB (((freemod)? CA (freemod)? despredB))*)

despredE <- (despredD (((freemod)? despredD))*)
```

`despredD` allows afterthought logical connection of `despredB` class items with `CA` logical connectives.

`despredE` allows left-grouped metaphorical modification of a series of `despredD`'s. Right grouping is achieved by using the `ge`...`geu` construction to pack the right group as a single `predunit1`, the last `despredD` item in a `despredE`.

```
descpred <- ((despredE (freemod)? GO (freemod)? descpred) / despredE)
```

A `descpred` is either a `despredE`, or a structure consisting of a `despredE` followed by `go` followed by a `descpred`, having the effect of causing the first component to be metaphorically modified by the second component.

A `descpred` is the sort of predicate which can occur after **le** in a description. Notice that except between **cui** and a following `CA` connective, pauses are not allowed in juxtaposed series of predicates which modify one another in these classes. This has a role in pause/GU equivalence.

```
senpred1 <- (predunit (((freemod)? CI (freemod)? predunit))*)

senpred2 <- (senpred1 / (CUI (freemod)?
 despredC (freemod)? CA (freemod)? despredB))

senpred3 <- (senpred2 (((freemod)? CA (freemod)? despredB))*)

senpred4 <- (senpred3 (((freemod)? despredD))*)

sentpred <- ((senpred4 (freemod)? GO (freemod)? barepred) / senpred4)
```

These classes of predicates, which occur as main predicates of sentences rather than in descriptions, are built in exactly analogous ways to the classes of predicates which occur in descriptions, except that freemods (and so pauses) are allowed between successive items in left-grouped chains of metaphorical modifiers (at the top level; the individual items are description predicates and so have restricted internal pauses; but this seems a natural enough pattern of articulation). Notice though that `senpred2`, and `kekpredunits` will not occur as first items in chains of modifiers (they can occur later because second and later items in these chains are taken from the description predicate classes. The `barepred` class is a very general class of predicates with the possibility of attached arguments, defined below.

```
mod1 <- ((PA2 argument (gap2)?) /
(PA2 !(barepred) (guu1)?))

kekmod <- (((NO1 (freemod)?))* (KA (freemod)?
 modifier (freemod)? KI (freemod)? mod))

mod <- (mod1 / (((NO1 (freemod)?))* mod1) / kekmod)

modifier <- ((mod / kekmod) ((A1 (freemod)? mod))*)
```

This rule describes the sentence modifiers (tense/location/relative modifiers). The basic building blocks are a `PA2` operator followed by an argument, possibly closed with a `gap`, or a `PA2` operator by itself, closed with `guu1` (this cannot be closed with **gu** to avoid confuson with APA or IPA words closed with `gu`, which is part of a proposal).

A `kekmod` is a forethought connected modifier structure.

A `mod` is suitable to be a final item in a `kekmod`: it is either a basic unit or a negated basic unit or a kekmod.

A `modifier` is either a `mod` or an afterthought logically connected series of `mod`'s.

```
maybebreak <- (V1 (stress)? ' ' !((([ ])* V1)))

realbreak <- (!(maybebreak) letter (stress)? ((([,])? ' ') / period / &end))

consonantbreak <- (C1 (stress)? ((([,])? ' ') / period / &end))

badspaces <- (!(([,] ' ')) ((!((maybebreak / realbreak)) .))*
maybebreak ((!(realbreak) .))* consonantbreak)

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 DJAN)) Letter))+ !(Letter))
```

`badspaces` is an evil hack, used to detect failure to have a mandatory pause (comma marked, after a consonant or before a vowel) before the end of the first name after a name marker word which does not actually stand right before a name.

`namemarker` is the small class of words after which a name word can occur without a pause.

`nonamemarkers` is the class of name words not containing a name marker such that the part of the word after the name marker is a well-formed name. NOTE: this class should probably be defined in terms of `Name` rather than in terms of `DJAN`.

```
name <- ((DJAN (((([ ])* (freemod)? CI DJAN) /
 ((([ ])* (freemod)? CI !(badspaces) (freemod)? predunit)
 !((&(nonamemarkers) !(AcronymicName) DJAN))) /
 (([ ])* (freemod)? CI (',' ([ ])+) DJAN) /
(&(nonamemarkers) !(AcronymicName) DJAN)))*) (freemod)?)
```

This is the class of serial names. The first unit in a serial name will be a name word. Other units will be predunits marked with **ci**, name words marked with **ci**, or unmarked name words containing no false name markers and not preceded in the serial name by a predunit.

Pauses after name words in the interior of serial names must occur, but do not have to be marked with commas.

After CI without a pause, a wellformed name will always be read as such. After CI marked with a comma, a name-final predunit will be recognized even without spaces.

