Electronic Grammars: Taking advantage of the possibilities

Mike Maxwell
CASL, University of Maryland, USA
Electronic Grammaticography Workshop
February 2011

...and a team: Evelyn Browne, Claudia Brugman, Thomas Conners, Anne David, Melissa Fox, Amalia Gnanadesikan, Nathanael Lynn, Michael Marlo, Corey Miller, Tristan Purvis, Paul Rodrigues, Jessica Shamoo, Alina Twist, Tamara Wehmeir

UNIVERSITY OF MARYLAND
CASL
CENTER FOR ADVANCED STUDY OF LANGUAGE

What was created thousands of years ago and is still useful today?

Assumption

- Language is one of the most complex things we learn, and grammar is a large part of that complexity.

Some values

- Grammatical description is a valu*able* form of language documentation.
- Grammars should be understand*able* (at least by other linguists!).
- Grammars should be reli*able*; in conjunction with lexicons, they should generate all and only the relevant structures of the language.
- Research should be reproduc*ible*, and therefore grammars should be test*able* by anyone (not just reported as accurate by the original researchers).
- Grammars (like lexicons and texts) should be archived, and should therefore be archiv*able*
- Grammars should be multi-purpos*able*:
  - For understanding texts
  - For language learning (in some sense)
  - For computational text analysis
  - For understanding related languages
  - For theorizing about language (i.e. for linguistics)
- And maybe grammars should be enjoy*able*!

Descriptive Grammars

- We all know what they are... but how good are they?
- From an email (from me to Jonathan Amith, 2008):

  '17 Neki.doc' [= a grammar document] says:

  Neki can be suffixed to a verbal base in the 3sgS...

  the tense/aspect/mood inflection is on the verb neki

  Does this mean that you never get plurals of this construction? It can't be used to say "We/You(pl)/They want to eat"? Or does it get used with plurals, but the -s affix doesn't agree with the plurality?
- Yawelmani Yokuts

  "At least three major publications in generative phonology describe the Yawelmani dialect of Yokuts as one in which an underlying contrast between /u:/ and /o:/ is neutralized... However, there are clear counterexamples to the general rule of long-vowel lowering... Yawelmani can no longer be used as a prime example of abstract underlying phonemes which are never realized on the surface." Juliette Blevins, "A Reconsideration of Yokuts Vowels" (2004)
- Examples could be multiplied many fold...

Is this really a problem?

- If languages (and grammars) really are the most complex thing we learn in our lives, then it's no wonder we can't debug them in our heads.
- "Reliance on grammars is not sufficient to give completely accurate data... Descriptive grammars rarely exemplify derivational patterns in any detail, rarely say what is productive and what is not, rarely tell you whether the examples they provide are regular or not, and typically provide lexicalized examples. In other words they do not usually provide precisely the kind of information that the morphologist would like to know about." --Laurie Bauer, "An overview of morphological universals" (2010)
- Can we do better?

Computational Grammars

- "Men will set the goals and supply the motivations, of course, at least in the early years. They will formulate hypotheses. They will ask questions. They will think of mechanisms, procedures, and models... The information-processing equipment, for its part, will convert hypotheses into testable models and then test the models against data which the human operator may designate... The equipment will answer questions. It will simulate the mechanisms and models, carry out the procedures, and display the results to the operator." J.C.R. Licklider "Man-Computer Symbiosis" (1960)
- Parsers and generators (e.g. morphological transducers) do this today
  - ...but for specific models of morphology/ phonology
  - ...and most linguists don't know how to use them.

Problems with computational grammars

- Oapan Nahuatl, Xerox xfst notation:

```
define iEpenthesis3 [[..] -> i || m Nulls _ %- Nulls Nasal];
   !This apparently only affects kim- 3plO- on verbs and i:m- 3plPossr- on nouns.
   ! Bled by nim2ni (an allomorphy rule), preventing it from applying to
   ! nim- 2plS- prefix…
! Reduplication:
define CopyInitC [[Cons1 ->  b   || .#. _ Vowel1 [h | ":"] %= b]                .o.
                 [Cons1 -> {ch} || .#. _ Vowel1 [h | ":"] %= {ch}]
                .o.
                 [Cons1 ->  d   || .#. _ Vowel1 [h | ":"] %= d]   !in Spanish loans
                .o.
                 [Cons1 ->  f   || .#. _ Vowel1 [h | ":"] %= f]   !in Spanish loans
                 ...
```

