A GLOSSARY OF MACGRAM TERMS

Prepared by JCB.

Actual Parse. The parse produced by the Parser (in the narrow sense, q.v.) in the present machine grammar of Loglan. Often abbreviated as 'AP', the Actual Parse is listed on the 3rd line of the Corpus entries. The AP often contains "inhuman" features like machine lexemes (q.v.), implicit punctuators (q.v.), decompounded CPDs (q.v.), and the numerous parentheses occasioned by these machine-oriented details. So the AP is often difficult to read, and may be thought of as the parse of an utterance in "machine Loglan", q.v. The HAP, or Humanized Actual Parse (q.v.), of a specimen is usually much easier to read and may be thought of as a parse of the corresponding utterance in "human Loglan", q.v.

allogram. The allograms of a grameme are the right-halves of all the grammar rules having that grameme as their left-half; see 'grameme'. Thus, allograms are equally permissible (i.e., grammatical) ways of "expressing" a grameme, that is, of expanding it (q.v.) when generating an utterance, or of reducing it (q.v.) when parsing one. Replacing the expression of one allogram by that of another of the same grameme in an utterance--replacing, say, a noun phrase by a pronoun in an English sentence--will of course often alter the grammatical structure, or parse (q.v.), of that utterance. But it will not change its grammaticality, q.v. Thus, interchanging allograms conserves grammaticality, just as interchanging allolexes conserves grammatical structure; see 'allolex'. (What we have here been calling "allograms" are sometimes called "alternates" by computer scientists, or more usually "the right sides of productions", or simply "right sides".)

allolex. The allolexes of a lexeme are the members of the set of grammatically equivalent words that comprise it; see 'lexeme'. Since all the allolexes of a lexeme are grammatically interchangeable, they are in a certain sense semantic variations within a common "structural meaning". Thus, replacing one connective by another in an English sentence--e.g., 'and' for 'or', when unaccompanied by 'both' or 'either'--will change its specific meaning. But the structural meaning has not changed: a "connection" has still taken place. So the substitution of one allolex for another (of the same lexeme) in an utterance will not alter its "grammatical structure", or parse, q.v. In natural languages words often belong to several lexemes. For example, many English verbs (e.g., 'bank', 'race', 'parse') may also be used as nouns; and vice versa ('dog', 'milk', 'bucket'). But in Loglan, lexemic boundaries do not overlap. So each Loglan word is an allolex of exactly one lexeme. The concept of allolex is parallel on the grammatical level to that of allophone on the phonological level (phonemically equivalent sounds) and to that of allomorph on the morphological level (semantically equivalent phoneme-sequences...for example, the varieties of the English plural morpheme, or the various combining forms of a primitive predicate in Loglan; see 'morpheme'). See 'allogram' for a similarly parallel notion on the level of "grammaticality".

ambiguous. A grammar from which more than one parse-tree (q.v.) can be

constructed for the same utterance is said to be ambiguous. Ambiguous grammars may, however, be "disambiguated" (q.v.) by using special auxiliary rules, principally "precedence rules" (q.v.), which will cause the parsers generated from such grammars always to select exactly one parse-tree for each utterance. In general, ambiguous grammars which have been disambiguated in this way are shorter, easier to understand, and yield parsers that run faster, than grammars of the same language which are unambiguous without such rules. The present grammar of Loglan is of the disambiguated type in that 7 potential "shift/reduce" conflicts (q.v.) exist among its rules. But these conflicts are never realized because the parsers constructed automatically by Yacc (q.v.) always select the "shift" over the "reduce" action in such cases unless instructed otherwise; see 'lookahead' for a description of these actions of a parser. It is this default action by Yacc that constitutes the disambiguating rule. It turns out that in all 7 cases in which this rule is applied, the "shift over reduce" action is exactly the one we want the Parser to take in order to construct a "humanoid" (q.v.) parse.

AP. An acronym for 'Actual Parse', q.v.

arg. An abbreviation of 'argument' (q.v.) used descriptively. '**arg**' is also the name of a grameme in the current grammar, but 'args' is not.

argset. The name of a grameme in the current grammar. The possible expressions (q.v.) of **argset** are (i) a string of one or more arguments following a predicate expression, (ii) such a string plus a following comma **gu**, (iii) a **gu** following an argumentless predicate expression, i.e., one which could take arguments but doesn't, and (iv) nothing at all following just such an argumentless predicate expression. This is not, of course, what we would normally mean by 'argument set' (q.v.) in English; for in describing things in English we do not normally mean to include null instances of them. **argset** and its constituent grameme **argset1** thus comprise a powerful but subtle set of rules for handling commas in Loglan; see p.11 of the Grammar for these rules.

argument. (1) The arguments of an utterance are its designations. (2) The arguments of a predicate (q.v.) are the designations of the one or more "objects" (in the widest possible sense) related by that predicate in some way. 'Argument' thus has the combined force of 'Subject', 'Object' and 'Object of Preposition' in traditional grammar. Both '**argument**' and '**arguments**' are also grameme names.

argument set. A descriptive phrase meaning a string of one or more arguments of the same predicate expression, q.v. When an argument set follows a predicate expression, the string is assumed to be composed of sutori (i.e., second or subsequent) arguments of that predicate and to commence with its second argument. But see 'shifted arguments' for the interpretation of argument sets in other positions. In the current grammar all argument sets are expressions of the **arguments** grameme; but only those following their predicates are expressions of **argset**, q.v.

arguments, shifted. See 'shifted arguments'.

attitudinal. A word that expresses some "attitude" of the speaker toward what he's saying: a certainty; a wish; an injunction; a request; etc. All attitudinals are members of the **UI**-Lexeme.

C, C-. Abbreviations of 'consonant'; used either in morphological formulas, or as a prefix, e.g., as in 'C-final'. But see 'C-conn' below.

C-conn. An abbreviation of 'causal connective'; contrasts with 'L-conn', 'logical connective'.

Carter-vocs. An abbreviation of 'Carter-vocatives'. These vocatives (q.v.) are formed by substituting the vocative operator **loi** for the descriptive operator of ordinary descriptions, and were first suggested by James Carter. Thus **Loi Ganfua go Redro nu Herfa** ('O Lady with the Red Hair!') is a Carter-voc. These odd expressions require very special grammatical treatment by a machine grammar, much of it handled by its Preparser, q.v. Not all of the extensive apparatus required is yet installed.

close-binding. Any operator is said to be close-binding when its scope is limited to the next following predicate word. Thus in **Da pa no preda prede**, the scope of the negative is limited to **preda**; but in **Da no pa preda prede**, the **no** is no longer close-bound because it now shares the long scope of the operator **pa**.

close-bound. An operator in a context in which it is "close-binding", q.v.

complex. An abbreviation of 'complex predicate'. (1) Any Loglan predicate word is said to be complex if and only if it is composed of a string of two or more morphemes (q.v.), or combining forms, at least the last of which is, or is derived from, a primitive predicate. E.g., **mormao** = **mor** + **mao**, in which **mor** is an allomorph of **morto** = 'dead', and **mao** is an allomorph of **madzo** = 'make'. Hence **mormao** is a complex predicate. (2) As a noun, a complex is any of the complex predicates of Loglan.

compound. An abbreviation of 'compound little word'. (1) Any Loglan little word (q.v.) is said to to be compound if it is composed of a string of two or more simple little words; e.g., **nahu** = **na** + **hu** meaning 'at' + 'what (time)?' = 'when?'. (2) As a noun, any of the compound "little" words of Loglan even if very long, e.g., **pacenoinacenoifa**. (3) As a verb, to form such a compound; in particular, the action of the Resolver (q.v.) in identifying a string of LWs in the speech-flow as constituting such a CPD, or, as in "recompounding" (q.v.), the action of the Postparser in reassembling a CPD which has been broken up by the Preparser for the Parser.

conflict-free. A grammar is said (by Yacc, for example) to be conflict-free if, in every state in which its LR parser (q.v.) might find itself, exactly one action is possible; see 'lookahead' for a brief account of the "actions" of an LR parser. The current Trial.19 grammar of Loglan is not conflict-free; but see the entries for 'ambiguous' and 'Yacc'.

conn. In these notes, an abbreviation of 'connective'. Often hyhenated as in 'L-conn' and 'C-conn', the logical and causal connectives.

context-free. A phrase-structure grammar (q.v.) is said to be context-free if all of its grammar rules (q.v.) have left-halves that consist of exactly one grameme, q.v. The current grammar of Loglan is a context-free phrase-structure grammar. In fact nearly all the classes of grammars which are well-understood, and so, useful in computer science,

are context-free. An example of a "context-sensitive" grammar rule is one in which an element C appears in both the left- and the right-halves of the rule, thus providing a "context" for the application of the rule. The computational problems presented by grammars containing such rules are apparently still intractable. Fortunately, the grammar of "machine Loglan" (q.v.) can easily be written in context-free form.

