Loglan Academy Agenda

Randall Holmes

September 12, 2015

Contents

This document contains the previous list of proposals before the Loglan Academy, and presents my Grand Proposal to bring the official definition of the language into line with something like my PEG parser. This is first presented in the form of a detailed review of the PEG grammar in three sections (Phonology, Lexing, and Grammar), then recapitulated in the form of a list of subproposals.

If other members of the academy have Proposals, communicate them to me and I will add them here! Members of the list may suggest things to us as well.

6/6/2014, important: Section 2.1 on Loglan phonology (culminating in formal definitions of name words and predicate words) should now have something close to enough supporting English to be read. I believe that there is actually very little in this section which is strictly speaking a change in the official language definition. The new vowel stream grouping convention is novel, I suppose, and the notation for syllable breaks and stresses is new. The

rule that the last two consonants in a syllable cannot be a non-continuant followed by a continuant is new but I hope will not be controversial. The proposal that q,w,x be dropped is included, and so is John's alternative construction of complexes using `zao`. The imposition of syllable structure on names is new, but virtuous.

6/7, 6/8: lexicography section reorganized for readability with minor corrections and all new text. Quotation and foreign text forms made more uniform 6/8.

6/9: The grammar section is rewritten as well. This should look a lot more like at least the outline of a reference grammar.

This (or at least my parts of it) can also be viewed as my complete working notebook on parsing Loglan. I'm planning to enhance it with examples and references to previous texts and support it with essays on particular topics. Parsing the NB3 corpus (completed) and the ongoing process of parsing material from L1 is informing corrections and improvements.

Although this document does recommend various changes in the language, my intentions are conservative. You may examine my parse of the entire NB3 corpus to see that the scope of the changes suggested here is relatively minor. The real underlying reason is that I want to replace LIP as the official standard parser, because it is impossible to update it (I think) and quite difficult to look under the hood to see what it is doing on the phonological and lexer levels. I would like to get to a position where we can regard this as the official parser; I am well aware that this will require a process of deliberation over individual subproposals and also a process of testing and debugging of this parser.

I have now added the titles but not the text for various essays that I have promised at the end of this document.

6/6 Proposal 8 has passed (and we now officially have one pause phoneme again). This means that I have the work assignment of actually revising Appendix H and getting things posted.

6/9 I am reformatting the list of proposals, and eliminating references to ones which have been completely processed into Appendix H.

6/11 `gui` can terminate any argmod1; CV and CVV cmapua can be prefixed to a predicate with ZAO. There is a new Proposal 4 listed.

7/30 technical changes to juncture rules preventing complexes with illegally placed syllable breaks from being parsed as borrowings. Change to zao constructions.

8/1/14: major changes to acronyms. What were acronymic predicates

are now acronymic names. The mandatory front markers on acronyms have been replaced with a single one which is mandatory on an acronym of one letter (so this can be told from a letteral) and optional on other acronyms. Both sorts of acronyms are terminated with an explicit pause. 8/2: fixed a bug in acronyms due to misimplementation of 8/1 revision; reintroduced front marking of dimensions (for good reasons); fixed a missing freemod in `sentence` class. 8/3: further refinements: acronymic names, like those with false name markers, are **always** marked. The `le blanu, Djan` construction now allows a CI marker on the name, and requires it if the name is acronymic or contains a false name marker. 8/4 further refinements: the CI which links predicates cannot be followed by a name.

8/4 zero drafts of essays on names and acronyms added.

notes to myself: I have not imposed stress rules on numerical or acronymic predicates. The parser recognizes the ends of such predicates without whitespace or terminal punctuation in any case, I believe. After 8/1, there are no acronymic predicates; acronymic names need no stress rules.

9/20/2014 changing the text to fit the new version with PO always long scope. I believe that this is the best all-around solution; a supporting essay is probably wanted. The issue has very little to do with the structure word break in LEPO, though this removes that problem. The real issue is that "PO sentence" predicates have very limited grammatical privileges in the previous version: one should be able to say `blanu po mi cluva` if one wants, but in the current version a PO sentence predicate cannot enter into a metaphor. I arrange things subtly so that a LEPO clause and a PO sentence predicate are both closed by GUO (it is never necessary to apply two closures as I hear it can be in the sister language).

9/30/2014 minor changes to parsing of borrowings.

12/6/2014 Minor tweaks to older text. Introduced CA-connected quantifiers (such as SUCERA, SUCENOIRA. Excluded NOI-initial compound PA words; NOI needs always to be a final form. NOI can be used to negate second and further components of the new compound quantifiers but not initially. It appears that logical transformations enable us to say all the things that are excluded by this last, and while these words were found in the NB3 corpus I have not seen them in Loglan usage. I can produce examples where they lead to ambiguity where one either has ANOI PACE... or A NOIPACE... with different meanings.

12/18/2014 Various adjustments which came up in the course of writing the lexicography section of the reference grammar. Put NOCA and NO-

CANOI forms back as possible links in PA (and now NI) words. Adds words AA EE OO added since trial.85 to UI0. Makes it possible to quote ZIY and ZIYMA with LIU (these are names of the letter y, and a problem because their form is phonetically irregular). Allows untensed A connectives to be closed with GU or pause.

5/15/2015 Fixed a bug in closures of A words with GU. Forbade spaces internal to words. There are many niggling little changes to effect this some of which might have been overlooked here.

5/16/2015 Refinements of the 5/15 update not yet entered here: in general, wherever the PA word component class can appear, whitespace followed by a PA word component is forbidden. After an A word, it is just whitespace followed by a PA followed by a a GU or pause that is forbidden. The idea is to ensure that where a space appears one can always pause; but there is an exception: when an A word is followed by a PA word then a space, replacing the space with a pause makes an APA word (possibly with an illegal space in it) which either breaks the parse or changes it essentially. I will enter the changes to the rules shortly. The 5/15 and 5/16 upgrades did detect some errors or at least usages open to question in the corpus. This also illustrates how nasty APA words are. I do need to write an essay about them, including how they cause trouble under LIP. I preserve them (for now) because IPA words seem to be important, and anything that defends IPA words also defends APA words.

also 5/16 I removed capitalization restrictions in the interior of acronymic names (so **la DaiNaizA** is legal). A solution which also works in acronyms used as dimensions might be appealing.

5/20/2015 Noticed that **ii** and **uu** with explicit juncture is an instance of RepeatedVowel. I made the 5/16 changes in the text as far as I could; I am beginning to think that enough small changes were made in the "no spaces inside words" initiative that there ought to be a full overhaul to ensure that all rules are directly copied from the current grammar rather than manually corrected with possible errors and omissions as now.

7/17/2015 Corrected a nasty bug in the juncture class (related to my implementation of the rule that last syllables of cmapua stressed before predicates must be followed by pauses). Allows converse forms of the BI class predicates (this should be uncontroversial: these predicates are clearly reasonable and likely to be wanted).

9/5/2015 New proposal concerning basic sentence structure: marked-predicate-initial sentences are not imperatives but gasents, and if the final

ga clause is missing **ga ba** is to be understood.

9/12 further action re proposal of 9/5, allowing modifiers to appear at the head of an imperative. It is useful to note that the changes of 9/5 and 9/12 should have hardly any effect on what sentences are parsable: the effects are mainly to recategorize which sentences are imperatives. An imperative is a sentence in which no terms or only modifiers precede an unmarked predicate. A sentence in which no terms or only modifiers precede a marked predicate is a declarative sentence with an unexpressed indefinite subject.

1 The Currently and Recently Active Proposals 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!

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.

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.

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`).

2 Description of the PEG Grammar and Grand Proposal

This is the text of the PEG grammar with commentary. I do have the aim of getting this (with whatever changes are needed) to be approved as the official Loglan parser, with the proviso that the English text should be consulted re intent in case of bugs.

This section is an official Proposal. Details are most certainly open to discussion. The overall Grand Proposal will be divided into three sections, Phonology, Lexicology, and Grammar.

2.1 Loglan Phonology

2.1.1 Introductory Remarks

6/6/2014: Rewriting the first section of the grammar into proposal language.

This section contains the exact text of the relevant parts of `Loglan.peg` as of 9 am on D-day 2014, with some corrections that I found I had to make in the course of reading the rules to write the English text. The English text will be a self-contained description of the phonology of Loglan, up to the level of formation of the various classes of words.

2.1.2 Letters and symbols

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

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

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

This text includes the proposal that the letters `qwx` included in NB3 and L1 accounts of the language be dropped. These letters remain usable in quotation forms and other forms which include foreign text, but they are not Loglan letters.

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

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

A juncture is a member of the class of symbols consisting of the hyphen, the apostrophe and the asterisk.

The hyphen stands for a syllable break and may appear between two syllables. To write a hyphen is optional, except that it may be required to force a desired pronunciation of a name. The parser deduces syllable breaks in Loglan text when not supplied with them explicitly.

The apostrophe stands for stress on a syllable and the asterisk for emphatic stress on a syllable. There is no difference between these that affects parsing of any Loglan utterance, but the distinction between the two forms can communicate emphasis. Either kind of stress can appear between two syllables, indicating stress on the preceding one, or at the end of a syllable which is not followed by another syllable.

To include stress is always optional. If stress is not given explicitly, the presence of a stress can be deduced from whitespace or terminal punctuation at the end of a predicate word.

An explicit pause is indicated by a comma followed by one or more spaces.

A hyphen is always followed by a letter.

A stress of either sort may not be followed by (whitespace followed by) a predicate word. It may not be followed by a hyphen (either sort of stress includes the sense of a hyphen) or by another stress mark. A stress marker may be followed by (and include, for purposes of the grammar) an explicit pause (a comma followed by whitespace) if it is followed immediately by a predicate word. This enforces the rule that one must pause after a stressed unit cmapua before a following predicate. 7/17 this rule is enforced using the second form `juncture2`, only in the context of the vowel classes used for building cmapua.

Note that a juncture symbol is never followed by another juncture symbol.

A style of writing Loglan which involves showing all stresses (and optionally all syllable breaks) and showing whitespace only after commas (explicit pauses) is intended to be supported. This we will call a phonetic transcript. JCB's own phonetic transcripts can be converted quite mechanically to this form.