- Problem 1: The formalism is baroque.
- Problem 2: Lifetime of computational grammars is limited.
- Problem 3: There is no direct connection between this computational grammar of Nahuatl and the descriptive grammar. Comments can refer to the grammar, but:
  - Misspelled affixes or glosses in comments can't be checked, so they can't be found.
  - References to section numbers can break if the descriptive grammar is edited.
  - The descriptive grammar does not refer back to the computational grammar.

Is the lifetime really a problem?

- Case study: Hermit Crab parser/ generator
  - Written 1992-1995 as plug-in to SIL's LinguaLinks
  - 1995: Microsoft C goes from 16-bit compiler to 32-bit compiler, some adjustments to code necessary
  - 1995: Arity Prolog goes from 16-bit compiler to 32-bit compiler, non-backwards compatible
  - 1995: Digitalk Smalltalk bought out by ParcPlace (incompatible versions; Digitalk's Smalltalk version later bought by Cincom, but licensing is unclear)
  - Implication: Grammars written for Hermit Crab are endangered
  - (Postscript: SIL re-wrote Hermit Crab in C, but data format is different)
- What about emulators?
  - If an emulator is good enough for parsing, it should be good enough for lexical databases, and we don't need XML!

The shelf-life problem

- The shelf-life problem is caused by the short shelf-life of parsers, and the differences in programming languages among parsers:
  - Oapan Nahuatl, Xerox xfst notation:

```
define iEpenthesis3 [[..] -> i || m Nulls _ %- Nulls Nasal];
  !This apparently only affects kim- 3plO- on verbs and i:m- 3plPossr- on nouns.
  ! Bled by nim2ni (an allomorphy rule), preventing it from applying to
  ! nim- 2plS- prefix.
```

    Same rule in Stuttgart sfst notation:

```
$iEpenthesis3Inverted$ = ((i:<>) _-> (m $Nulls$ __ "-" $Nulls$ $Nasal$))
$iEpenthesis3$ = ^_$iEpenthesis3Inverted$
  %This apparently only affects kim- 3plO- on verbs and i:m- 3plPossr- on nouns.
  % …
```

- ...and that's just between two finite state transducers. What if next-generation morphological parsers have a different (better!) treatment of morphosyntactic features?
- So the *real* lifetime problem is not the parsing engine's shelf-life, but the lifetime of the parsing engine's programming language.

Solving the shelf-life problem

- An independent and stable formalism is the solution.
  - It must provide a reasonable notation for linguistic constructs.
  - It must allow for observational adequacy across a wide typology of languages (but not necessarily descriptive adequacy, much less explanatory adequacy).
  - It must "play" with other standard formalisms (ISO LMF for lexicons, TEI/ISO for features).
  - It must be understandable (at least with effort), i.e. self-documenting insofar as possible (not APL).
  - It must be documented.
  - ...and obviously, it must be in Unicode.
- But how to convert this into a parsing engine's programming language?

Our notation

- Based on a linguistic model (expressed in UML); will be (better) documented
- Implemented in XML
- Intentionally, the model does not use the latest linguistic theories (moving target problem):
  - No Optimality Theory
  - No distinct prosodic level
  - No autosegmental phonology
  - Not even any phonological features! (just phonemes)
  - ...but extensible.
- Cf. Bruce Hayes "Introductory Phonology" (2008):
  "…one also wants to be able to describe phonologies in a way that is accessible to human inspection… using phonetic symbols when they lead to no harm."
- Observational adequacy and understandability is the goal.
  - But: under-/over-application of phonological rules in reduplication?