CPD, CPDing/ed. Abbreviations of 'compound' (q.v.), 'compounding' and 'compounded'.

CPX. An abbreviation of 'complex', q.v.

de-localizer. The little word **gi**, an optional left-mark (q.v.) of any modifier. When marked with **gi**, such a modifier is to be taken as an utterance modifier--that is, heard as modifying the utterance as a whole--despite its local grammatical attachment to the immediately preceding word.

decompounding. The action of the Preparser in breaking up sequences of little words which constitute compound words in human Loglan (q.v.) but which must be treated as strings of separate words--actually, as strings of lexemes (q.v.)--by the Parser. An example is **raba**, the universal quantifier. The Preparser breaks up such words; lexes them (q.v.); and hands the Parser a string of lexemes as the representation of each such word; in the case of **raba**, **RA DA**. It would be easy to write a grammar that did not require decompounding. Such a grammar would provide "tracks" for words like **raba** to move through the grammar as "special cases". But such a grammar would be much longer, require many more lexemes, and the added detail would be uninteresting. To decompound some but not all compound words--in particular, not those compound words which, like number-words and compound tenses, behave grammatically exactly like some simple little words--seems the better strategy. All decompounded words are recompounded (q.v.) by the Postparser.

deCPDing/ed. Abbreviations of 'decomounding' and 'decompounded'.

derivation. A word used in computer science to denote a sequence of strings, each composed of gramemes and/or lexemes (q.v.), such that each string (except the first) may be obtained from the preceding string by the application of some grammar rule (q.v.) of some grammar. If the first string is the Initial Grameme (q.v.) of that grammar, and the last string in the sequence is composed entirely of lexemes, then the sequence is said to be a "derivation of that utterance (in lexemic representation) by that grammar". But if we allow the grammar rules to be applied in any order, there will usually be many possible derivations of the longer utterances of any language of reasonable size. Only some of these derivations will be interesting, in the sense of describing possible performances of constructible parsers (q.v.) and/or generators (q.v.) of that language. So we are usually concerned only with selected subsets of derivations; for example, "rightmost derivations", in which the rightmost grameme in the string is always selected for expansion (q.v.); or "leftmost derivations", in which the leftmost is always selected. "LR parsers" (q.v.), for example, always produce rightmost derivations, which accounts for the 'R' in the acronym.

derive. (1) In the morphological context, complex predicates are said to be derived from their defining metaphors, which are always expressed in

the primitive predicates of the language. E.g., **mormao** is derived from **morto madzo** via its constituent affixes (combining forms, or allomorphs) **mor** and **mao**, which are in turn derived from those primitives. Thus **mor** is derived from **mor(to)** and **mao** from **ma(dz)o**. (2) In the grammatical context, some computer scientists use the word 'derive' to describe each step in a grammatical "derivation"; see above. Thus, 'A derives B' is used to describe the replacement of some grameme (or "nonterminal") A by one of its allograms (or "right sides") B during the course of a derivation. We prefer the less puzzling usage 'A expands as B' or 'A is expanded to B' to denote individual steps in a derivation, or in the generation of an utterance, because it contrasts happily with 'B reduces to A' or 'B is reduced to A', which is the standard description of the same step taken in reverse during a parse (q.v.); see 'reduce'.

disambiguate. If a grammar is ambiguous, it is possible that it can be disambiguated--that is to say, made unambiguous in its effects--by supplying its parser (q.v.) with precedence rules, q.v. Yacc was especially designed for the disambiguation of ambiguous grammars through precedence rules; for such grammars are in some practical ways superior to grammars of the same languages which are unambiguous without such rules. The current grammar of Loglan has been disambiguated in this way; see 'ambiguous'.

discursive. A word used in an utterance to call attention to relationships between that utterance and previous utterances in a discourse, e.g., 'on the other hand' and 'in particular' in English. Loglan discursives, like attitudinals (q.v.), are members of the **UI**-Lexeme.

ek. An English word, coined by John Parks-Clifford, denoting the Loglan V-form connectives **a/e/o/u** (excluding **i**) and all their compounds, e.g., **anoi**. As a verb, 'to ek' means to form a connection by using an ek.

expand. In these notes, 'expand' means to replace a grameme (q.v.) by one of its allograms (q.v.) while generating an utterance. Expansion thus contrasts with "reduction" (q.v.), which is the reverse step made in parsing an utterance. Thus, during parsing, allograms are "reduced to" their grammemes; during generation, grammemes are "expanded as (or into)" some selected one of their allograms if they have several. But see 'derive', sense (2), for a synonymous usage in computer science.

expression. In these notes the phrase 'expression of (some grameme)' always refers to the (sub)string of words in an utterance that "corresponds to" (i.e., was generated by or parsed as) that grameme in the course of its generation or parsing. Thus, the phrase **La Djan** in **La Djan, mrenu** is an expression of **argument**; but it is also an expression of **arg**, of **arg2**, and of **arg1**. **Djan**, on the other hand, is an expression of the **name**-grameme only; for **name** is the only grameme to which **Djan** ever corresponds during the course of the parse. **La** expresses no grameme. It is simply an allolex of the **LE**-Lexeme called for by the **LE name** allogram of **arg1**. These examples will be found on pp.11-12 of the Grammar.

FIFO. An acronym for 'first in, first out'; contrasts with 'LIFO', q.v.

free mod, freemod. Abbreviations of 'free modifier', q.v.

free modifier. Any of the set of lexemes **UI/LOI/KIE/DJAN**, but **DJAN** only when not preceded by **LE** (which now includes la), together with any strings of lexemes which they may have "gobbled" (q.v.) during preparsing. These lexemes (together with their gobbled companions) are permissible modifiers of any word in an utterance--including one of themselves--and in that sense are "free to go anywhere". Free modifiers are in turn caused by the Preparser to be gobbled by the immediately preceding lexeme if there is one, and so are never seen by the Parser unless they are initial. In that case the first of any initial string of free mods is taken as a **headmod**. So the free mods are also the allograms of **headmod**, q.v.

front. Used as a verb, 'to front' something means to move it to the front of an utterance. 'Fronted' thus applies to expressions so moved.