In some situations, pauses will be deducible from whitespace without an explicit pause, and in some cases stress can be deduced from such. Most whitespace has no phonetic import at all.

We note here that Loglan now has one pause phoneme, though it may be expressed in various ways.

```
Lowercase <- (lowercase / juncture)

Letter <- (letter / juncture)
```

These classes add junctures to the indicated class of letters.

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

\end{description}

This is the comma, representing an explicit pause.
Introduced 8/2, all rules mentioning this must have been modified at that point

\begin{description}

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

These characters are the supported forms of terminal punctuation. This writer is for the most part not inclined to replace Loglan words with bits of punctuation, though willing to explore such styles if clearly defined.

2.1.3 Sounds and sound combinations

```
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]
```

The Loglan consonants. Their pronunciations are mostly as an English speaker would expect. `g` is never soft as in `gem`, but always hard as in `gun`. `c` is pronounced as English `sh` as in *sh*oe in all contexts. `j` is pronounced as the voiced consonant corresponding to `c`, which is the pronunciation of `z` in a*z*ure in my idiolect. `n` is usually pronounced as English *n*ose, but before `g` or `k` as in English fi*n*ger or ta*n*ker. Since the hard `ch` of Scottish `loch` is no longer the sound of the banished Loglan `x`, it is available as an alternative pronunciation of Loglan `h`.

Nothing in this paragraph is intended to be new.
See below for syllabic pronunciation of `mnlr`.

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

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

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

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

u <- ([Uu] (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)?)

V1 <- [AEIOUYaeiouy]

V2 <- [AEIOUaeiou]

C1 <- (!(V1) letter)

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

EMono <- (([Aa] [o]) / ([AEIaei] [i]))

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

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

This body of rules is related to Loglan vowel sounds and vowel grouping.

The regular Loglan vowels are `aeiou`. `y` is a special vowel.

The regular vowels are pure sounds.

English f*a*ther (not h*a*t) sounds like Loglan `a`. English long `i` as in k*i*te sounds like Loglan `ai`.

English g*e*t sounds like Loglan `e`. English long `a` as in g*a*te sounds like Loglan `ei`.

English `ee` as in b*ee*t or m*e*te, or `i` in mach*i*ne, sounds like Loglan `i`. This is a pure sound; an English speaker has a bad habit of following it with something like consonantal `y`.

English `aw` as in l*aw* or `o` as in l*o*st sounds like Loglan `o`. The English long `o` as in b*o*ne sounds like Loglan `ou` (though Loglan `ou` is definitely two syllables).

English `u` as in l*u*re or t*u*ne sounds like Loglan `u`. English speakers should avoid the habit of closing this up with something like consonantal `w`.

These sound values for the regular vowels are a bit odd to an English speaker but quite usual in continental Europe.

The special vowel `y` has the "schwa" sound of `a` in English sof*a* or `u` in English h*u*nt. An English speaker should avoid the bad habit of turning unstressed regular vowels into `y`: these must always be clearly pronounced. John Cowan suggests the more definite sound in English l*oo*k for `y`; I like this (it can be an allophone; the schwa is clearly appropriate in some contexts, but clearer articulation might be useful in others).

The letter pairs `ao`, `ai`, `ei`, `oi`, when grouped together, have mandatory monosyllabic pronunciation. The pronunciation of `ao` is surprising, as in English c*ow*. The pronunciations of `ai`, `ei`, `oi` are as in English t*ie*, p*ay*, b*oy*.

The letter pairs with initial `i` or `u` may be pronounced as two syllables or as monosyllables with intial `i` (resp. `u`) taking on the pronunciation of English consonantal `y` (resp. `w`). These are called optional monosyllables.

Other vowel pairs if they appear together in a word must be pronounced as two separate syllables. See below for a special rule for the pairs `aa`, `ee`, `oo`.

Nothing above this point in the account of vowels and vowel grouping is intended to be new.

When a stream of regular vowels is presented (in the context of a borrowing or name word: a stream of vowels in a cmapua is organized into a stream of VV words with a possible initial V word), one segments it starting at the left using the following precedence: first, take a mandatory monosyllable, if present; next, take a single vowel if it is immediately followed in the stream by a mandatory monosyllable (which will then be taken); next, take an optional monosyllable; next, take a single vowel. Repeat as necessary. Whitespace or juncture interrupts this process (juncture can be used to force different grouping). This algorithm for grouping vowels is a new proposal.

```
CVVSyll <- (C1 Mono)

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

LW1 <- (((V2 V2)
/ (C1 !(BrokenMono) V2 (juncture)? V2)
/ (C1 V2)) (juncture2)?)
```

The word classes above are used to represent units from which structure words (Loglan *cmapua*) can be constructed. The actual rule for constructing structure words phonetically has very limited use in Loglan grammar and in fact does not appear in this section.

Before we enter the lexer section, all we need to know is that unit cmapua are of one of the forms V, VV, CV, CVV. We really truly need to know nothing else (the whole compound cmapua parsing algorithm in NB3 is irrelevant). In the lexer section, we will construct various classes of compound cmapua in individual word classes (which will just happen to satisfy the NB3 rules, but without any use of them). The one place where the NB3 rules are used is in determining what a single cmapua word is for purposes of LIU quotation, which naturally comes later. I believe that LIP never uses the NB3 compound cmapua rules at all in any way!

12/18/2014 A hack Added ZIY and ZIYMA to this class by force so that LIU will be able to quote these names of the letter Y. This needs some fixes: there is no need to add ZIYMA, and the option of putting a syllable break in ZIY should be supported.

```
caprule <- ((uppercase / lowercase) ((lowercase / juncture))* !(letter))
```

This rule enforces the Loglan capitalization convention. A string of letters and junctures unbroken by any other characters will contain an uppercase letter only at the beginning. Capitalization is always optional.

```
InitialCC <- ('bl' / 'br' / 'ck' / 'cl' / 'cm' / 'cn' / 'cp' / 'cr'
/ 'ct' / 'dj' / 'dr' / 'dz' / 'fl' / 'fr' / 'gl' / 'gr' / 'jm' / 'kl'
/ 'kr' / 'mr' / 'pl' / 'pr' / 'sk' / 'sl' / 'sm' / 'sn' / 'sp'
/ 'sr' / 'st' / 'tc' / 'tr' / 'ts' / 'vl' / 'vr' / 'zb' / 'zv'
/ 'zl' / 'sv' / 'Bl' / 'Br' / 'Ck' / 'Cl' / 'Cm' / 'Cn' / 'Cp'
/ 'Cr' / 'Ct' / 'Dj' / 'Dr' / 'Dz' / 'Fl' / 'Fr' / 'Gl' / 'Gr'
/ 'Jm' / 'Kl' / 'Kr' / 'Mr' / 'Pl' / 'Pr' / 'Sk' / 'Sl' / 'Sm'
/ 'Sn' / 'Sp' / 'Sr' / 'St' / 'Tc' / 'Tr' / 'Ts' / 'Vl' / 'Vr'
/ 'Zb' / 'Zv' / 'Zl' / 'Sv')

MaybeInitialCC <- (([Bb] (juncture)? [l])
/ ([Bb] (juncture)? [r]) / ([Cc] (juncture)? [k])
/ ([Cc] (juncture)? [l]) / ([Cc] (juncture)? [m])
/ ([Cc] (juncture)? [n]) / ([Cc] (juncture)? [p])
/ ([Cc] (juncture)? [r]) / ([Cc] (juncture)? [t])
/ ([Dd] (juncture)? [j]) / ([Dd] (juncture)? [r])
/ ([Dd] (juncture)? [z]) / ([Ff] (juncture)? [l])
/ ([Ff] (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]))
```

The rule `InitialCC` lists the pairs of Loglan consonants which may appear in initial position in a syllable. The rule `MaybeInitialCC` identifies such pairs even if they are separated by a syllable break (this is used to detect certain error conditions). The pairs `sv` and `zl` accidentally omitted in L1 are restored.

```
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]))

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]))
```

These rules define sequences of consonants which may not occur in a Loglan word. They are taken directly from NB3 and L1. Some additional sequences are proscribed by a new rule which appears below, but they are ones which are rather unlikely to appear in a proposed word.

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

The "repeated vowel" sequences `aa`, `ee`, `oo` have a special associated rule. Since they are not optional monosyllables, the two vowels in such a sequence must appear in adjacent syllables. The special rule is that one of them must be stressed (this does not need to be shown explicitly). This rule also applies to `ii`. `uu` when they are explicitly disyllables (noticed and revised 5/20/2015).

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

Syllabic <- [lmnr]

Nonsyllabic <- (!(Syllabic) C1)

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

The consonants `mnlr` admit use as syllabic consonants (we now deprecate the usage "vocalic", though it remains embedded in our texts and in some class names in this grammar). The only consonants which appear repeated

in Loglan orthography are the continuants `mnlr` (the ones which can be syllabic). When they appear repeated, they are always syllabic, and if a syllabic use of these consonants is intended, they will appear repeated. Note that this forces a change in the spelling of certain names, but note also that JCB suggests this in L1 as an alternative convention. The distinction between a consonantal `mnlr` and the syllabic (doubled) version is phonemic, in spite of incorrect statements in L1.

Syllabic consonants occur only in names and in borrowed predicate words.

The class `Badfinalpair` is used to enforce a new rule that the final two consonants in a syllable cannot be a non-continuant followed by a continuant, as such a sequence would basically have to be pronounced as another syllable. This forbids certain three letter sequences of consonants not explicitly forbidden in L1 or NB3, and forbids certain placements of syllable breaks.

2.1.4 Syllables, to the definitions of name and borrowed predicate words

```
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))
```

These rules define the sequences of consonants which can appear at the beginning of a Loglan syllable. The first version is used in predicate words, and the second in names.

Such a sequence of consonants can have between one and three letters in it, all consonants. Each successive pair of these letters must be a permissible initial pair. (This rule appears in NB3). The final consonant in the sequence cannot be the first consonant in a syllabic consonant pair. An initial consonant sequence will of course not be followed by a juncture. In a predicate word, an initial consonant sequence will not be followed by `y`; this is permissible in a name.