Morphology coverage

- Inflectional and derivational morphology
- Morphosyntactic features, exception features (e.g. for stem allomorphy)
- Prefixes, suffixes, infixes...; listed irregular (suppletive) forms
- Templates for inflectional affixes (but inflection can also be modeled without templates)
- Phonologically conditioned allomorphs (roots, stems, affixes)
- Inflection (conjugation, declension) classes
- Reduplication*
  C V (C) X
  1 2 3 4 → 1 a 3 1 2 3 4
- ...and other kinds of process affixes*
- Compounding, incorporation*

**Converter for parser is incomplete for these items*

Phonology coverage

- Phonemes (and graphemes, boundary markers)
- Natural (or unnatural!) classes defined as sets of phonemes
- Phonological rules
  - Including epenthesis and deletion rules
  - Rules apply in linear sequence
  - Rule strata can be defined*
  - Rules can apply simultaneously, or left-to-right or right-to-left iteratively
  - Environments definable as regular expressions over phonemes, graphemes, boundary markers, natural classes, and strings
  - Rules can be restricted to a particular part of speech
  - Rules can require an exception feature

**Converter for parser is incomplete for these items*

Conversion to parser

- Steps:
  - 1. Combine the formal grammar XML fragments scattered throughout the descriptive grammar into a single XML file. (*easy*)
  - 2. Parse this file into a network of linked objects representing the formal grammar. (*done, won't change*)
  - 3. Starting at the "root" of this network, output the objects in the parsing engine's formalism and required order, using standardized file names for lexicon files for each part of speech. (*messy, will change with a new parsing engine*)
  - 4. Extract the relevant parts of lexical entries into the standardized files. (*dictionary- and grammar-dependent, but usually not too complex; possible complications with conjugation/ declension class marking*)
  - 5. Run the parsing engine's compiler to convert the grammar (in the parsing engine's formalism) + lexicon files into the parser's internal representation. (*easy*)
- Implemented in Python.

Revisiting the problems with computational grammars:

- Problem 1: Opacity of formalism: solved
- Problem 2: Lifetime of formalism: solved
- Problem 3: Connection between descriptive and formal grammar?