```
LAO <- ((L a) !(badspaces))

LANAME <- (([ ])* LAO name (gap2)?)

LANAME2 <- (([ ])* LAO (',' ([ ])+) name (gap2)?)
```

This is the basic name construction with the article **la**. A comma pause is permitted but not required between **la** and the name. But if a comma is present non-name words will be sought first by the parser. Phonetically there is the assumption that if a comma is not written one does not pause.

```
HOIO <- (((H OI) / (L OI) / (L OA) / (S IA) / (S IE) / (S IU)) !(badspaces))

voc <- ((([ ])* HOIO name (gap2)?) /
 (HOI !(badspaces) (freemod)? descpred (((((comma)? CI (comma)?) /
 (comma &(nonamemarkers) !(AcronymicName))) name))? (gap2)?) /
 (HOI !(badspaces) (freemod)? argument (gap2)?) /
 (([ ])* HOIO (',' ([ ])+) name (gap2)?) / (H OI stringnospacesclosedblock))
```

This is the general construction of a vocative (a species of free modifier). The first case, **hoi** followed by a serial name, is similar to `LANAME`.

If HOI is not followed by an explicit pause, a name will be read aggressively. If it is followed by a pause, it will try other readings first.

Only **hoi** may be followed by a foreign name (alien text): the other words usable as vocative markers may not be followed by alien text.

```
descriptn <- (!(LANAME) ((LE (freemod)? descpred) /
 (LE (freemod)? mex (freemod)? descpred) /
 (LE (freemod)? arg1 descpred) /
 (LE (freemod)? mex (freemod)? arg1a) /
(GE (freemod)? mex (freemod)? descpred)))
```

This is a basic class of descriptive arguments.

```
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 (originally two cases of the following class but pulled out on its own because more bracketing forms have been added to solve the GUO GUO problem).

```
arg1 <- (abstractn / (LIO (freemod)? descpred (gap2)?) /
 (LIO (freemod)? term (gap2)?) / (LIO (freemod)? mex (gap2)?) /
(LIO stringnospaces) / LAO / LANAME /
(descriptn (freemod)? (((((comma)? CI (comma)?) /
 (comma &(nonamemarkers) !(AcronymicName))) name))? (gap2)?) /
 LANAME2 / LIU1 / LIE / LI)
```

This is a class of basic arguments built with articles.

Will read a name aggressively after LA without a comma; will read a name as last resort after LA with a comma.

```
arg1a <- ((DA / TAI / arg1 / (GE (freemod)? arg1a)) (freemod)?)
```

This is the full class of “atomic” arguments, adding in the pronouns.

Below we construct subordinate clauses.

```
argmod1 <- (((__LWinit (N o) ([ ])*))? ((JI (freemod)? predicate (gui)?) /
(JIO (freemod)? sentence (gui)?) / (JIO (freemod)? uttAx (gui)?) /
(JI (freemod)? modifier (gui)?) / (JI (freemod)? argument (gui)?)))

argmod <- (argmod1 ((A1 (freemod)? argmod1))*)
```

The basic kinds of argument modifier (subordinate clauses) are given (discussed above in the reference grammar). These can be linked with afterthought logical connectives.

```
arg2 <- (arg1a ((argmod (gap2)?))*)

arg3 <- (arg2 / (mex (freemod)? arg2))

indef1 <- (mex (freemod)? descpred)
```

```
indef2 <- (indef1 (gap2)? ((argmod (gap2)?))*)

indefinite <- indef2

arg4 <- ((arg3 / indefinite) ((ZE2 (freemod)? (arg3 / indefinite)))*)

arg5 <- (arg4 / (KA (freemod)? argument (freemod)? KI (freemod)? argx))

arg6 <- (arg5 / (DIO (freemod)? arg6) / (IE1 (freemod)? arg6))

argx <- (((NO1 (freemod)?))* arg6)

arg7 <- (argx ((ACI (freemod)? arg7))?)

arg8 <- (!(GE) (arg7 ((A1 (freemod)? arg7))*))

argument <- (((LAU wordset) / (arg8 AGE (freemod)? argument) /
arg8) ((GUU (freemod)? argmod (gap2)?))*)
```

The construction of arguments from basic arguments and argument modifiers is narrated above in the reference grammar. It is important to notice that top level arguments cannot begin with **ge**, to avoid ambiguity with the `AGE` connectives.

Note the ability to use **guu** to attach a subordinate clause to a complex argument.