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

VowelSegment2 <- (NextVowels / RepeatedVocalic)
```

This rule defines the "vowel" in a Loglan syllable (which may be a monosyllabic pair of vowels or a syllabic consonant). The rule `Nextvowels` chooses a single vowel or a pair of vowels from a stream of vowels in the way described above.

9/30/2014 The vowel segment in a predicate may not be followed by a syllabic consonant.

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

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

SyllableFinal1 <- ((FirstConsonants)? !(RepeatedVocalic)
 VowelSegment !([\'*]) ([-])? (&(Syllable) / &([y]) / !(letter)))

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

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

SyllableFinal2b <- ((FirstConsonants)? !(RepeatedVocalic)
VowelSegment [\'*] (&([y]) / !(Letter)))

StressedSyllable <- (((FirstConsonants)? !(RepeatedVowel)
 !(&Mono V2 RepeatedVowel)  VowelSegment
!(Badfinalpair) (FinalConsonant)? (FinalConsonant)?) [\'*])

FinalConsonant <- (!(NonmedialCC) !(NonjointCCC)
```

```
 !(Syllable) C1 !((juncture V2)))

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

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

This 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 sequence `aa ee oo` 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 can be simplified (FIX).

The rule `JunctureFix` is a technical device to prevent the parser from allowing borrowings which are variants of illegal complexes formed simply by moving syllable breaks. A vowel-final syllable may not be followed by an initial consonant pair – 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 may not end with a consonant followed by followed by an initial consonant pair broken by a juncture. The reason for this is that neither of these situations is allowed to occur in a complex predicate, because syllable breaks in a complex must respect breaks between djifoa. The point of this rule is to prevent complexes with illegally placed syllable breaks from parsing as borrowings.

9/30/2014 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 ([\'*])? !(Letter))) Lowercase))*
C1 ([\'*])? !(Letter) (!(.) / comma / &(period)
 / &(Name) / &(CI)))) ((Syllable2)+
 (!(.) / comma / &(period) / &(Name) / &(CI))))
```

A name word (this is only a first approximation: further rules in the grammar not presented in this section are used to peel words of other classes off the front of a name word) is optional whitespace followed by a sequence of letters and junctures satisfying the capitalization rules, resolvable into syllables (permitting `y` as a vowel segment and with no restriction on repeated vowels), and ending in a consonant and an optional final stress, followed by an explicit pause (the comma being included in the word for parsing purposes) or immediately followed by end of text, or non-included terminal punctuation, or another name, or the word `ci` (the last two provisions enable us to write serial names without commas).

This is phonetically seemingly far more restrictive than the rule given in L1, but in practice the only change which seems to be required for names in the corpus is to double any consonants used syllabically.

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

A syllable whose "vowel" is a syllabic consonant.

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

A borrowing tail is the two or three syllables final in a borrowing beginning with the stressed syllable, followed by an optional syllable with a syllabic consonant (such syllables are not counted for determination of predicate penultimate stress) followed by an unstressed syllable. The end of the borrowing is identified either by finding an explicit stress or by seeing whitespace after a final syllable (which cannot be explicitly stressed).

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

A pre-borrowing is a stream of syllables which are neither explicitly stressed nor stand at the beginning of a borrowing tail, followed by a borrowing tail. This class is used in the algorithm for finding the left end of a borrowing, below.

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

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

CCVV <- ((&(BorrowingTail) C1 C1 V2 [\'*] V2)
/ (&(BorrowingTail) C1 C1 V2 (juncture)? V2
(!(Letter) / ((juncture)? [y]))))

Borrowing <- (!(CVCBreak) !(CCVV) &(HasCCPair)
 !((V2 (juncture)? MaybeInitialCC V2))
```

```
!(CCSyllableB) (((!(BorrowingTail)
!(StressedSyllable) !(JunctureFix) Syllable))* !(CCSyllableB) BorrowingTail))
```

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.

There are no CCVV 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).

9/30/2014 The CC found in HasCCPair cannot be a permissible initial broken by a juncture.

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

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


StressedBorrowingAffix <- (!(CVCBreak) !(CCVV)
 &(HasCCPair) !((V2 (juncture)? MaybeInitialCC V2))
 !(CCSyllableB) (((!(StressedSyllable) !(SyllableFinal2a)
```

```
!(JunctureFix) Syllable))+ SyllableFinal2b) (juncture)? [y] ([-])? !([,]))
```

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.

I used to restrict the length of the sequence of vowels before the first consonant pair; I no longer do, but I still think very long ones would be absurd.

2.1.5 Djifoa ("affixes") to the definitions of complex predicate and predicate in general

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] !([\'*]) ([-])? !([y]) &(letter))
```

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)) !([\'*]) ([-])?)
```

A CV is the final syllable of a five letter djifoa. It is never stressed.

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

A Cfinal is the final consonant of a CVC djifoa. It will be manifested either as a consonant followed by a y-hyphen or a consonant not starting a forbidden sequence of consonants and not followed (mod possible intervening juncture) by a vowel.

```
hyphen <- (!(NonmedialCC) !(NonjointCCC)
 (([r] !(((juncture)? [r])) !(((juncture)? V2)))
/ ([n] (juncture)? &([r]))
 / ((juncture)? [y] !([\'*]))) ((juncture)? &(letter))
 !(((juncture)? [y])))
```

This is a formal description of all the various kinds of hyphens used to fix pronunciation problems at djifoa boundaries (including y-hyphens). This will be an `r` not followed by another `r` or a vowel, or an `n` followed by an `r`, or a y-hyphen. No hyphen is followed by a `y` (mod intervening juncture). The hyphen is regarded as part of the preceding djifoa for parsing purposes.

```
StressedSyllable2 <- (((FirstConsonants)?
 VowelSegment !(Badfinalpair)
FinalConsonant (FinalConsonant)?) [\'*])
```

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

```
CVVStressed <- (((C1 &(RepeatedVowel)
V2 !([\'*]) ([-])? !(RepeatedVowel) V2 (hyphen)?) (([\'*] / juncture))?)
/ (C1 !(BrokenMono) V2 [-] V2 [\'*]) / (C1 !(Mono) V2 V2 [\'*]))
```

This 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.

```
CVV <- (!((C1 V2 [\'*] V2 [\'*]))
 ((C1 !(BrokenMono) V2 (juncture)?
!(RepeatedVowel) V2 (hyphen)?) (juncture)?))
```

The completely general form of a CVV djifoa, completely decorated with options of hyphens and junctures.

```
CVVFinal1 <- (C1 !(BrokenMono) V2 [\'*] V2 ([-])?)

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 !([\'*]) ([-])?)

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

CVV djifoa which may be final in a complex, either because of an explicit stress on the first syllable of two or because they have two syllables and end in non-letters (forms 1 and 2)

Form 5 is final in a borrowing affix

Forms 3 and 4 are possible or actual final CVV monosyllables, actual if followed by a non-letter, possible if not explicitly stressed or broken by an explicit juncture.

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

CVCStressed <- (C1 V2 !(NonmedialCC) !(NonjointCCC) C1 [\'*] (yhyphen)?)
```

CVC djifoa, stressed and otherwise. These cannot be final in a complex.

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

CCVStressed <- (InitialCC !(RepeatedVowel) V2 [\'*])

CCVFinal1 <- (InitialCC !(RepeatedVowel) V2 !([\'*]) ([-])?)

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

CCV djifoa, their stressed form, and two final forms, the certainly final form without a following letter or juncture and the possibly final form without an explicit stress.

```
CCVCVMedial <- (CCV C1 [y] ([-])? &(letter))

CCVCVMedialStressed <- (CCV [\'*] C1 [y] ([-])? &(letter))

CCVCVFinal1 <- (InitialCC V2 [\'*] CV)

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

The CCVCV five letter djifoa and variations. The first two are the medial forms with final y in stressed and unstressed forms. Two final forms are given for CCVCV but of course expression of the final vowel forces this to be final in any case. This could be simplified though it is not an error (FIX).

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

CVCCVMedialStressed <-
((C1 V2 ([\'*] &(InitialCC)) !(NonmedialCC) C1 C1 [y] ([-])? &(letter))
/ (C1 V2 !(NonmedialCC) C1 [\'*] C1 [y] ([-])? &(letter)))

CVCCVFinal1a <- (C1 V2 [\'*] InitialCC V2 ([-])?)
```

```
CVCCVFinal1b <- (C1 V2 !(NonmedialCC) C1 [\'*] CV)

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

The CVCCV five letter djifoa in their variations. This case is more complicated because there are two different places where an internal syllable break or stress can be placed: CV-CCV or CVC-CV. The final forms again can probably be simplified.

```
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.

```
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.

```
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) / (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.

```
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)?
 (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.

```
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.

6/11 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.

7/30 added option of an explicit pause after ZAO. I am suspicious of my earlier belief that ZAO could be followed by a vowel initial predicate without a pause; I think the ZAO would affix to the front of the predicate word instead (a pause guards against this).

2.2 Loglan Lexicography

2.2.1 Introductory remarks

This section deals with details of Loglan that are for the most part not manifest in the previous official formal grammar. The word classes are defined by LIP using rather opaque internal representations, and there are clearly bugs. Our program is 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. 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.

6/7/2014, complete rewrite: Here I am working on drafting a proper narrative version of the lexer section. This involved moving things around; alphabetical order is not a narrative structure.

2.2.2 Auxiliary rules

```
__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).

This happens in the previous grammar only before VCV letterals, which are still present in this grammar but which I intend to remove.

```
__LWbreak <- (!(Oddvowel) !((!((([ ])* Predicate))
(A / ICI / ICA / IGE / I))) ((comma &((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). It isn't clear to me that this class in its present form has much to do with cmapua in particular (LOOK INTO THIS).