GA. The **GA**-Lexeme. When shown in the AP of a Corpus specimen, an "implicit punctuator", q.v.

generator. (1) In these notes, usually short for 'speech generator'. In the broad sense, such a generator is any device--a computer or a human brain--capable of "acquiring" (learning or being programmed with) a grammar of a certain language, together with such other rules and peripheral devices as enable that device to use that grammar (i) to generate a string of lexemes, (ii) to select an appropriate allolex for each lexeme, and (iii) to transform those allolexes (words) into a stream of speech-sounds or printed letters which form an utterance of that language. Corresponds to 'parser', sense (1), but works in the opposite direction. (2) In the narrow sense, that part of such a device that performs step (i), above. (3) In computer science, 'generator' usually means an "automatic parser generator", given an appropriate grammar as input. Yacc (q.v.) is such a parser generator.

gobble. When a lexeme is made to carry a record of certain immediately subsequent lexemes, or, in the case of strong quotation, of a string of foreign words which can neither be lexed (q.v.) nor parsed, and the subsequent lexemes and/or foreign strings are then removed from the utterance to be parsed, the first lexeme is said to have "gobbled" the eliminated material. Gobbling is an action caused by the Preparser (q.v.) in preparing an utterance for the Parser, q.v. The gobbled material may be viewed as the "grammatical noise" (q.v.) in the original utterance. The gobbled words and strings are restored by the Parser to the Actual Parse (q.v.) after parsing and before it is handed to the Postparser, q.v.

grammar. (1) In the broad sense, the word 'grammar' is often used to denote the entire system of rules, including preparsing and postparsing rules, by which utterances are either generated or parsed by the devices which "acquire" these rules. This is the sense of the word, for example, in the phrase 'the current machine grammar of Loglan'. (2) In the narrow sense, a grammar is a set of rules, expressed entirely in terms of grammemes, lexemes and the rewriting sign (see 'grammar rule' below), from which parsers and/or generators can be constructed which are purportedly capable of parsing and/or generating all of the utterances of some language. The Trial.19 Grammar is an early approximation of such a grammar. But the language which the grammar in this narrow sense generates and parses is "machine Loglan", q.v. This is quite a different language from "human Loglan", q.v. So it is only the grammar

in the wide sense--the system of rules composed of Preparser, Parser, and Postparser--that accepts human utterances as input and delivers "humanized parses" as output.

grammar rule. A grammar rule of a context-free (q.v.) phrase-structure (q.v.) grammar, such as Loglan's is, is composed of exactly one element in the left-half, which is always a grameme (q.v.)--or a "nonterminal symbol" as it is called in computer science--followed by the "rewriting sign", which is usually a right-pointing arrow, followed by a right-half composed of one or more elements, which may be gramemes and/or lexemes ("terminal symbols") in any order, which together constitute an allogram (q.v.) of the grameme in the left-half. Rules defining the allograms of the same grameme are generally grouped together, and in that case their common left-half is not repeated but remains implicit.

grammatical structure. In these notes, synonymous with 'parse', q.v.

grammaticality. Formally considered, grammaticality is a property of certain strings but not others made from the words of a language, given some grammar of that language. If a given string parses on that grammar, it is grammatical; if it does not parse, it is not grammatical. It need not, of course, parse "correctly"; see 'parse'. On the other hand, when empirically considered, grammaticality is evidently a property of an utterance for a given listener (or reader). If he can "understand" the utterance in the limited sense of being able to parse it with whatever grammar he has in his head, he will say that it is grammatical. Thus if the listener can arrive at **any** conclusion as to how the utterance was "put together" by the speaker, then it is grammatical...for him. Again, of course, his parse need not be the correct one. He need not have guessed correctly how the speaker actually put the utterance together. And in ambiguous languages--that is to say, natural ones--he will often guess wrongly. (But even if he discovers later that his guess was a wrong one, he is likely to insist that his interpretation, too, was a "grammatical" one.) Moreover, and again like the machine parser, the human listener does not even need to know the meanings of all the individual words he hears in an utterance in order to parse it. It is only necessary that he be able to lex (q.v.) them, again perhaps incorrectly. Thus, 'The groos are fleaner than the glitches' is parsible--and hence grammatical--to our English ears because we can confidently assign 'groo', 'flean' and 'glitch' to English lexemes on the basis of their positions in this sentence. Thus, grammaticality is a weak criterion, easily satisfied between listener and utterance. Yet it is probably because it is only this weak property of allograms that must be conserved in evolution, namely that the new allograms used by innovative speakers yield equally parsible utterances for their listeners when interchanged with old allograms, that the grammars of all human languages have evolved so luxuriantly. See 'allogram' and 'grameme' for related notions.

grameme. A grameme is the common left-half of all the rules in a grammar which have that left-half. The right-halves of the rules which thus "define" a grameme are its allograms (q.v.), and the exhaustive set of allograms for any grameme exhibits all the ways in which that grameme may be expressed (q.v.) in utterances. Thus in the current grammar of Loglan, the grameme named **arg1** may be expressed as a variable, a name, a description, a quotation, or as a **LEPO**-clause (event-description); see p.11 of the Grammar. What this means is that in any utterance in which

argl is expressed in some way---say, by a name--that expression may be replaced by an expression of any other of **argl**'s allograms--say, by a variable--without changing the grammaticality of that utterance. That is, if grammatical, it will remain grammatical; if ungrammatical, it will remain ungrammatical. Thus a grameme may be seen as a set of "grammaticality-conserving" alternatives (its allograms), just as a lexeme (q.v.) is a set of "structure-" or "parse-conserving" alternatives (its allolexes), a morpheme as a set of "meaning-conserving" alternatives (its allomorphs), and a phoneme as a set of "signal-conserving" alternatives (its allophones). The names of grammemes are completely arbitrary. Apparently they exist for the convenience of grammarians only. For while grammemes must have neurological counterparts in human brains--else we could not parse sentences--they do not, apparently, appear in human consciousness. (What we are here calling "grammemes" are what computer scientists concerned with the design of programming languages call "nonterminal grammar symbols", or simply "nonterminals".)

group. (1) As a noun, a group is a non-initial substring of some predicate string (q.v.) which is marked in one of three ways. It is either hyphenated with **ci** (see 'group-hyphen') or its left boundary is marked with **ge**, q.v. If a **ge**-marked group is non-final in a predicate string, its right boundary must also be marked with the group-ender **gue**, q.v. (2) As a verb, 'to group' means to form a group within a predicate string. Groups function grammatically as single predicate words.

group-ender. The little word **gue**, one of the optional punctuators (q.v.) of Loglan; see 'group' for the uses of **gue**.

group-hyphen. The little word **ci**; see 'group'. Hyphenated groups of length three or greater are right-associative, q.v. Unmarked predicate strings, which are of course the commoner forms, are left-associative.

group-starter. The little word **ge**; see 'group'.

GU, GUE. The **GU** and **GUE** Lexemes. When either of these lexeme names appears in the AP of a Corpus specimen, it is a sign of an "implicit punctuator", q.v.

HAP. An acronym for 'Humanized Actual Parse', q.v.

HB-tags. The series of arguments ordinals **pua/pue/pui/puo/puu** which permit one or more arguments of a predicate to be spoken out of normal word-order. The acronym 'HB' is from 'Hixson-Bonewits', a nom de plume of the loglanist who first suggested such ordinals.

head-kekked. A predicate string (q.v.) whose first, or head, predicate words are kekked; see 'kek' and 'kekable'.

head-predas. An abbreviation for 'predicate words at the head of a predicate string'; see 'head-kekked'.

headmod. The name of a grameme in the current grammar whose allolexes are the free modifiers **UI**, **LOI**, **KEK** and **DJAN**; see 'free modifiers'. A **headmod** is thus a free modifier, or a string of free modifiers, at the head of an utterance.

high noise alternative. Whereever the grameme **gap** occurs in the grammer, either of its allograms **GU** or **PAUSE** may be used. It is thought that the spoken comma **gu** will be the alternative that speakers will use in noisy conditions, or with interlocutors (like machines) who are not good at guessing what they mean. This provision of high vs. low noise alternatives for a punctuation grameme reflects what is expected to be the dual use of Loglan in both human-to-human and human-to-machine communication.

human Loglan. This phrase is used in two senses in these notes. (1) It denotes the Loglan that loglanists actually write and speak, what might better be called "current Loglan". But (2) 'human Loglan' is also used to denote the slightly different language in which the specimens in the Corpus are now written. It is this second sense of 'human Loglan' which compares most meaningfully with 'machine Loglan', q.v. For it is this new version of human Loglan, the one which has emerged from our interactions with MacGram, which is of course the version of the human language which is now understood by MacGram.