The ordered and unordered list construction is in an entirely unsatisfactory position in the grammar. I believe that it should appear in `arg1`; I have to examine reasons why this was not done.

```
term <- (argument / modifier)

terms <- (term (((freemod)? term))*)

modifiers <- (modifier (((freemod)? modifier))*)
```

`term` is the class of arguments and modifiers. `terms` contains series of terms. `modifiers` is such a sequence with no arguments.

```
word <- ((arg1a (gap)?) / (UI1 (gap)?) /
 (NI (gap)?) / (PA2 (gap)?) / (DIO (gap)?) / (predunit1 (gap)?) / indef2)

words <- (word)+

wordset <- ((words)? lua)
```

Part of the internal grammar of ordered and unordered lists, which I will not examine until their status is regularized. I have in mind to eliminate them completely, handling the semantic space by adding suitable operators to class ZE2. This might be enhanced by nonetheless providing the same right and left markers for such constructions, optionally. The resulting notations would look very like the usual set and list notations (and the markers would also be usable with **ze**, which might be handy). This requires considerable care in design, however.

```
termset1 <- ((terms (guu)?) /
(KA (freemod)? termset2 (freemod)? KI (freemod)? termset1))

termset2 <- (termset1 ((A1 (freemod)? termset1))*)

termset <- ((terms (freemod)? GO (freemod)? barepred) / termset2 / guu)
```

These are the termsets which appear as internals of a predicate. A `terms` followed by an optional closure with **guu** or a forethought connected structure of a `termset2` and a `termset1` is a `termset1`; an afterthought connected sequence of `termset1`'s is a `termset2`.

A `termset` is a `termset2`, or a `terms` followed by **go** followed by a `barepred` predicate metaphorically modifying the predicate to which the termset is attached (an odd but potentially useful construction), or a solitary **guu** (the last is much more useful than one might suppose).

```
kekpred <- (kekpredunit (((freemod)? despredD))*)
```

The purpose here is to allow head modification with kekpreds in sentence predicates, which was not allowed in earlier versions of the grammar. This calls into question the use of the distinction between sentence and description predicates; I need to think about this.

```
barepred <- ((sentpred (freemod)? (termset)?) /
 (kekpred (freemod)? (termset)?))
```

This is the general class of untensed predicates with attached termset.

```
markpred <- (PA1 barepred)
```

This is the class of tensed predicates.

```
backpred1 <- (((NO2 (freemod)?))* (barepred / markpred))

backpred <- (((backpred1 ((ACI (freemod)? backpred1))+ (freemod)? (termset)?)
 ((((ACI (freemod)? backpred))+ (freemod)? (termset)?))?) / backpred1)

predicate2 <- (!(GE) (((backpred ((A1 !(GE)
 (freemod)? backpred))+ (freemod)? (termset)?)
 ((((A1 (freemod)? predicate2))+
 (freemod)? (termset)?))?) / backpred))

predicate1 <- ((predicate2 AGE (freemod)? predicate1) / predicate2)
```

This series of classes implements logically connected predicates with shared termsets using connectives of three series, the ACI binding most tightly, left grouped, the A binding less tightly, left grouped, and the AGE binding most loosely, right grouped. The ACI connectives are treated in the same way as the A connectives here; they are very strange in the trial.85 grammar. The systematic distinction between marked and unmarked forms is abandoned (it is provably not doing anything in the trial.85 grammar, either). The top-level sentence predicates cannot begin with **ge** due to ambiguity with the AGE connectives (something similar happens with arguments).

Shared termsets may be affixed to ACI linked and A linked predicates; the termset of the final item in the logically linked predicate may need to be closed off with GUU explicitly (or may need to be supplied as a solitary GUU); the precise ways in which this can be done are somewhat limited by the allergy of PEGs to left recursion, but I believe that in practice this grammar will be found fully capable.

In theory, this construction is rather different from the analogous construction in trial.85, but I believe that it will be found in practice to support much the same space of speakable complex predicates.

```
identpred <- (((NO1 (freemod)?))* (BI (freemod)? termset))

predicate <- (predicate1 / identpred)
```

This completes the definition of sentence predicates. These are either logically complex predicates of the last type discussed or the identity predicates (possibly negated). Note that the identity predicates *can* enter into logical composites via `kekpredunit`, for example.

```
oneargument <- (((modifiers (freemod)?))?
argument ((modifiers (freemod)?))?)

gasent1 <- (((NO1 (freemod)?))* (PA1 (freemod)?
barepred ((GA2 (freemod)? oneargument))?))

gasent2 <- (((NO1 (freemod)?))* (PA1 (freemod)?
sentpred (modifiers)? (GA2 (freemod)? terms)))

gasent <- (gasent2 / gasent1)
```

We differ from trial.85 `gasent` in having the final (ga terms) optional – so that a solitary tensed predicate is an observative (with suppressed indefinite first argument) rather than an imperative.