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

PAUSE <- (!(CANCELPAUSE) comma
 !((A / ICI / ICA / IGE / I)) !((&(V2) Predicate)))
```

PAUSE contains those pauses which have potential grammatical significance. Such a pause will not be followed by a logical connective or a vowel initial predicate. It will not be an instance of CANCELPAUSE: a pause followed by `y` followed by a pause ("uh"...) or by the word `cuu` is considered to have been inadvertant. Of course CANCELPAUSE can also be used deliberately for effect!

Pauses after names or after stressed syllables before predicates are also not PAUSE, as they are consumed by previous strings (there are other instances of this kind of explicit pause consumed before it can get into PAUSE).

2.2.3 Logical connectives

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

A <- (__LWinit !(Predicate)
(((N u) &((u / (N o)))))? ((N o))? A0 ((N OI))?
```

```
!([ ]+ PA ((!Predicate G u)/PAUSE))
 ((PA? ((!Predicate G u) / PAUSE)))? !(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 which will be closed with either `gu` or an explicit pause. The only change here from the L1 version of the language is the requirement that APA words be closed with `gu` or an explicit pause.

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.

12/18/2014 Allows an untensed A word to be closed with GU or a pause.

5/15/2015 fixes a bug in the 12/18 innovation.

5/16/2015 forbids an A word without a PA suffix to be followed by whitespace then a pause or GU closed occurrence of PA.

A1 is inhabited by top level A words.

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

AGE <- (A (PA)? !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.

5/15 forbids spaces in these words.

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

CA1 <- (!Predicate (((N u) &(((C u) / (N o)))))?
 ((N o))? CA0
 ((PA ((G u) / PAUSE)))? !(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.

12/18/2014 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 <- (__LWinit i !([ ]+ PA) ((PA ((!Predicate (G u) )/ PAUSE)))? __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.

5/15 updates to prevent spaces in middles of words.

5/16 refinements to the 5/16 updates forbidding replacement of a PA suffix form with whitespace followed by a PA suffix form. It is possible that the 5/16 mod to class I should follow the form of that to class A (forbidding only whitespace followed by a pause or GU closed PA suffix form) but I think the present form is justifiable.

```
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)) __LWinit KOU)

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

KI <- (__LWinit (K i) ((N OI))? __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.

2.2.4 Numerals, letters, acronyms, pronouns

```
NI0 <- !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 OI) / (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 a) / (S e) /  (S i)
(S o) / (S u) / (H i)) !(Oddvowel))
```

Atomic numeral and quantifier words.

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

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. I am not sure what the name for `y` should be: `ziy` is supported but surely would cause problems. C1-`ai` and C1-`ei` name upper and lower case consonants. C1-`eo` names Greek letters. We should have CVV letterals for `qwx`; these letters are quite often used in mathematics.

The principal function of these words is as pronouns!

```
NI1 <- (NI0 ((M a))? ((M OA (NI0)*))? !(Oddvowel))
```

This rule 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` is different 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.

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

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

5/16 guarded from being initial in a predicate, as many word component forms are in the 5/15 and 5/16 updates.

```
IE1 <- (__LWinit IE __LWbreak)
```

The interrogative "which", also used in argument constructions.

```
NI <- (__LWinit (IE1)?
 (((((RA / (NI1 &(NI1))))* NI1) / RA)
(CA0  ((((RA / (NI1 &(NI1))))* NI1) / RA))*
 !((([ ])+ !(Predicate) (NI1 / RA)))
((&((M UE)) Acronym (comma
/ !(.) / &(period)) !((C u)) !((P UI))))?
(((C u) / ((comma)?
 ([ ])* P UI &(NI))))? !(Oddvowel))

mex <- (__LWinit NI __LWbreak)
```

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 string of NI1's and RA's in which a RA is not final unless it is the sole item in the string: this component is a complex number or quantifier; one may optionally append an acronym followed by an explicit pause as a dimension, and there is a final option of appending `cu`. A NI item may be terminated by the construction `pui`, or by an explicit pause followed by `pui` (yes, I really mean for the pause to be before `pui`). The termination option is new, avoiding the need for explicit pauses to determine boundaries between adjacent NI constructions (though one can use pauses to define such boundaries as well).

A fix is needed to ensure that a NI class form ends after an acronym.

12/6/2014 allowing CA0 connected quantifiers (with noninitial negations with NOI allowed)

12/18/2014 optional NO and NOI attached to CA0 are now internal to that class

```
Acronym <- (([ ])*
 ((M UE) / TAI0 / ([Zz] V2 !(V2)))
 ((NI1 / TAI0 / ([Zz] V2 (!(V2) /
([Zz] &(V2))))))+ !((([ ])* !(Predicate)
 (NI1 / TAI0 / ([Zz] V2 (!(V2) / ([Zz] &(V2)))))))
 (([,] &((([ ])+ !(Predicate) (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 and to form acronymic names (no longer acronymic predicates).

5/16 capitalization in the interior of acronymic names (but not dimensions) made free by eliminating caprule here. This restores usages exhibited in NB3 and other Loglan sources, along the lines of `la DaiNaizA`. A similar update permitting free capitalization in dimensions would be trickier to implement but not undesirable.

```
TAI <- (__LWinit (TAI0
/ ((G AO) !(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 ![ ] NI))? !(Oddvowel))

DA <- (__LWinit DA1 __LWbreak)
```

Pronouns. These can be suffixed with mathematical expressions using `ci`. 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.

5/15 changes to avoid spaces in middles of words.

There can be a need to pause after an acronym (of either kind) to avoid it absorbing a letteral. [I am contemplating **requiring** a pause after any acronym in all contexts].

2.2.5 Tense/location/relation operators

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 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) / (D II) / (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)
/ (V i) / (V u) / KOU) !(Oddvowel))
```

The atomic tense/location/relation operators. The causal ones have already been given in class KOU.

5/16 guarded from being initial in a predicate.

```
PA <- (!Predicate ((!(PA0) NI))?
((KOU1 / PA0))+ (((CA0
((KOU1 / PA0))+))*
!([ ]+ PA)
((ZI / ((', ')?
([ ])* J UU &(PA))))? !(Oddvowel))
```

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 with JUU (possibly preceded by a pause) followed by another PA word. JUU is a new device enabling one to separate adjacent PA words without pausing (although one can still also separate such words with an explicit pause).

12/6/2014 eliminated the option of an initial NOI, which can be ambiguous with a preceding A connective such as ANOI

12/18/2014 optional NO and NOI attached to CA0 are now internal to that class

5/15 updates to prevent spaces internal to words. !Predicate often replaces LWinit in definitions of word components.

5/16 a PA word not closed with ZI or JUU forms cannot be followed by whitespace and another PA: PA words must be conglomerated where not explicitly separated with commas as part of the no spaces in words reform.

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

A PA word as a preposition, so to speak.

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

PA1 <- (((PA2 / (GA (freemod)?)) __LWbreak)
 ((!(PAUSE) 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.
5/16 guarded against being initial in a predicate.

2.2.6 Articles and quotation constructions

Below, find a block of articles, in the most general sense, including quotation constructions.

```
LE <- (__LWinit ((L EA) / (L EU)
/ (L OE) / (L EE) / (L AA) / (L e)
/ (L o) / (L a)) ((DA1 / TAI0))?
 (PA)? __LWbreak)
```

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

```
LA <- (__LWinit (L a)
((DA1 / TAI0))?
!([ ]+ PA) (PA)? __LWbreak)
```

The name constructor. It also belongs to LE and can be adorned similarly, but there are some specifically LA constructions. The details of this separation are new features (or at least adjustments and clarifications) of this grammar.

5/16 PA suffix component further guarded in a way which I am not sure has any actual grammatical effect.

```
(* LEPO <- (__LWinit ((L e)
/ (L o) / (NI1)+ / RA) PO1 __LWbreak) *)

LEFORPO <- (__LWinit ((L e) / (L o)
/ (NI1)+ / RA) __LWbreak)
```

The abstraction designation constructors. `lepo` is the commonest, but examination of the rule will reveal other forms. The 9/20/2014 modification removes this class. LEFORPO describes the operators which can appear initially to an abstract clause description as of 9/20/2014. It includes some quantifier words as in SUPO. The components of what used to be a LEPO word are now separate words and can for example be separated by a pause.

```
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. Below, the beginning of quotation constructions.

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

affixes (in the true sense) which can qualify the sense of a quotation (as text or sound)

```
LI1 <- (L i)

LU1 <- (L u)

LI <- (__LWinit LI1 !(V2) (Quotemod)?
 ((([,])? ([ ])+))? utterance0 (', ')?
__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.

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

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.

```
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 explicitly at the end of a LAO construction.

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 ([ ] !NI)? Quotemod?
stringnospaces (CII1 ([ ] !NI)? 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`) 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 !(V2) (Quotemod)?
 ((([,])? ([ ])+))? (Name / Predicate
 / (LW (([,] ([ ])+ !([,])) / &(period)
/ !(.) / &((([ ])* 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.

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

```
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 onomatpoeic predicate meaning to make the given sound.

2.2.7 Assorted grammatical particles, somewhat classified

Below, find a block of terminators and boundary markers with various grammatical functions. These are best understood by looking at their use in the grammar.

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

CUI <- (__LWinit (C UI) __LWbreak)

GA2 <- (__LWinit (G a) __LWbreak)

GE <- (__LWinit (G e) __LWbreak)

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

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

GO <- (__LWinit (G o) __LWbreak)

GU <- (__LWinit (G u) __LWbreak)

GUI <- (__LWinit (G UI) __LWbreak)

GUO <- (__LWinit (G UO) __LWbreak)

GUU <- (__LWinit (G UU) __LWbreak)

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

More grammatical particles below. These ones construct subordinate clauses (strictly speaking `je`/`jue` constructions are tightly bound arguments). We also class the `dio` case tags as clause constructors or modifiers.

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

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

`je` and `jue` link arguments very tightly to predicates (or modifiers, according to a new proposal). `je` links first terms after the predicate, `jue` links second and subsequent items.

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

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

The two classes above construct subordinate clauses, the first with terms or predicates, the second with sentences. See the grammar.

```
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)
```

These are the case tags, including the positional ones (and the `lae` and `lue` constructions which are rather different in intent but have the same grammar).

Grammatical particles below used to construct or modify predicates.