Humanized Actual Parse. The parse produced by the Postparser (q.v.) in the present machine grammar of Loglan. Often abbreviated as 'HAP', the Humanized Actual Parse is given on the 4th line of Corpus entries. It represents a "humanization" of the Actual Parse produced by the Parser for that specimen. The Postparser performs its humanization by removing any machine lexemes (q.v.) or lexemic pauses (q.v.) which the Preparser may have inserted for the Parser's benefit, by removing any implicit punctuators (q.v.) the Parser may have encountered while parsing the specimen, and by recompounding any CPDs that may have been decompounded (q.v.) by the Preparser. It also removes the surplus parentheses it will always create by these removals and recompoundings.

humanoid. A vague but nevertheless probably deeply meaningful criterion which we attempted to satisfy in the writing of the Target Parses, q.v. As the word suggests, we wanted to construct the parses toward which we meant to steer MacGram in "human-form" ways, that is, in ways that would suggest "clamping routes" for the understanding of the utterances in question that would be immediately meaningful to human heads...on the assumption that there are some basic parsing principles that are "hard-wired" in all human heads. Whether we succeeded in making MacGram's own parses sufficiently humanoid in this or any other sense--for these were the targets that were ultimately hit when MacGram successfully parsed the entire Corpus; see 'Yacc'--is still an open question. If there is a set of "expression-substitution" experiences which will cause human listeners to parse Loglan utterances in these ways, we have succeeded; if not, we have not.

HumGram. An affectionate contraction of "human grammar", the still-unknown actual grammar of "human Loglan" (q.v. sense (2)), an image of which was conjured up in the minds of the MacGram workers during the writing and revision of the Target Parses, q.v. Thus, the MacGram workers not only developed an increasingly firm notion of what the (new) human language should be like, as the project approached conclusion, but also some very strong intuitive notions (no doubt derived from their own parsing and/or generating behavior as loglanists) as to how the specimens submitted to MacGram "should parse". It was this entirely hypothetical grammar which was responsible for the increasingly "humanoid parses", the HAPs, which we managed finally to evoke from

MacGram; and which was, for a time, known to us as HumGram. But to the extent we were successful, HumGram is now functionally equivalent to MacGram. So a separate name for it is no longer quite so useful.

identity interrogative. The little word **hu**, a replaceable interrogative; see 'interrogative' below.

inflector. Any member of the **PA, NU, NO** or **PO** lexemes which occurs immediately before a predicate word or expression is in "inflecting position", and hence an inflector.

Initial Grameme. Every grammar has exactly one grameme (q.v.) with which it starts every generation and ends every (successful) parse; that is its "Initial Grameme". In the current grammar of Loglan, the Initial Grameme is **utterance.** To the Generator (q.v.), the appearance of **utterance** under its scanner must in some sense represent the urge to speak: 'I have something to say.' To the Parser, the appearance of the same grameme under its scanner must in some similar sense mean 'I have understood.'

implicit punctuator. Any of the lexeme labels **GA, GU** or **GUE** which the Parser puts into the Actual Parse (q.v.) at the points where it has found an "error". This error is always occasioned (given a grammatical utterance) by its having found one of the optional punctuator gramemes **ga/gu/gue** in the string to be parsed but no lexeme of the required type--a real **GA, GU** or **GUE**--to match up with it. In these cases it is forced to accept the "error grameme" **err** as the active allogram of the punctuator grameme in question. The Parser tells us that it has encountered such a pseudo-error by inserting the label of the lexeme that **wasn't** there into the parse. This is, of course, completely arbitrary: we might have had it insert asterisks at these points. But the occurrence of these capitalized punctuator words in the Actual Parses allows us to study how the machine grammar is working at these often crucial points in the utterance where punctuation might have been used but wasn't. All such implicit punctuators are, of course, removed from the Humanized Actual Parses by the Postparser, q.v.

interrogative. Any of the interrogative particles of Loglan. There are now three types of these: (i) the identity interrogative **ie**, (ii) the truth interrogative **ei**, and (iii) the replaceable interrogatives **ha/-he/ho/hu. Ha** is used like an ek, i.e., an **a**-type connective, and so requests its own replacement with an ek as an answer; **he** is used like a predicate word, and so requests replacement with a **PREDA** or some other predicate expression; **ho** is used like a number word, and so requests a **NI** or some more complex mathematical expression as an answer; and **hu** is used like an argument, and so requests an argument as an answer. **Ei** behaves grammatically like any other attitudinal and so is a member of the **UI-** Lexeme. **Ha, he** and **ho** are undistinguished members of the **A, PREDA** and **NI-**Lexemes, respectively. **Hu** is not a member of the **DA-**Lexeme only because it is used by the Preparser to identify **nahu** = 'when'-type compounds; but grammatically, the simple word **hu** behaves just like **DA.** So only **ie** has its own special grammar and for that reason, its own lexeme.

inverter. The little word **go**, sometimes called "the inversion operator". **go** occurs optionally inside predicate strings (q.v.) and informs the listener that the part of the string that precedes it is normally (that

is, when spoken without **go**) spoken last, and vice versa; in other words, that the predicate string has been inverted at this point.

kek. An English word, coined by John Parks-Clifford, denoting the Loglan kV-form connectives **ka/ke/ko/ku** and all their compounds, e.g., **kanoi**. As a verb, 'to kek' means to form a connection by using a kek as a prefix and one of the allolexes of **KI** (**ki** or **kinoi**) as an infix between the connected forms.

kekable. The name of a grameme (q.v.) in the current grammar. A **kekable** is a predicate string (q.v.) whose first elements may be, but need not be, a kekked pair (q.v.) of predicate words. Such strings are used only as the operands of descriptions. The **kekable** grameme thus contrasts with the **predstring** (q.v.) grameme which may not have kekked head-predicates and whose expressions function as predicate expressions, q.v.

L-conn. An abbreviation of 'logical connective'; contrasts with 'C-conn', 'causal connective'.

left-associative. A string of three or more elements is left-associative if their parse groups leftmost pairs of them first. Thus, given full parenthesization of a parse, as in the Corpus, the left-parentheses will cluster at the left edge of a left-associated string: e.g., (((A B) C) D), just as right-parentheses will cluster on the right in right-associated strings: (A (B (C D))). Left(Right)-associated strings of similar elements in a parse--names or predicates, for example--are almost always generated by left(right)-recursive rules, q.v.

left-mark. Any mark, like a leading kek (q.v.), that marks the left-boundary of some expression of a grameme in an utterance.

left-recursive. A grammar rule is recursive if the grameme appearing as its left-half occurs also in its right-half. That is to say, the grameme recurs in its own allogram, q.v. A recursive rule is left-recursive if the recurring grameme is the left-most element in its allogram, or if the only elements standing to its left in the allogram are marker lexemes (like **KA**) or machine lexemes, q.v. If a rule is left-recursive, the concatenated strings of elements which will be generated by repeated applications of the rule are also left-associative (q.v.), or "left-grouping". There is some reason to believe that most, and possibly all, optional continuations in human languages are left-associative. On this hypothesis nearly all recursive rules in Loglan grammar have been made left-recursive. Not surprisingly, there are some computational advantages in left-recursion as well.