Further, we enforce the rule that the (ga terms) component contains exactly one argument or all the arguments in the sentence. This takes a little cunning, but is achievable.

```
statement <- (gasent / (modifiers (freemod)? gasent) /
 (terms (freemod)? predicate))
```

The statement class is modified to include all gasents as modified (thus including what were formerly tensed imperatives). In the former (terms gasent) construction, the terms are constrained to all be modifiers, which was always the intention.

```
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)))
```

We remark here on the extreme freedom of the final item in these forethought linked sentences. Look down at `uttA1` to see this.

```
sen1 <- ((modifiers (freemod)? !(gasent) predicate) /
statement / predicate / keksent)
```

The first and third cases here are the imperative sentences. The first case would parse as a statement, but is captured first by this class (modifiers followed by an untensed predicate should be imperative, and indeed this was always the intention; this parser captures this intent).

```
sentence <- (sen1 ((ICA (freemod)? sen1))*)
```

afterthought connected `sen1`'s

```
headterms <- ((terms GI))+
```

Shared fronted final arguments to be attached to a sentence in the next construction.

```
uttAx <- (headterms (freemod)? sentence (gap2)?)
```

A `sentence` with shared final arguments in fronted position. It is important to notice that fronted terms with **gi** will distribute over sentences linked with ICA connectives.

```
freemod <- ((NOUI / (SOI (freemod)? descpred (gap2)?) /
DIE / (NO1 DIE) / (KIE (comma)? utterance0 (comma)? KIU) /
 (([ ])* (H UE) name (gap2)?) /
 (HUE !(badspaces) (freemod)? descpred (((((comma)? CI (comma)?) /
(comma &(nonamemarkers) !(AcronymicName))) name))? (gap2)?) /
(HUE !(badspaces) (freemod)? statement (gap2)?) /
(HUE !(badspaces) (freemod)? termset1) /
 (HUE stringnospacesclosedblock) / (([ ])* (H UE) (',' ([ ])+) name (gap2)?) /
voc / CANCELPAUSE / PAUSE / JO / UI1 /
(([ ])* '...' ((([ ])* &(letter)))?) /
 (([ ])* '--' ((([ ])* &(letter)))?)) (freemod)?)
```

The free modifiers. These are discussed in the reference grammar. The important change here is the elimination of unmarked names as vocatives. The class `freemod` contains all free modifiers other than pauses; it is used to allow pause/GU equivalence, and if it is set equal to freemod, pause/GU equivalence is disabled. The class `freemod` preserves a very few cases where things must remain pauseless. freemod and freemod are now the same as freemod.

Ellipses and dashes (double hyphens) are freemods, enhancing our punctuation.

There is no preprocessing in this grammar, so `freemod` appears ubiquitously in other rules.

will read a name aggressively after HUE without a comma pause; with explicit comma will try other readings first.

```
uttA <- ((A1 / IE1 / mex) (freemod)?)

uttA1 <- ((sen1 / uttAx / NO1 / links / linkargs / argmod /
 (terms (freemod)? keksent) / terms / uttA) (freemod)? (period)?)
```

Note the various fragments of utterances allowed here as answers to questions, along with full classes of sentences.

```
neghead <- (NO1 (gap / PAUSE))

uttC <- ((neghead uttC) / uttA1)

uttD <- (uttC ((ICI (freemod)? uttD))*)

uttE <- (uttD ((ICA (freemod)? uttD))*)

uttF <- (uttE ((I (freemod)? uttF))*)

utterance0 <- (!(GE) ((!(PAUSE) freemod (period)? utterance0) /
 (!(PAUSE) freemod (period)?) / (uttE 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) /
 (uttE IGE utterance) / (I (freemod)? (period)? ((&(I) utterance))? end) /
(uttF ((&(I) utterance))? end) / (I (freemod)? uttF ((&(I) utterance))? end) /
 (ICA (freemod)? uttF ((&(I) utterance))? end)))
```

The top-level construction of utterances is discussed above in the reference grammar. Recall that the class `end` signifies end of text or the beginning of a Loglan utterance in another voice marked by `#`. There is no analogous ability of the class utterance0 to be chained in this way: `#` is not a quotable or parenthesizable piece of Loglan punctuation.

11 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,
haitosaiofo, 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, haitosaiofo, 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 ZO
%{/*:  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 {$$=NodeY1 ("argx", &$1);}
|   NO3 argx {$$=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);}
;

%%
```