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

convert descriptions to predicates.

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

NU <- (__LWinit ((NUO (NI)? (freemod)?))+ __LWbreak)
```

rearrange order of or duplicate arguments (conversion). These can be composed.

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

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

PO <- (__LWinit PO1 __LWbreak)

POSHORT <- (__LWinit POSHORT1 __LWbreak)
```

Construct abstraction predicates from ordinary predicates. POSHORT is inhabited by experimental forms which are always short-scope and cannot be put into LEPO words. See the grammar.

9/20/2014 the long scope PO words are now **always** long-scope, so they are removed from the POSHORT class.

Below find grammatical particles used in the construction of freemods.

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

register markers (indications of attitude)

```
HOI <- (__LWinit (H OI) __LWbreak)
```

the vocative marker

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

the right scare quote

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

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

quotes for forming parenthetical expressions as freemods.

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

a device for forming spoken smilies.

```
UIO <- ((UA / UE / UI / UO / UU
/ OA / OE / OI / OU / IA / II
/ IO / IU / EA / EI / EO / EU
/ AE / AI / AO / AU / AA/ EE/OO/(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)) !(Oddvowel))

NOUI <- ((__LWinit (N o) ([ ])* !(Predicate)
UIO __LWbreak) / (__LWinit UIO N OI __LWbreak))

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

The VV attitudinals (including negative and compound forms). Also the discursive (numeral)`fi` operators. I just dropped NAHU words from this class per John's proposal and will watch for problems.

12/18/2014 AA, EE, OO added.

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

The inverse vocative marker.

The final item in the lexer section is the logical negation operator. Initial NO in certain classes of words and before subordinate clauses must be exclused from this class. 5/15 NO cannot be followed by a space then a KOU word (to avoid confusion with NOKOU words).

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

2.2.8 The two large word classes

Here are classes associated with the large classes of words (names and predicates).

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

DJAN <- (Name / AcronymicName)
```

A string of class DJAN is either a name or an acronym with the same terminal conditions as a name (except for not being consonant final). Note that there is no reason to impose stress rules on these acronyms now that they are names rather than predicates.

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

these are the words with the grammar of the identity predicate `bi`.

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

PREDA <- (([ ])* &(caprule)
(Predicate / LWPREDA
```

```
/ (!([ ]) NI RA)) !((A / ICI / ICA / IGE / I))
 ((',' ([ ])+ &((V2 / A))))? ((!(PAUSE) freemod))?)
```

The class PREDA includes more than predicate words. It includes predicate variables and interrogatives (class LWPREDA) and the numerical predicates. 8/1 the acronymic predicates have been removed. It should be noted that I have done nothing to enforce stress rules on predicates other than predicate words proper. I note that `he` cannot satisfy them as it is a monosyllable! But I should make some provision to require penultimate stress in the other forms.

2.3 Loglan Grammar Proper

2.3.1 Introductory Remarks

This section is edited from the previous official Loglan grammar. There certainly are material changes from the trial.85 grammar, some of which are embodied in proposals already passed, but this level of the grammar is better documented in previous versions. Much of the material here is a direct translation from the BNF grammar of trial.85 to PEG form. Some bugs have been in the past found to lurk in my choice of order in lists of alternatives.

6/9: as with the previous two sections, I have rewritten this completely as comments on a recent text of Loglan.peg, making occasional changes as I went through. The commonest issues were right grouping instead of left and leaving out optional freemods. This version gives a narrative of my understanding of the grammar as it goes through: it is at least the skeleton of the reference grammar in my head. It should be possible to identify the changes I am proposing to make, at any rate; for the most part it is quite conservative.

In general, this document needs a further pass giving examples!

6/9, rewriting the grammar section of the Grand Proposal for readability.

2.3.2 Closing forms

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

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

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

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

guu1 <- ((((PAUSE)? GUU (PAUSE)?) / PAUSE) (freemod)?)
```

Variants of the closing comma GU for closing specific constructions. GUO closes LEPO clauses and now also PO predicates. GUI closes argmods (subordinate clauses). GUE closes sets of links marked with `je/jue`. GUU closes termsets. Each of these may in certain circumstances be represented by a pause. An explicit word of any of these forms consumes a pause that either precedes or follows it (I do not allow a word of one of these kinds followed by a pause to count double, for example).

GUU1 is a special variant of GUU: a modifier consisting of a PA word by itself cannot be closed with `gu`, only with `guu` or pause (a new rule, related to our device for closing APA words and kin).

```
lua <- LUA

geu <- GEU
```

LUA closes explicit sets and lists.

GEU closes predicate constructions beginning with GE. The new form `geu` and the original form `cue` in L1 are both supported.

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

The general grammatical "comma" `gu`, used to close many constructions. It can often be expressed as a pause. An explicit `gu` word consumes any pauses adjacent to it: unlike JCB, I will not allow a `gu`(pause) to count as two `gu`'s: it is quite natural to pause next to such a word.

The special class GUU1 and differences in detail of pause/GU equivalence are new features in this subsection.

2.3.3 Tightly bound terms and modifiers with JE/JUE

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

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

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

jelink <- (JE (freemod)? term)

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

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

`je` followed by a term or modifier is a first argument after a predicate, bound very tightly. `jue` followed by a term or modifier is a second argument or further argument after a predicate, bound very tightly. `jue` links can be conglomerated with logical connectives, both forethought and afterthought. A logical conglomeration of [`je` arguments followed by a sequence of (logically conglomerated) `jue` links] , possibly terminated with GUE, is a general construction of this kind.

The ability to use modifiers as well as terms with `je` and `jue` is a new feature.

2.3.4 Basic predicate constructions

```
predunit1 <- ((SUE / (NU (freemod)? GE (freemod)?
despredE (freemod)? (geu)?) / (NU (freemod)? PREDA)
/ (GE (freemod)? descpred (freemod)? (geu)?)
/ (PO (freemod)? uttAx (guo)?)
/ (PO (freemod)? sentence (guo)?) /
(ME (freemod)? argument (gap)?) / PREDA)
((!(PAUSE) freemod))?)
```

This is a list of predicates which are atomic in a sense. The NU GE and GE clauses are as it were parenthesis constructions which allow somewhat more complicated predicates to be packaged as basic units of predicate constructions. 9/20/2014 added PO sentence predicates closed with GUO to this class.

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

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

Negating a predunit1 one or more times gives a predunit2.

NO2 is a negation which will not be tightly absorbed by a following predunit2 (a negation of an entire predicate rather than of a component of a metaphor).

```
predunit3 <- ((predunit2 linkargs) / predunit2)
```

Attaching tightly bound arguments with JE/JUE to a predunit2 gives a predunit3.

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

A predunit is either a predunit3 or a (short-scope) abstraction from a predunit3.

This is an important stopping point, as this is the kind of predicate which is allowed to appear as a component in a serial name. It is important to notice that such a predicate can have no metaphorical connections (modifications of one predicate by another) except inside a GE construction.

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

A kekpredunit is a (possibly multiply negated) forethought logically connected pair of top level predicates. Notice the reentry of the yet-to-be-defined full complexity of predicates into this construction.

There follows a series of rules developing the class of predicates which can appear in descriptions, which differ from sentence predicates in being permitted to have a forethought construction as an initial element of a metaphor.

Explicit pauses are not permitted between components of metaphors of this kind, because such a pause inside a description will break its end off as a sentence predicate. The ability to pause is controlled by the presence of optional freemods.

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

A despredA is a most tightly bound metaphor construction: it is a series of predunits and kekpredunits bound with the grammatical hyphen CI. Explicit pauses next to CI are permitted here.

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

This rule allows the use of the left boundary marker CUI for grouping of modifiers in a metaphor.

```
despredC <- (despredB)+
```

A chain of despredB's modifying one another in a metaphor, used for grouping after CUI in the previous rule. Understood as grouped to the left. I could theoretically allow explicit pauses in this class, but I do not.

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

Top level logical connection with CA in metaphors in descriptions. Understood as grouped to the left.

```
despredE <- (despredD)+
```

Top level chaining of predicates modifying one another in a metaphor in a description. Note that explicit pauses are not permitted.

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

The top level class of predicates used in descriptions. The final construction is attachment of a top level description predicate as a modifier in final position using GO.

There follows the parallel construction of classes of predicates which can be used in sentences. These differ in not allowing initial forethought connected predicates in a chain of modifications in a metaphor and in permitting explicit pauses in chains of predicates modifying one other.

```
senpred1 <- predunit (freemod? CI !(comma? name) freemod? predunit)*
```

despredA, without the kekpredunits.

```
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)
```

The rest of the construction of the class sentpred precisely parallels the construction of descpred. Note that optional freemods are provided so that one can pause. Note also that tails of these sentence predicates are description predicates as kekpredunits are permitted to appear in positions other than the initial ones. I am uncertain whether this will lead to restrictions on the ability to pause explicitly which will be noticeable to any significant extent in speech or writing; I could write separate classes for tails of sentence predicates that did allow pauses but I will not do it unless it becomes clear that it is actually needed.

The construction of sentence predicates will continue below after the ability to attach sets of terms is set up.

2.3.5 Modifiers (prepositional phrases)

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

mod1 is a very general class of prepositional phrases. The case of a PA2 (tense/location/relation/causal) word by itself is special: it is guarded against being followed by a predicate to avoid confusion with predicate markers and may not be terminated by gu (use guu or pause) to avoid confusion with the tail of an APA word. The latter is a new feature.

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

This class supports repeated negations of forethought connections of modifiers. This is a new feature or a repair of the trial.85 grammar, which had very restricted forethought connections.

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

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

These two classes complete the ability to afterthought connect modifiers. modifier is the top level class of this kind (logical conglomerations of prepositional phrases).

2.3.6 Names and vocatives

```
namemarker <- (([ ])* ((L a) / (H OI)
/ (C i) / (H UE) / (L IU) / (G AO)))

nonamemarkers <- (([ ])* ((!((namemarker
DJAN)) Letter))+ !(Letter))
```

The name markers are the privileged class of words after which a name word may appear without a preceding pause. LA is the name article; HOI and HUE are the vocative and inverse vocative markers; CI is used as a hyphen in serial names; LIU is single word quotation and GAO can form a letteral from a name.