lex. (1) As a noun, a synonym of 'word', q.v. (2) As a verb, 'to lex' means to assign all the words in an utterance (including compounds) to their lexemes. Lexing is a function performed by the Preparser, q.v. It presupposes the existence of a Resolver (q.v.) which establishes the word-boundaries in an utterance before presenting it to the Preparser.

lexeme. A set of grammatically equivalent words in a language. The test of grammatical equivalence is whether one word may replace another, in all of its occurrences, without changing the "grammatical structure", or parse (q.v.), of any of the utterances in which it may occur. Thus, on the grammatical level the concept of lexeme is parallel to that of phoneme on the phonological level (phonemes are sets of phonologically

equivalent sounds) and to that of morpheme on the morphological level (morphemes are sets of semantically equivalent phoneme-sequences: words or word-parts). The notion that the lexicon of a language may be divided into lexemes is intimated by the "parts of speech" conventions of traditional grammar. But a speech part, say the English Verb, actually consists of numerous similarly behaving lexemes ("types of verbs") which must be kept distinct from one another in any formal account of English. It is a remarkable fact about Loglan not only that its lexemes are very few (around 60) but also that the bulk of its vocabulary, namely **all** its predicates, or all its noun-, adjective- and verb-like words, are genuinely members of just one lexeme: the **PREDA**-Lexeme. What we are here calling "lexemes" are called "terminal grammar symbols", or simply "terminals", by computer scientists.

lexemic. Any feature of an utterance which is necessary for determining the identity of its lexemes; contrasts with 'morphemic', q.v.

lexemic pause. A pause which functions like a word in the parse, and which therefore is a lexeme; contrasts with 'morphemic pause', q.v.

LIFO. An acronym for 'Last In, First out', the "stacking" principle.

link. Either of the two little words **je** or **jue** employed in the construction of linked arguments, q.v.

linked argument. Arguments linked to a single, preceding predicate word by the linking operators **je** or **jue** are called linked arguments. At one time, the privilege of having linked arguments was confined to final predicate words in strings used in descriptions. This restriction has now been lifted. One consequence is that non-final predicate words in any predicate string may now be given linked arguments. So now the "internal specification" of predicate strings (e.g., as in 'faster-than-light ship') may be accomplished with the same grammar rules that attach arguments to descriptions.

LIP. An acronym derived from 'Loglan Interactive Parser'. LIP is a piece of software which embodies the present state of MacGram (q.v.) and which allows the user to examine the parses and parse-trees of freely contrived Loglan utterances, provided they are grammatical on the present grammar, and so to examine the interactions of the grammar rules with the Preparser and Postparser algorithms.

little word. In Loglan, any word which is neither a predicate nor a name is called a "little word" regardless of its length. Simple little words are of the forms V/VV/CV/CVV, and so are genuinely little. But compound little words, also known as compound words or simply compounds (q.v.), are composed of strings of simple little words, and the strings may be of any length. E.g., a number word is composed of CV-form syllables and as many of these may be concatenated in a single word as the speaker can deliver in one breath. Much the same thing is true of tense operators: witness **noipacenoinacenoifa**, which is a bit difficult to deliver pauselessly on the first try. Thus the only practical limit on the length of Loglan compounds is the morphological requirement--necessary for the unique resolution of words by the Resolver (q.v.)--that no word may contain a pause.

long scope. Certain predicate operators, like all members of the **PA**-Lexeme,

have long scope when used as inflectors (q.v.), in that their sense applies to the entire subsequent predicate expression, including any arguments. Other predicate operators, like **no** and the members of the **PO**-Lexeme, are close-binding (q.v.) as inflectors, and so have short scope unless they are followed by a long-scope operator. In that case they, too, acquire the long scope of the long-scope operator.

lookahead. The action of an LR parser (q.v.) when it "looks at" (determines the identity of) the next one or several of the lexemes still to be parsed. Thus, at any instant in the course of a parse, except at the very beginning, the parser will have shifted some leftmost portion of the utterance it is parsing onto a "stack", a region that corresponds in the parsing model to human temporary memory. It is from the elements closest to the "top" of this memory stack that the parser finds the allograms which it replaces with their grammemes, thus "reducing" (q.v.) them, and, usually, the number of elements in the stack, or the load on its temporary memory, as well. Which allogram the parser selects for replacement, or whether it decides not to reduce at all but to shift another lexeme onto the stack, is partly determined by what it finds by its lookahead; that is, by the information given by its scanner about what is coming up. The number of lexemes an LR parser can "look ahead" at in this way is a measure of its lookahead capacity, or what amounts to the size of its "scanning window". At present the size of the scanning windows in efficient LR parsers is severely limited. In the parsers built by Yacc it is limited to one; see 'LR1, LR2, etc.'

LR parser. An LR parser is one which scans the input string from Left to right and produces a "**R**ightmost derivation (q.v.) in reverse". This definition--which apparently produced the acronym 'LR' now so firmly installed in computer science--is a bit paradoxical. For to perform a "rightmost derivation in reverse" is actually to perform a series of reductions (q.v.) in which the **leftmost** allogram capable of reduction is chosen at each step. Reduction is one of the two essential actions of an LR parser, the other being "shifting". The parser shifts a lexeme into its temporary memory when it moves its scanning window one place further to the right along the input string. Reductions then take place on the material stored in memory. The action sequences of an LR parser are sketched in more detail under 'lookahead' above. The most efficient current parsers which can be automatically generated by parser generators like Yacc (q.v.) are apparently of the LR type; and this is in fact the kind of parser Yacc generates when given a suitable grammar. Accordingly, the current Loglan Parser is an LR parser.

LR1, LR2, etc. The number sometimes attached to the acronym 'LR' indicates the number of lexemes which the scanner of the parser in question can accommodate as it moves along the input string. It is therefore a measure of the size of the parser's "scanning window"; see 'lookahead'. The parsers generated by Yacc, and hence the current Loglan Parser, are all LR1. If there were an LR2 parser for Loglan, the number of machine lexemes (q.v.) which now distinguish the human from the machine form of the language (see 'machine Loglan') could immediately be reduced from 9 to 3, and probably 2 of the remaining 3 (M11 and M12; see p.4 of the Grammar) could also be eliminated. But M7 would remain so long as this curiously esoteric feature of the language--provision for predicate variables in prenex (q.v.) quantifiers--were retained. Still, the small size of the "scanning windows" of contemporary computer-generated

parsers is, in a sense, both an artificial and probably a temporary constraint on the design of "speakable languages", q.v. For the human "scanning window" may well be an order of magnitude larger than the best we can model on computers now.

LYCES. An acronym for 'the Loglan Yaccing & Corpus-Eating System'. This is the computer-assisted system of grammar-writing designed and built by MacGram workers for the development of an appropriately "humanoid" (q.v.) machine-grammar of Loglan. LYCES works by measuring the successive "parsing tracks" made by a sequence of trial grammars through a large corpus of specimens of "human Loglan", q.v. The measurements supplied by LYCES are detailed enough so that the user may be certain that each trial grammar kept for further development has shed at least some of the vices but kept all of the virtues of the preceding one. The current grammar of Loglan is the nineteenth "yacced grammar" (see 'yacc') produced in this way at our San Diego Research Center...about seven grammars beyond the one that first parsed the whole corpus correctly. The corpus in question is, of course, the Corpus presented in this Notebook.

LW. An abbreviation of 'little word', q.v.

M-insertion. An abbreviation of the phrase 'insertion of machine lexemes' (q.v.), a function of the Preparser, q.v.

M-lexeme. An abbreviation of 'machine lexeme', q.v.