`i` was a name marker for JCB and is not for us. SIA and SIU were name markers for JCB and SIE by extension would be, for good reasons, which could be implemented by in effect allowing these words to be used as vocative markers. I have not done this (one must for the moment say `Sia hoi Djan`) but this looks like a virtuous change (FIX).

nonamemarkers expresses the condition on a name word that there is no occurrence of a name marker in the word in non-initial position whose end starts a shorter name word. This is a weaker condition than the old condition of containing a name marker, but in fact it is all that is needed. (It is a strictly weaker condition because we now impose phonetic shape on names; some occurrences of name markers as in `Uacinton` are clearly harmless: the occurrence of `ci` is followed by `nton` which is not a name word.

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

This is the serial name construction. A serial name is a sequence of items, the first of which is a name word, which may include name words and predunits. Predunit items must be marked with CI (recently approved). Name words which do not satisfy the nonnamemarkers condition must be marked with CI (Appendix H, but modified). Any item following a predunit item must be marked with CI (new). Everything is set up here with the idea that any name word will appear marked either by a name marker or a preceding name, or an explicit initial pause.

Name words in serial names do not have to be separated by explicit commas, though the pauses are still there (and one is permitted to write them). There is only one pause phoneme: the pauses in serial names are not of a shorter variety as in Appendix H.

8/3 acronymic names, like those with false name markers, must be marked with CI if they appear as sutori components of serial names.

```
LA0 <- (L a)

LANAME <- (([ ])* LA0 ((',' ([ ])+))? name (gap)?)
```

Where LA appears followed by something which can be read as a serial name (possibly with an intervening pause), it is read that way. Where this is an unintended reading, perhaps the speaker should have paused somewhere. This is part of the subject of the promised essay on names.

```
HOI0 <- (H OI)

voc <- ((([ ])* HOI0 ((',' ([ ])+))? name (gap)?)
/ (HOI (freemod)? descpred (gap)?)
/ (HOI (freemod)? argument (gap)?) / (HOI (gap)?))
```

The class of vocatives (a sort of free modifier). Notice that HOI energetically looks for serial names as LA does. I am considering adding SIA, SIU, SIE as alternative vocative markers. Could the register markers also be vocative markers?

Note that all vocatives must be marked. A name by itself does not even parse as an utterance.

2.3.7 Arguments and terms

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

Basic sorts of descriptive phrase. As I make this more like a reference grammar, I should give examples of these. I do not know exactly what the GE initial form is for, though I can imagine.

```
arg1 <- ((LEFORPO (freemod)? PO freemod? uttAx (guo)?)
/ (LEFORPO (freemod)? PO freemod? sentence (guo)?)
/ (LIO (freemod)? descpred (gap)?)
 / (LIO (freemod)? term (gap)?) / (LIO (freemod)? mex (gap)?)
/ LAO / LANAME / (descriptn ((!(PAUSE) freemod))? (((((comma)? CI comma?)
/ (comma &(nonamemarkers) !(AcronymicName))) name))? (gap)?) / LIU1 / LIE / LI)
```

More descriptions. Note that LANAME is read in preference to descriptn (la is still in class LE so if what follows it is not a serial name it can be read as a more complex construction). The optional name which can follow a descriptn here is preceded by an explicit pause or a CI marker (required if the name is acronymic or contains a false name marker), completing the solution to the name marker problem (essay promised). No occurrence of a name word occurs in the grammar which is not marked by a name marker word or a preceding explicit pause.

9/20/2014 changed structure of abstraction arguments. It is important to note that the LEFORPO PO sentence GUO construction does not contain a PO sentence GUO predicate as a constituent. If one has a PO sentence GUO predicate as a proper initial component of a predicate from which one wishes to construct a description, prefix it with GE to avoid it being absorbed into an abstract description construction. LEFORPO also contains some quantifier words to allow things like SUPO.

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

More arguments. More attachments of GE, of which I need to contemplate examples. The pronouns get into this class.

```
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 (gap)?))*)
```

Subordinate clauses made from terms using JI words or sentences using JIO words. These can be negated (already possible in LIP but invisible in the grammar: it was hidden in the lexer). These can be linked with afterthought connectives (is there any particular reason we cannot use forethought connectives?). Note that the dedicated word GUI is used to terminate these clauses.

6/10: I am allowing GUI to terminate (JI argument) and (JI modifier) as well. I am convinced that we want this by actually trying to write something.

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

An arg2 is an arg1a possibly with subordinate clauses attached.

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

An arg3 is an arg2 or an arg2 preceded by a quantifier.

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

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

indefinite <- indef2
```

an indef2 or indefinite is a descriptive predicate preceded with a quantifier instead of an article, possibly with subordinate clauses.

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

an arg4 is a chain of arg3's or indefinites linked with ZE (mixed arguments).

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

an arg5 is an arg4 or a general argument forethought connected to an argx (not far below).

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

an arg6 is an arg5 possibly modified by a case tag or the operator IE (interrogative which). Repeated modifications are supported.

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

An argx is a possibly multiply negated arg6.

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

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

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

Afterthought logical linkage of arguments. ACI binds most tightly, followed by the usual A1. The arg8 construction which has possible A1 linkages cannot start with GE since in the next rule it can be linked with AGE. AGE linkage is damaged in trial.85; it is fully capable here (and right grouping rather than left grouping, as it is completely afterthought).

GUU may be used to attach a subordinate clause to a complex argument at the very top level. I am not sure of the effects of this.

Why do LAU constructions appear here – why for example can one not use case tags with them? I have left the whole issue of explicit sets and lists uncommented so far; it needs study.

```
term <- (argument / modifier)

terms <- term (freemod? term)*

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

A term is an item which can be attached to the predicate of a sentence, either an argument of the predicate or a prepositional phrases. The class terms just gives a sequence of terms.

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

words <- (word)+

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

Innards of explicit set and list constructions, which I am leaving uncommented for now.

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

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

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

The termset which can appear after a predicate. Series of arguments and modifiers possibly terminated with GUU, then forethought logically connected, then afterthought logically connected. The final move is to allow a final predicate with GO after a termset which modifies the predicate to which the termset is attached, an odd move but quite usable.

The explicit empty termset (just a GUU) is surprisingly useful.

2.3.8 Advanced predicates

It should be noted that I have made systematic changes in this section, and also that they have much less impact than one might think. I do not draw any distinction between marked and unmarked predicates after markpred and barepred classes themselves, and it is provable (easily) that the further distinctions between marked and unmarked classes in trial.85 are redundant. Further, I privilege ACI connectives completely as a family of connectives binding more tightly than A1 connectives, which means that the name of the backpred class is motivated purely historically.

The more complex changes, which nonetheless probably will never have much effect on what one speaks or writes, have to do with attachment of additional termsets to logically linked predicates with termsets of their own. The rule handling this in LIP is elegant, complete, and entirely impossible to implement in a PEG, because of its degree of left recursion. My original implementation gave exactly the right class of strings (provably) but structured them entirely misleadingly. The solution I give here ought to work for all practical purposes. There are no examples of this kind of construction in the NB3 corpus; I should write some.

Under all ordinary circumstances, this should behave exactly as the core LIP predicate functions do.

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

A forethought connected predicate. I am not sure I follow the reasons for the precise form given. But apparently kekked head modifiers *are* allowed (as of trial.85), with restrictions to prevent the right hand part from coming apart.

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

A sentence predicate with no tense marker followed by a termset.

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

A sentence predicate with a tense marker or an abstraction predicate (over a sentence). PO sentence predicates moved to predunit1 (and closed with GUO) 9/20/2014.

```
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))
```

This is the locus of the differences between my approach and the trial.85 approach. First of all it is more compact (no distinctions drawn between marked and unmarked classes). Secondly, ACI is a fully privileged connective. Thirdly, attaching additonal terms to termsets to connected predicates is handled differently (and supported for ACI as well).

backpred1 is a backpred or markpred, possibly repeatedly negated using a NO which is not consumed by a following predunit2 (so modifying the entire predicate rather than the head modifier).

backpred is a backpred1 or a backpred1 followed possibly by backpred1's linked with ACI connectives followed by a termset (to be attached to all the backpred1s) followed possibly by backpreds linked with ACI connectives followed by a termset. For the shared termset to attach properly may require that empty termsets (just GUU) be attached to the final linked backpred or backpred1.

predicate2 cannot begin with GE (to avoid ambiguity when linked with AGE) and otherwise is constructed by logical linkage with A1 connectives from backpred exactly as backpred is from backpred1.

This is quite baroque and needs a suite of examples to be understood properly. The peculiar approach of adding linked backpred1's first with shared termset then linked backpreds (in the case of building backpreds) is driven by the need to preserve left grouping of logical connectives while allowing some nesting of the construction. An essay is promised. My conjecture is that for all the complications, this is a difference that makes no difference: in practice, shared further termsets will be rare and those that appear should be simple.

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

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

predicate <- (predicate1 / identpred)
```

The construction of top level predicates is completed. predicate2's can be afterthought/right-grouped connected with AGE, with no consideration of shared termsets. Identity predicates (possibly negated) appear as a final alternative.

2.3.9 Sentences

```
gasent <- (((NO1 (freemod)?))*
(PA1 (freemod)? barepred (GA2 (freemod)? terms)?))
```

This is a verb initial sentence (VOS): it is optionally multiply negated, followed by a tense marker, followed by a bare predicate (including final termset), followed by GA2 followed by initial terms. The use of a different class name for ga in this case signals a different function.

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

A statement is a gasent, or a gasent with additional terms before the predicate (OV(O)S(O))), or a predicate with a set of terms before it (SV or SOV). 9/12/2015 modifications to this in order to eliminate imperatives with marked predicates (entered belatedly)

```
keksent <- (((NO1 (freemod)?))*
((KA (freemod)? sentence KI (freemod)? uttA1)
/ (KA (gap)? sentence KI (freemod)? uttA1)
/ (KA (freemod)? headterms (freemod)? sentence KI (freemod)? uttA1)))
```

A forethought linked sentence. The extreme freedom allowed for what the final item is is rather odd (look at class uttA1). The class sentence appears below (sentences built with ICA connectives).

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

A large class of sentences (a solitary predicate is an imperative) to be linked with ICA in the next rule. 9/12/2015: modifiers !gasent predicate captures imperatives with fronted modifiers.

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

sentences linked with afterthought sentence connectives 8/2 fixed missing freemod

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

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

Attach fronted final arguments to a (possibly ICA linked!) sentence to get, among other things, OSV sentences. I have only now realized that the headterms here are potentially a shared termset among sentences linked with sentence level connectives!

Here is a good place to note that I am not happy with the convention that the fronted set of terms must contain the last argument (since some predicates have quite unfamiliar and rarely used last arguments). I suggest allowing use of a positional case tag to set the position of the first term in a headterms set. This is a semantic, not a parser issue.

2.3.10 Utterances

```
freemod <- ((NOUI / UI1 / (SOI (freemod)? descpred (gap)?)
/ DIE / (NO1 DIE) / (KIE utterance0 KIU)
/ (([ ])* (H UE) ((',' ([ ])+))? name (gap)?)
/ (HUE (freemod)? statement (gap)?) / (HUE (freemod)? terms (gap)?)
/ voc / CANCELPAUSE / PAUSE / JO) (freemod)?)
```

The free modifiers. There are a lot of them, which ought to be carefully listed here as this becomes more like a reference grammar. The most important one to note specifically is the explicit pause; it is to control the use of pauses that optional freemods appear in most medial positions in rules, but not in final positions (or in final positions with the pause option excluded) so that gap and related rules have a chance to consume pauses for pause/GU equivalence purposes.

```
uttA <- ((A1 / IE1 / mex) ((!(PAUSE) freemod))?)

uttA1 <- ((sen1 / uttAx / NO1 / links / linkargs
/ argmod / (terms (freemod)? keksent) / terms / uttA) (period)?)
```

Fragmentary utterances, I believe mostly provided as answers to questions. These can disguise parse failures – some parse errors will not cause you not to say anything, but to do something like utter a list of terms instead of a sentence. The complete sentences sen1 are included in uttA1 so it is a more general utterance class.

```
neghead <- (NO1 gap)

uttC <- ((neghead uttC) / uttA1)
```

An uttC is a possibly repeatedly negated uttA1. Notice that the utterance level negation must be followed by `gu` or a pause. Typically, omitting this pause will still negate a sentence, by negating the first argument, which has the same logical effect – but not all sentences begin with an argument, so be careful!

```
uttD <- (uttC ((ICI (freemod)? uttD))*)

uttE <- (uttD ((ICA (freemod)? uttD))*)

uttF <- (uttE ((I (freemod)? uttF))*)
```

These classes are linked by progressively looser classes of connectives. ICI connectives are more tightly binding logical connectives; ICA are the vanilla logical sentence connectives; I are the usual top level utterance separators (but a higher level class IGE appears below!)

```
utterance0 <- !GE ((!(PAUSE) freemod (period)? utterance0)
/ (!(PAUSE) freemod (period)?) / (uttE IGE utterance0)
/ (I (freemod)?) / uttF / (I (freemod)? uttF) / (ICA (freemod)? uttF))