MacGram. An affectionate abbreviation for 'the machine grammar of Loglan', i.e., the one built with LYCES, q.v. Contrasts with 'HumGram', q.v.

machine lexeme. An artificial lexeme which is created by the Preparser and inserted in the specimen at those points where an LR1 parser (q.v.) would not be able to continue the parse because of the limitation of its lookahead, q.v. In a sense what the Preparser does is report back to the Parser what sorts of lexemes are coming up. The Preparser knows what is coming up because, in this approximation of a human parsing model, it is able to scout ahead of the Parser and send back information in the form of machine lexemes, stationing them as it were in the specimen at the points where the information they provide will be most useful to the Parser when it reaches them. (Currently, there are nine machine lexemes, each one appropriate to a particular type of sequel.) So by the time the Parser does encounter a machine lexeme, at least that local region of the utterance will have been transformed into something that **can** be parsed LR1. These artificial lexemes are called "machine lexemes" because their insertion in a specimen of "human Loglan" (q.v.) helps to convert the specimen into an utterance of "machine Loglan" (q.v.); and every utterance of machine Loglan can be parsed LR1. The machine lexemes are removed by the Postparser after the LR1 Parser's work is done.

machine Loglan: The language in which the Preparsed Strings (q.v.) are written. Machine Loglan differs from human Loglan in four ways: (i) Machine lexemes have been inserted in order to make the utterance parsible LR1; see 'machine lexemes'. (ii) Certain types of compound little words have been decompounded, q.v. (iii) The "grammatical noise" created by various kinds of free modifiers (q.v.) in the human language has been removed. Finally (iv), lexemic pauses (q.v.) have

been identified and inserted in the specimen as if they were words. So in some ways human Loglan (q.v.) has been stripped down to make a language which the machine can read. Yet in other ways the machine forms have been elaborated with strange, redundant signals which the human mind evidently finds unnecessary. Machine Loglan is thus much more like a programming language than human Loglan is. Yet the two versions of Loglan are algorithmically equivalent. Any grammatical utterance in one of them can be reliably translated into a unique utterance in the other. In fact, it is just such acts of swift translation which the Preparser and Postparser now regularly perform in the service of MacGram.

McG. An acronym for 'MacGram', which is in turn an abbreviation of 'machine grammar'; in particular, the machine grammar of Loglan; see 'MacGram'.

metaphorizer. The lexeme **JA.** It is the semantic function of ja and kin to mark the immediately preceding word or phrase as a metaphor. But **JA** is grammatically non-significant. So it is part of the "grammatical noise" (q.v.) gobbled by the Preparser, q.v.

monolexic. Said of lexemes which have exactly one allolex, q.v. Most monolexic lexemes are punctuation words. The negative **NO** and one or two other non-punctuation lexemes are also monolexic.

morpheme. A morphological concept parallel to 'phoneme' at the phonological level, to 'lexeme' at the lexical level, and to 'grameme' at the grammatical level. A morpheme is a set of contextually distributed forms (words or word-parts) all of which have the same verbal meaning or perform the same syntactic function. Thus 'a' and 'an' are semantically equivalent contextual variants of the indefinite article in English; and so, allomorphs of the same morpheme. Most Loglan non-predicate words are monomorphic, that is, have only one form. Nearly all Loglan complex predicates and a large majority of its primitives are polymorphic. That is to say, the primitives have combining forms, some long, some short; and so the complexes (q.v.) derived from such primitives also have variant forms, ranging from short to long, the choice of which may be suited to the context.

morphemic pause. Morphemic pauses are pauses required by the Resolver (q.v.) for the identification of word-boundaries, principally the right-boundaries of names. Unlike lexemic pauses (q.v.) they do not appear in the Preparsed String, q.v. Once the word-boundaries they have helped to establish are known, their work is over. So all signs of them can be dropped from the Preparsed String.

nonterminal. An abbreviation of 'nonterminal grammar symbol' and therefore a synonym of 'grameme', q.v.

optional punctuators. The three optional punctuators of Loglan are **ga/gu/gue,** the same words whose lexemic representations **GA/GU/GUE** appear as "implicit punctuators" (q.v.) in the Actual Parses, q.v. Each punctuator has a restricted set of occasions on which it may or may not be used. But its use or non-use on those occasions will always make a difference in the parse, q.v. So the optional punctuators of Loglan are not often optional in the sense that their presence or absence makes no real difference in the claim of the sentence, like some usages of the English comma. Instead, the punctuators of Loglan nearly always

make a substantial difference. Thus their implicit appearances in the Actual Parses of the Corpus mark the "empty slots" in which they might have been used, and so made a difference had they been used.

parse. (1) As a verb, 'to parse' an utterance is to decide how all its lexemes (q.v.) are to be "clamped together" and in what order. Such a decision may not be correct even if successful, that is, even if it leads to a parse of the whole utterance. The parsing decision is correct if and only if the "parse-tree" (q.v.) describing that decision is formally identical to the parse-tree by which the utterance was generated. Given a disambiguated (q.v.) context-free grammar, such as Loglan now has, and given human speakers and listeners capable of acquiring that grammar, that decision will always be the correct one. (2) As a noun, the word 'parse' in these notes refers to any representation--like a parse-tree or a full parenthesization of the utterance--of the way the lexemes of the utterance were "clamped together" during the parsing operation. Thus in the notes to the Corpus, the word 'parse' always refers either to the Actual Parse (AP), to the Humanized Actual Parse (HAP), or to the Target Parse (TP), all of which are fully parenthesized representations of some form of the utterance. See these entries for more details. Note that this definition does not require that the grameme labels at the nodes of the parse-tree be either known or knowable. They can be entirely missing from the representation of the parse, as indeed they are missing from the parses shown in the Corpus.

parse-tree. A parse-tree, such as that produced by LIP (q.v.), is a branching graph showing how the pieces of the utterance--the "leaves" of the parse-tree, or the lexemic representations of its words--were put together by the parser (q.v.) in such a way that they all lead "downward" to, or can be seen as "growing out of", a single grameme at the "root" of the parse-tree. This root grameme is always the Initial Grameme of the grammar in question. In our case the root of all parse-trees is the grameme **utterance.** Moving upward from root to leaves, one generates the utterance; moving downward from leaves to root, one parses it. If the graph describing the way an utterance was generated is the same graph as the one describing its parsing, then the parse is a correct one. If both the parser and the generator (q.v.) are using the same disambiguated grammar (q.v.) to construct these graphs, there will be only one possible parse of any grammatical utterance. So the two graphs will always be identical.

Parser/parser. (1) In the broad sense, a parser is any device--a computer or a human brain--capable of "acquiring" (learning or being programmed with) a grammar of a certain language, together with such other rules as enable the device to read or listen to utterances of that language, and to report its "understandings" of those utterances in some way. LIP (q.v.) is such a device. The word 'Parser' in the phrase 'Loglan Interactive Parser' suggests that LIP is a parser in this broad sense. (2) In the narrow sense (in these notes, sometimes capitalized), the Parser is that portion of any such device that receives strings of lexemes as input from other portions of it and delivers parse-trees (q.v.) as output to still other portions of it. The Parser that is now part of LIP was generated by Yacc (q.v.) from the Trial.19 Grammar. But LIP also contains both a Preparser and a Postparser, q.v., whose business it is to prepare the string of lexemes for the Parser and to "clean up" the parse-trees it produces to make them more intelligible.

phrase-structure grammar. A grammar that consists entirely of "rewriting rules", of the sort described under the entry for 'grammar rules' (q.v.), but not restricted to rules with single grammemes in their left-halves. Thus, phrase-structure grammars may be either "context-sensitive" or "context-free", q.v.

pointer. The lexeme **LAE** has two allolexes, **lae** and **sae**, each of which forms a "pointer" designation: a description which "points to", and so indirectly designates, something other than the thing described. **Lae** takes designations of signs as operands and refers, thus indirectly, to the thing or things designated by those signs. **Sae** is the inverse of **lae** and so takes designations of anything as operands and refers, thus indirectly, to the sign or signs of those designated things.

Postparser. That part of LIP (q.v.) that receives parse-trees (q.v.) from the Parser (q.v.), removes the machine lexemes (q.v.) from them--as well as all signs of their having been there--reassembles certain compounds that have been broken up by the Preparser (q.v.), removes all the implicit punctuators (q.v.) found by the Parser, restores the free modifiers (q.v.) to their places, and generally "humanizes" the parse. What the Postparser produces is therefore the Humanized Actual Parse, or HAP, which it creates by modifying the Actual Parse, or AP, that was handed it by the Parser.

PP. An acronym for 'Preparser', q.v.

PPS. An acronym for 'Preparsed String', q.v.

precedence rules. Two kinds of conflicts may occur to an LR parser, q.v. One is whether to shift or reduce (see 'lookahead') when reducing is possible. The other is whether to reduce to one grameme or to another, when allograms of both grammemes have become available for reduction. Precedence rules are capable of settling both kinds of conflict. Yacc (q.v.) has a facility for assigning precedence values to any grammar rule and to any lexeme. So to dissolve a shift/reduce conflict in favor of the shift, for example, it is only necessary to assign a high precedence to the lexeme that is to be shifted and a lower precedence to the rule involved in the competing reduction. Reduce/reduce conflicts are dissolved by giving differing precedence values to the competing rules. All 7 conflicts in the Trial.19 grammar are shift/reduce conflicts. And so all are easily dissolved by Yacc's (q.v.) default procedure, which gives automatic precedence to the lexeme to be shifted. See 'lookahead' for an account of the actions of an LR parser.

pred-sign. Any of the set of lexemes which signal an upcoming predicate or predicate expression to the Preparser. The Preparser uses such lists of lexemic signs both to insert machine-lexemes (q.v.), to identify lexemic pauses (q.v.), and to find the right-boundaries of such "gobbled" strings as vocatively-used descriptions ("Carter-vocs", q.v.); see the Preparser Program for all such lists of preparsing signals.

pred-string. An abbreviation of 'predicate string', q.v. This descriptive phrase contrasts with **'predstring'**, which is the name of a grameme; see **'predstring'** below.

pred-string hyphen. The little word **ci**. **Ci** is an infix (or "linking operator") used only in predicate strings to perform grouping operations, q.v. It is thus similar to the English hyphen. Thus **ci** provides an alternative pattern of usages to the parenthesis-like words **ge** and **gue** for forming groups within predicate strings, q.v.

PREDA, preda, pred. '**PREDA**' is the name of the **PREDA**-Lexeme; but all of these i.e. any predicate word.

predexp. The name of a grameme in the current grammar, namely one that includes all "predicate expressions', q.v.

predicate. (1) In the lexical sense, a predicate word. Thus, any of the numerous noun-, verb- or adjective-like words of Loglan; more precisely, any member of its **PREDA**-Lexeme. (2) In the most usual grammatical sense, any string of such words, including those used in descriptions; see 'predicate string'. But there are many other grammatical senses of the word 'predicate'. Thus, of the 62 grammemes in the current grammar, fully 27 of them define 'predicate' in some way! It is, therefore, very difficult to use the single word precisely when talking about Loglan utterances. It is better to use such expressions as 'predicate word', 'predicate string', 'predicate expression' and others suggested in this Glossary.

predicate expression. The main predicate of a declarative sentence ("statement"), with all its sutori arguments, if any, or an imperative sentence taken in its entirety. The phrase thus encompasses all uses of predicate words and strings, plus any argument sets (q.v.) that may be attached to them, except predicate strings used in descriptions. The grameme **predexp** covers the domain of predicate expressions in the current grammar. **kakable** (q.v.) covers those strings used in descriptions.

predicate string. A string of one or more predicate words, together with any of their close-bound (q.v.) operators and any linked arguments (q.v.) of the constituent predicate words. This descriptive concept is very closely related to two grammemes in the current grammar. One of them is **predstring**, which is composed of just such predicate strings but whose initial predicate words may not be kekked (q.v.); these will end up as predicate expressions, q.v. And the other is **kekkable**, whose initial predicate words may be kekked; and these will end up as the operands of descriptions.

predicator. The little word **me**. **Me** combines with the immediately following argument to form a predicate; it is thus a predicate-maker, or "predicator".

predstring. The name of a grameme (q.v.) in the current grammar. A **predstring** is a predicate string whose initial predicate words may not be kekked; see 'predicate string'. Contrasts with **kekable**, q.v.

prenex. An abbreviation of 'prenex quantifier'. These are the fronted sentential quantifiers favored by logicians which in English are usually read 'For any x' (the universal quantifier) and 'There is an x such that' (the existential quantifier). In Loglan these standard prenex forms are very simple: **raba** for the universal and **ba** for the existential. But the grammar of these forms is now fully generalized.

Any argument preceded by a quantifier may now be used as a prenex.

Preparsed String. The specimen after it has been prepared for the Parser by the Preparser. Often abbreviated 'PPS', the Preparsed String is found on the 2nd line of all Corpus entries and may be regarded as an utterance in "machine Loglan", q.v.

Preparser. That part of LIP (q.v.) that receives the specimen and prepares it for the Parser. See 'machine Loglan'. The Preparser Program listed in this Notebook shows most of the things that the Preparser does in preparing the Preparsed String for parsing. (Other preparsing actions are scattered about in other programs of the LIP system that are too long to list here.) Sometimes abbreviated 'PP'.

recompounding. An action of the Postparser, which locates all the compound little words that have been decompounded by the Preparser, and puts them back together again to make "good, human Loglan" (q.v.) for the HAP (the Humanized Actual Parse).

recursive. See 'left-recursive'.

reduce. (1) The opposite of 'expand', q.v. To reduce an allogram (q.v.) is to replace it with its grameme, q.v. Thus reducing is to parsing as expanding (or "deriving", q.v.) is to generating; it is to move downwards in the parse-tree (q.v.) toward the "root" rather than upwards towards the "leaves". (Parse-trees, in this image, are rightside-up, rather than upside-down as computer scientists apparently prefer to grow them.) Reduction is also to move in the opposite direction from that indicated by the conventionally rightward-pointing arrow of the "rewriting sign" in a grammar rule. (2) In a derivative sense, reduction is also usually, but not necessarily, to reduce the load on the temporary memory, or stack, of the parsing device by replacing a string of more numerous elements in it (the allogram) with one (the grameme).

replacing interrogative. An interrogative like **ha/he/ho/hu** which the speaker is asking to be replaced by some word or phrase of its grammatical type; see 'interrogative'.

Resolver. A component of MacGram that hasn't been built yet, namely that part of it that will eventually accept utterances presented to it as uninterrupted strings of characters, with stress, diphthongal vowels, and only obligatory pauses shown, and then resolve such imitations of the speech-stream into words, including the recognition of compound little words, q.v. The result of the Resolver's work will therefore be written specimens of substantially the same form as those presently listed in the Corpus, and so be ready for preparsing by the present Preparser. The rewriting of the resolution algorithm which will perform this work has been deferred until the new morphology of the complex predicate is completed.

right-associative. See 'left-associative'.

right-mark. Any mark, like the comma **gu**, that marks the right-boundary of some expression (q.v.) of a grameme in an utterance.

right-recursive. See 'left-recursive'.

right side. A phrase used by some computer scientists with a sense synonymous with our word 'allogram', q.v.

rule. An abbreviation of 'grammar rule', q.v.