utterance <- !GE ((!(PAUSE) freemod (period)? utterance)
/ (!(PAUSE) freemod (period)? !(.)) / (uttE IGE utterance)
/ (I (freemod)? (period)? !(.)) / (uttF !(.)) / (I (freemod)? uttF !(.))
 / (ICA (freemod)? uttF !(.)))
```

Here are the top level utterance constructions. Utterance0 appears in embedded utterances (as in quotations or parenthetical remarks); class utterance proper is required to end at the end of text. An utterance may not begin with GE (to avoid ambiguity with the IGE connective). They may be freemods or have initial freemods. Otherwise they are built in natural ways which can be read from the rule.

3 List of Subproposals of the Grand Proposal

In this section I am going to try to break down the various suggestions embodied in the three Grand Proposal sections and add reference and examples.

Many actions which I took which might be regarded as changes, I view as making precise what was never made precise, or hidden in the innards of the phonological or lexing components of LIP.

I may also have failed to notice some local items.

Phonology: The phonology section is really a fairly faithful implementation of the phonology in L1 and NB3 (mod my choosing to make things precise which seemed to be underspecified: I do add though that in many cases my precise version of JCB's apparently vague formulations is exactly supported somewhere else; our Founder knew what he was about). There are some proposed changes.

irregular letters removed: `qwx` are no longer Loglan letters.

change in the vowel grouping rules: Streams of vowels in borrowings and names are grouped differently, in the way described un-

der the rule `Nextvowels` above. Vowels in cmapua continue to be grouped in pairs in lockstep.

names are pronounceable: names must be parsed into syllables in a way similar to that used for borrowings. In practice, this appears not to exclude any name appearing in the corpus, as long as the following rule is followed.

syllabic consonants are always doubled: this convention, proposed already in L1, has been adopted.

final consonant restriction: The last two consonants in a syllable cannot be a non-continuant followed by a continuant (as this would essentially force another syllable).

new notation and rules for juncture and stress: New notation is introduced for junctures and stress, and rules are imposed on optional insertion of syllable breaks and stresses. In no case do these affect what words can be parsed (in intention). In particular, syllables cannot cross djifoa boundaries in complex predicates. Multiple options are often supported for placement of junctures. The intention is to support parsing of "phonetic transcripts", defined as strings in which stresses are explicitly shown and whitespace occurs only after explicit comma pauses.

reminder of changes made already since L1: One needs to recall that we have already imposed the rule that predicates with more than six letters cannot start with CVCC or VCCV where the CC is an initial pair, and forbidden CCVV borrowings, as well as forbidding repeated vowels that force stress in borrowings. These rulings have effects in the corpus.

Subproposals in the lexing section:

JUU separates adjacent PA words: JUU is now a break between PA words only, rather than a general substitute for structure word breaks.

Close APA words and kin: APA words are now APAGU or APA(pause); CAPA or ICAPA becomes (I)CAPAGU or (I)CAPA(pause), IPA becomes IPAGU or IPA(pause). A solitary PA word as a modifier must be closed with GUU or pause, not GU (GU would only end

the word). Untensed A words may also be closed with GU which is potentially useful in linking modifiers.

No VCV letterals?: I propose to eliminate the VCV letter names for vowels, though I leave them in for now and add the ZIV and ZIVma replacements I have in mind as alternatives. VCV letterals are the only feature of the grammar which can create Cvv/V junctures in little words, apart from single letter abbreviations of vowels in acronyms, which I have eliminated.

What is the name for y? (ZIY(MA)) qwx need CVV names.

multizero forms: MO for 000 is replaced with MOA. MA for 00 remains the same. These forms can only appear after a digit (this prevents any conflict with the other use of MA). 4/24 improved form of powers of ten (see rule NI1), allowing for example `nemoato` for one million or `tomamoate` for one hundred billion.

math forms: are more or less the same except that acronyms used as dimensions are prefixed with `mue`. PUI is now a break between math words, part of elimination of structure word breaks 4/24. 4/24 improvements to powers of ten.

acronyms: I eliminated single letter abbreviations for vowels in acronyms; abbreviation with a preceding Z is supported. Of the three uses of acronyms, one is eliminated (there are no multiletter pronouns; I am not sure that JCB envisaged these, either), acronymic predicates are replaced by acronymic names (front marked with a name marker or pause, end marked with a pause) and acronymic dimensions in NI words have front marker MUE (actually part of the acronym) and are closed with a mandatory pause.

letteral pronouns: are single letters, possibly with numerical suffixes attached with CI. No pause is required between a series of letters used as pronouns. This is important. Fluent handling of series of single letter pronouns are much more important than supporting a class of multiletter pronouns which would seldom be used and would cause headaches for the more common usage.

removed KAPA, KIPA words: These cause really confusing unintended parses when one attempts to link modifiers logically in the current LIP.

quotation: The LIU single word quotation operator (with NIU as a variant) will quote any phonologically acceptable cmapua, using the NB3 phonetic definition rather than attempting to recognize current cmapua as LIP does. There are serious advantages to this in discussing such things as proposed words. ZIY is allowed as a LWunit, ensuring that LIU ZIY(MA) parses.

There is an entirely new strong quotation proposal. The existing one is not PEG parsable. This will get a separate essay.

Other quotation operators are more or less the same.

RANA words: proposal withdrawn.

(CA)PA words: We give an exact grammatical definition which is similar in concept but gives more words than LIP does. I think that we actually have the same A series words, but more CA series words. Reverting to my original idea that compound PA words are linked with (NO)CA(NOI) terms (in particular, NOI-initial compounds are outlawed).

logical compounds of NI words: Added the ability to logically link quantifiers with (NO)CA(NOI) forms.

LA and LE are somewhat separate again: LA is in class LE but has special uses which are checked for first in constructing names. The name marker restriction no longer exists: names can include name markers, though such may need additional markers or pauses to be used in some contexts. This requires a detailed essay.

cancelling GU pauses: The species of pauses which can replace GU can be cancelled (one can pause accidentally or for effect without unwanted grammar effects) by following the pause with the little word CUU or with `y` followed by a pause.

Subproposals found in the grammar section:

PAUSE: PAUSE is a freemod. freemods in final positions in class definitions are mostly forbidden to be pauses to allow pause/GU equivalence to operate. General modifications are made to ensure that explicit commas can appear whereever a pause is allowed or required, to support the phonetic style of orthography.

linking modifiers with JE and JUE: This earlier proposal is implemented in the provisional grammar.

full use of forethought connectives on modifiers: The rules for use of KA...KI...with modifiers were so restricted as to be almost useless. Full ability to use these connectives with modifiers is now supported.

name marker list extended: There are more name markers (words which can immediately precede a name word without a pause).

serial name repair: CI must precede predunit components in serial names. Proposal 8, approved. There is no longer a special pause phoneme used in serial names. CI must precede a name component which includes a name marker which is followed by a well formed name (Appendix H requires use of CI if the name contains a name marker at all, but the weaker condition stated here is all that is needed) CI precedes name components which follow predunit components (new).

LANAME: LA possibly followed by a comma marked pause followed by a name word is parsed as a name (regardless of ways in which the name segment could be parsed by the grammar); this class is tested for before other description classes. In general, rules involving name are arranged to that all names are marked with either name markers or pauses. In constructions like LE BLANU, DJAN the name must be preceded by an explicit pause.

unmarked vocatives eliminated: This is essential to eliminating the name marker restriction on names. Allowing unmarked vocatives can cause vast parts of sentences to be parsed unintentionally as names. This change is already approved. (It is still possible to make errors in enunciation which can cause unintended parses of large parts of sentences as names; the phonetic parser makes it easier to study when this happens, and I believe that sensible conventions can be stated to avoid this naturally).

forethought logical connection of subordinate clauses?: I didnt change this, but I wonder why we cannot forethought connect argmods. Note that we can negate them (we already could in LIP but this was hidden in the lexer).

argument fixed so that AGE can be chained: This seemed like an error – one could not connect more than two arguments with AGE.

classes markfront/markekpred and barefront/bareekpred eliminated: The distinction between them plays no role in the grammar even in trial.85. Their interactions are now covered by the interaction between the classes backpred and predicate2.

ACI rationalized and termset attachment supported: ACI is now a fully privileged logical connective and termsets may be attached to sets of predicates linked with ACI.

attachment of termsets to A-linked predicates slightly different: The original rule is highly left recursive. I wrote one that is slightly different and that I think is usable in practice and much more PEG friendly. An essay with examples is called for.

in arguments and predicates, classes linked with AGE exclude initial GE: its all in the title. This removes a potential ambiguity, and one would not expect GE groups to be initial anyway.

scope of PO words: The words PO, PU, ZO always have long scope. The words POI, PUU, ZOO always have short scope. The syntactical privileges of PO predicates are normalized (they can modify and be modified for example). There are no longer LEPO words but PO X and LE PO X are separate constructions both closed with GUO (Lojban has a double closure problem in this context). This eliminates the LE, PO problem, though that was not the reason (and not even to my mind sufficient reason) for doing this: the real reason I did it was that the status of PO (sentence) predicates was absurd. In the LE PO X class there is actually a special class LEFORPO of articles allowed, including some quantifiers.

4 Essays (pending)

In this section I have given the headings for the essays I have promised to write.

4.1 On Quotation (pending)

This essay should describe my strong quotation proposal and perhaps talk about other related issues (such as the question of what is a word to be quoted by LIU).

4.2 On Names (zero draft)

Loglan names in this version are required to be resolvable into syllables. Names must be consonant final, with the exception of acronymic names (the former acronymic predicates, the subject of another essay). This reduces the scope of the name marker problem: an occurrence of one of the name marker words in a name with the further property that what follows is a well-fomed name is called a *false name marker*. Phonetic occurrences of name markers which are not false name markers in this sense do not create a problem (as in `Uacinton`).

In this grammar, we are allowing the use of names which contain false name markers. We need to investigate how this is made safe. The general problem is how to determine the left boundary of a name word: the right boundary is always evident, being either end of text or a comma or a space followed by a name or a space followed the word CI, preceded by a consonant (except in the special case of an acronymic name).

As before, a name can occur without preceding pause only after a name marker. But further, the situations in which a name word can appear not following a name marker (possibly with a preceding pause) are quite restricted.

In this grammar, unmarked vocatives have been excluded. The only names which can appear without preceding markers are names which are not acronymic and which do not contain false name markers, and which are immediately preceded by another name (and so by a pause) or by a description (into which they are incorporated, as in `le blanu, Djan`, in which case they are preceded by an explicit pause. In both of these contexts, one also has the option of marking the name with CI instead (and an acronymic or false-name-marker-infested name must be so marked). To disambiguate the latter situation, the occurrences of CI which link predicates cannot precede a phonetic name. This allows `le blanu ci redroladjan` to be parsed correctly as containing the name `redroladjan` (however bizarre this is), while `le blanu ci redro, ladjan` parses correctly as `Le blanu ci redro, la Djan`, which is a pair of terms (the final `ladjan` cannot be a name word so cannot be appended to the description because it contains a false name marker and so must be marked).

Subject to these rules, the left end of a name is as far to the left as possible. A false name marker can be forced to become a true name marker by pausing after it, which is always supported (one can always insert an explicit pause after a name marker in this parser, mod bugs, without affecting

its performance as a name marker), so we can always fix the left end to be in the right place by pausing suitably.

We now discuss the pragmatics of saying what you mean to say. First of all, if one always pauses before a name whether it is marked or not, all problems are avoided. But it doesn't seem that this is necessary.

Secondly, a breathgroup ending in a consonant which contains a name marker but does not begin with a name marker cannot be a name unless it is preceded immediately by a name marker followed by a pause. `Ladjan, clu'valameris` parses correctly as "John loves Mary" because `clu'valameris`, while it *is* a name word, can never appear unmarked. It is quite typical, as here, that a name marker problem is not at all visible in standard orthography: `La Djan, cluva la Meris` is quite clear. It is in phonetic transcripts that these difficulties become visible (audible?)

A strategy which removes many problems is to pause after a vowel as soon as possible after a breathgroup which ends with a consonant: this prevents it absorbing a following breathgroup ending in a consonant, as a further component of a serial name, by accident.

It is always possible to force the beginning of a name to be where you desire it to be, by pausing directly in front of it.

A general warning is that sensible orthography generally parses complex name situations without any difficulty. To really explore the potential difficulties, one should work with a phonetic transcript so that one can see what spaces actually should be comma-marked. It would be handy if the parser would force one to put an an explicit pause somewhere in an interval where one is needed for name marker reasons, and it may be possible to do this.

Although I do not propose this, requiring that all names be penultimately or finally stressed would remove all errors involving peeling a predicate off the front of a name.

4.3 On Acronyms (zero draft)

This essay reflects recent conversations on the Loglan list. John Cowan pointed out that acronyms really ought to be designators not predicates (speaking of the acronymic predicates, not of the dimensions in dimensioned numbers, which are a different problem). I agree with him, but I further stipulate that acronyms should not be pronouns either (they should not live in the same space as the letterals). In fact, the clear solution is that they should be *names*.

Replacing acronymic predicates with acronymic names creates no semantic problems: JCB's `Dainaiza` (or my `mie Dainaiza` of previous versions) becomes `mela Dainaiza`. And JCB's *lo Dainaiza* becomes our *la Dainaiza*. [note that my front marker `mie` for acronymic predicates is now not needed at all; the front marker for dimensions is still needed.] Further, I think that acronyms in English are clearly proper names grammatically (while letters in mathematical English are very closely analogous to Loglan letteral pronouns).

Acronymic names are naturally left and right marked: we require them always to be preceded by a name marker, and always followed by a pause or end of text (they are a species of name word and share most of their properties in this respect).

The reason that acronymns should not be pronouns and should always be clearly left and right marked is that the use of a pause free sequence of letters as a sequence of arguments as in `Mi donsu bai cai` is far more important grammatically than any use of acronyms. A solution which requires pauses between such arguments is damaging a basic central feature of the language in the service of a peripheral feature.

To prevent acronyms of length one from being confused with letterals, we provide an optional front marker `mue` internal to the acronym which is mandatory if there is only one letter in the acronym. This is used in `la Muesai`, sulfur.

The other use of acronyms is as dimensions in NI class words. We require that all dimensions begin with `mue` whatever their length, and that dimensions end with explicit pauses. This prevents any conflicts of acronymic dimensions with letterals or numerals.

Use of `mue` removes problems with distinguishing `todai` (two of D, D being a letteral) from `tomuedai` (two dollars).

On the phonetic level, we propose that the names of the vowels are `ziV(ma)` (MA inducing capitalization) and that these can be abbreviated to `zV` in acronyms but never to just `V`. This prevents the `Cvv-V` situation from occurring in acronyms (and thus from occurring anywhere in my current version of the grammar, though I leave it as a possibility). I seriously deprecate the VCV vowel names.

4.4 On Syllables (pending)

This essay should discuss rules for placement of syllable junctures and their rationale (and how little they matter?)

4.5 On Negation (pending)

This essay should discuss Loglan negation. I don't propose changes; I think the current situation is manageable, but an overview should be given. The semantics of negative arguments should be discussed. The issue of whether we need some other related constructions could also be discussed; Lojban has quite different notions of negation built in.

4.6 On Pauses

A comprehensive essay on uses of pauses, phonetic, freemod and gap (PAUSE/GU equivalence).

4.7 On Linking Termsets

Title is tentative. This is an essay on the very nasty structure of `predicate2`, which I have modified. The collapse of certain distinctions in trial.85 (some were redundant), the regularization of the ACI connectives, and changes to the way termsets are attached to logically linked predicates, with examples. This will be technical.

4.8 On ME

An argument between JCB and myself about `me` (the operator constructing predicates from arguments), not high priority.

5 Appendix: Grammatical Vocabulary

It is a good thing for us to have grammatical terms in our own language.

katpua: B is the predicate/word with meaning F and number of places X in (Loglandical) language Y

cmapua: B is a structure word of grammar class F with meaning/function X in Loglan family language S.

djifoa: V is a combining form or "affix" (deprecated Loglan jargon) of B in context X in language Y

furdjifoa: X is a complex (Loglan grammatical term) containing primitive Y via affix Z in loglandical language W

giurpua: X is a primitive predicate (root word) with meaning Y, number of places Z, in language W

nampua: X is a name word (in L, a consonant final single word) naming Y in language Z

purkle: X is the word class of word Y in language Z

nurcmapua: X is the structure word class of structure word Y in language Z

takfoa: B is an expression/utterance/speech form of grammar class X in language F.

nurtakfoa: X is the grammar class of utterance Y in language Z

purtcu: X is a metaphor (in the sense of Loglan grammar) in which predicate Y modifies predicate Z

Note that **nufu purtcu** means that Y modifies Z in metaphor X

We need more of these. I have in mind nercmapua for unit cmapua for example.