semantic. In these notes, the word 'semantic' denotes the contributions to meaning made by any non-grammatical feature of an utterance. It is thus roughly equivalent to 'non-structural'. For example, when we say that the allolexes of a lexeme differ "semantically but not grammatically", we mean that the differences in their meanings can only come from our knowledge of the "semantics" of the language, not from anything we may learn by parsing them in sentences. Similarly, when we say that all the free modifiers (q.v.), as well as words like the "delocalizer" **gi**, "make semantic contributions only", we mean that they may be removed from or added to any utterance and its structural meaning will not change. (On this view, a question and the corresponding statement in Loglan have essentially the same "structural meaning"; as in this grammar they clearly do.) Thus, the presence of such words makes a kind of "grammatical noise". But it is semantically very rich noise...strangely independent of when it happens to be produced by the speaker in the course his utterance.

shek. An English word, coined by John Parks-Clifford, denoting the Loglan cV-form connectives **ca/ce/co/cu** and all their compounds, e.g., **noca**. As a verb, 'to shek' means to form a connection by using a shek.

shift. (1) An action of the parser (q.v.) when it moves its scanning-window one more place to the right during its left-to-right scan of an input string. (2) Having done so, the single lexeme that was previously under the scanner--of an LR1 (q.v.) parser, that is; but just the leftmost lexeme in case the scanning-window accommodates more than one--is also said to have been "shifted to the top of the stack"; that is, it becomes the most recent entry in the temporary memory of the device. See 'lookahead' and 'LR parser' for more details of the parser model.

shifted arguments. Any rightmost portion of the complete set of sutori arguments of a predicate expression may be spoken (or written) first, then marked with the shift-mark **guu**, and then followed by that predicate expression with or without a first argument in normal position, and then, optionally, by any leftmost portion of the sutori set that has not already been spoken. Shifted arguments thus make it possible to omit medial members of the argument set, q.v.

short-scope. Close-bound operators (q.v.) are said to have short scope.

simple word. An abbreviation of 'simple little word'; see 'little word'.

speakable language. Used in a sense roughly equivalent to 'human language' to include Loglan with the natural languages, but also to distinguish it from the "formal languages"--the programming languages and the various mathematical and logical notations--which are like human languages in some important structural ways but are not speakable. Yet Loglan is like these formal languages in having been designed; and now, like some of them, in being machine-parsible. At present, Loglan is an odd and, it may be, the only member of both sets.

strong quotation. Quotation with **lie** and a pair of arbitrarily chosen identical terminators between which any string of representable speech sounds, including foreign ones, may be quoted. Thus the strings quoted with **lie** are not offered to the Parser for parsing, but are instead gobbled (q.v.), along with the two terminators, into the lexeme **LIE** before presentation to the Parser.

superset. Any set of which a second set is a proper subset is a "superset" of the second set. That is to say, it has all its members and some besides. Many features of the new "human Loglan" developed through interaction with MacGram produce supersets of corresponding utterance domains in old Loglan. For example, the sutori arguments of a predicate expression may now be linked arguments of its final predicate word. So **La Djan, farfu je la Djek** is now grammatical. This sentence was not grammatical as recently as 1980.

sutori. A Loglan word, imported into English by loglanists, which means 'at least second', and hence 'second or subsequent'. For example, one can speak of the 2nd through the final arguments of a predicate as its "sutori arguments", a usage that appears frequently in these notes.

Target Parse. The parse shown between #-signs on the 5th line of the Corpus entries. These were our sometimes-altered target conceptions of what constituted "humanoid parses" of the Corpus specimens. It was these parses which we tried to make the HAPs approach during the final stages of our work with MacGram; see LYCES. But often, of course, we would be nudged by MacGram into awareness that the HAP it was then producing for a given specimen was actually a better parse of that specimen than the one we had previously adopted as its TP. When this happened, our conception of the HumGram (q.v.) changed; and we adopted these "Good HAPs" as the Target Parses for the next pass through the Corpus.

tense operator. Any member of the **PA**-Lexeme when in inflecting position, q.v.

ternary. A structure with three elements; contrasts with 'binary'. For example, any infix--e.g., **CI** or **A**--produces ternary structures.

terminal. An abbreviation of the phrase 'terminal grammar symbol' as used by computer scientists, and therefore a synonym of our word 'lexeme', q.v.

TO. An acronym for 'Tense Operator', q.v.

TP. An acronym for 'Target Parse', q.v.

truth interrogative. The little word **ei** = 'Is it true that...?' **Ei** has the grammar of any other attitudinal, and so is just another member of the vast **UI**-Lexeme; see 'interrogative' and 'semantic' for more details.

V, V-. Abbreviations of 'vowel'; used either in morphological formulas, or as a prefix, e.g., as in 'V-final'.

voc. An abbreviation of 'vocative', q.v.

vocative. Any feature of an utterance used to call the attention of a specific hearer or hearers; also, as an address of an intended recipient.

word. From the standpoint of a grammar, a "word" is any allolex of any of its

lexemes; that is, it is any lex. But our grammar (in the narrow sense) is a grammar of "machine Loglan" (q.v.), not of "human Loglan". And there are many compound words in human Loglan, e.g., the universal quantifier **raba**, which are so unified in meaning, so like contrasting simple words in function (e.g., **raba** alternates with **ba**, the existential quantifier, in quantifying sentences), that the grammatical operation that originally formed them is in a sense lost to our minds. So for us to write such words as strings of separate words, as the grammar now insists we do for it, would be to do violence to our human sense of the language. But this raises an interesting question. When does what was once a phrase become a word? When did 'never the less' become the blurted 'nevertheless' in English? The answer probably is, When it was used so frequently as an alternative to single words that it acquired the lexemic label of those single-word alternatives in the minds of English speakers and listeners. If this is so, **raba** and kin ought probably to be given their own lexeme...even at the expense of complicating the grammar. But we resist this move at this early stage of our macgrammatical explorations.

Yacc, yacc. An acronym for 'Yet Another Compiler-Compiler'; a noun when capitalized, and in these notes, a verb when entirely in lower case. Yacc is an automatic LR1 parser generator which takes context-free phrase-structure grammars as input. It was especially designed to make possible the disambiguation of ambiguous grammars by the use of "precedence rules" (q.v.), and was published by Stephen C. Johnson and his associates at Bell Labs in 1975. Yacc uses the "parsing tables" it generates to find any grammar conflicts and to exhibit them to the user, to declare the submitted grammar to be "conflict-free" (q.v.) when it finds none, and to disambiguate certain types of conflicts automatically when these exist. If only these types of conflicts exist in the input grammar, Yacc will still generate a parser for it after disambiguating it; and of course it will do so for any grammar it finds conflict-free. Yacc is the centerpiece of LYCES, q.v. All the trial grammars used in developing the current MacGram were accepted as such only if they "yacced". That is to say, if, when submitted to Yacc, a grammar was found either to be conflict-free--all the trial grammars prior to Trial.12 were of this class--or to have only "benign conflicts" of the sort Yacc could disambiguate--Trial.12 and all subsequent grammars are of this second class--then the Yacc-generated parser for that grammar was sent on to parse the Corpus. If a grammar didn't yacc in either of these two senses--both of which produce unambiguous parsers--it was back to the drawing-board until it did. The difference between the Trial.11 and Trial.12 grammars was that Trial.12 finally cured what had seemed the incurable problem of redundant **gu**'s. It did this by removing **gu** from two allograms in the Trial.11 grammar that seemed to be the fountainheads of all the "extra **gu**-ing". And this made the grammar ambiguous. The redundant **gu**'s were mostly gone, it turned out; but 7 shift/reduce conflicts had also appeared. But Yacc's automatic dissolution of these 7 conflicts, by giving precedence to the shift action in each case, produced a grammar that parsed the Corpus perfectly. Not only were there no redundant **gu**'s, but all the virtues of former grammars were preserved as well. It was the first grammar to do so. (All subsequent explorations have been "off the Corpus".) So Johnson's 1975 remark that these grammars are often "better grammars" has certainly been borne out in this case.

JCB, 15 Jun 82