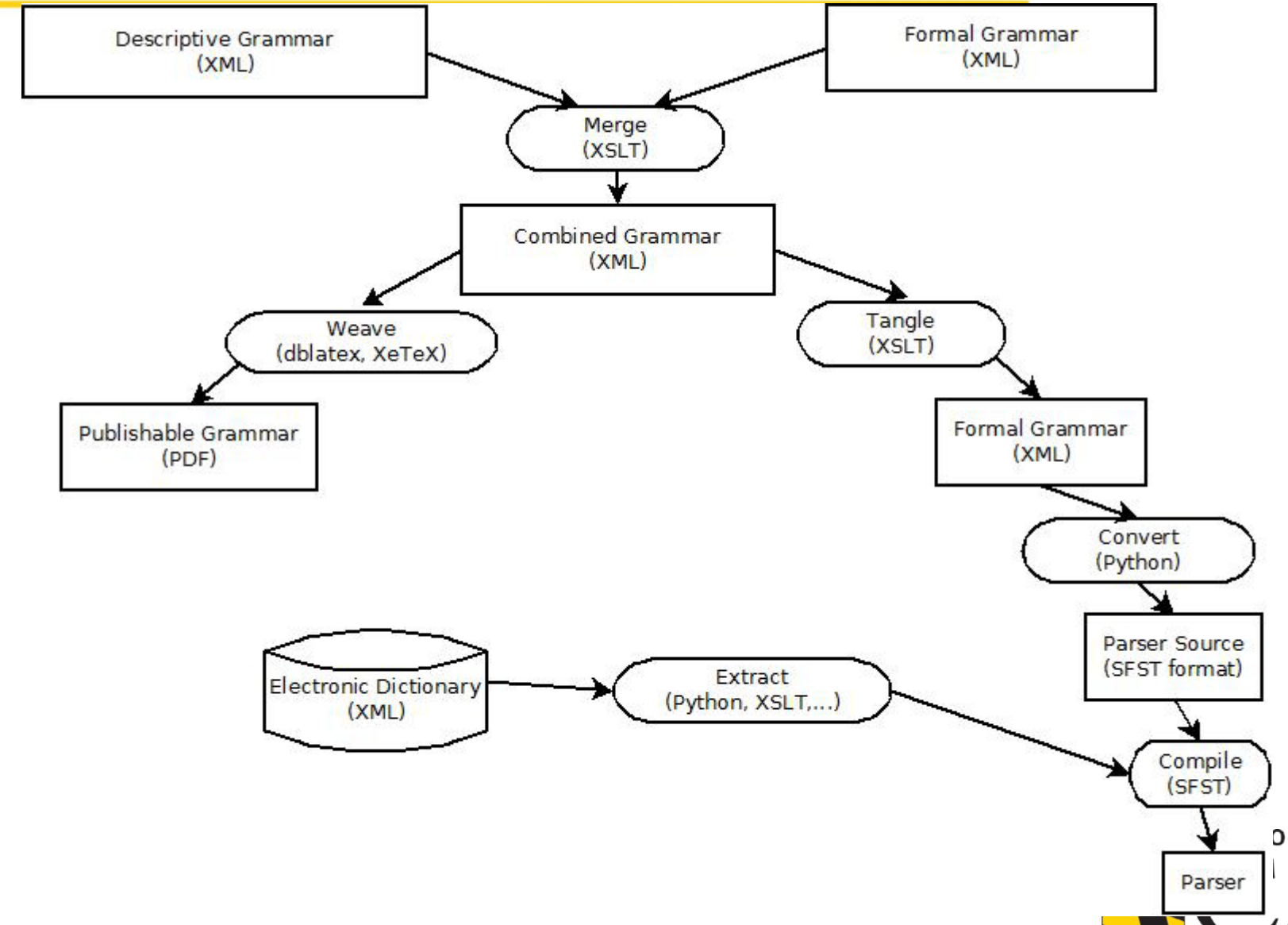
Descriptive vs. Computational Grammars

	Descriptive Grammars	Computational Grammars
Understandable by humans	Yes	No
Processable by computers	No	Yes
Unambiguous	No	Yes
Contains inline examples and/or interlinear text	Yes	No (except as comments)
Contains paradigm tables	Yes	No (except as comments)
Contains test data (inline examples, interlinear text, paradigms,...)	Yes	No
Testable	No	Yes
Debuggable	No	Yes (given test data and tools)
Contains alternative analyses	Yes	No (except as commented-out code)

Can descriptive and computational grammars be married?

- Combine descriptive and formal grammars into a single document
- Literate Programming (Donald Knuth, Stanford)
  "A traditional computer program consists of a text file containing program code. Scattered in amongst the program code are comments which describe the various parts of the code... In literate programming the emphasis is reversed. Instead of writing code containing documentation, the literate programmer writes documentation containing code." --Ross Williams "Funnel Web Tutorial Manual" (2000)
  - Order of exposition need not be the same as the order for computing.
  - Not a big hit among computer programmers; why?
    - Programmers hate to document (but linguists...)
    - Computer programs are long and getting longer (but grammars...)

The Process



Descriptive Grammar (XML)

```
both, as can be seen in <xref linkend="T_PersonalEndings" />.
</para>
<table xml:id="T_PersonalEndings">
  <title>Personal Endings</title>
  <tgroup align="center" cols="4">
    <colspec colname="_1" colnum="1" colwidth="0.3*" />
    <colspec colname="_2" colnum="2" colwidth="0.3*" />
    <colspec align="center" colname="_3" colnum="3" colwidth="1.0*" />
    <colspec align="center" colname="_4" colnum="4" colwidth="1.0*" />
    <thead>
      <row>
        <entry align="center" nameend="_2" namest="_1"></entry>
        <entry align="center"> Singular</entry>
        <entry align="center">Plural </entry>
      </row>
    </thead>
    <tbody valign="middle">
      <row>
        <entry align="left" nameend="_2" namest="_1">
          <emphasis role="bold">1st</emphasis>
        </entry>
        <entry>
          <ilr:inlineExample>
            <ilr:langData lang="pus" script="Arabic">ـم</ilr:langData>
            <ilr:langData lang="pus" script="IPA">-əm</ilr:langData>
            <ilr:gloss />
          </ilr:inlineExample>
          <ilr:inlineExample dialect="SW,M">
            <ilr:langData lang="pus" script="Arabic">ـمه</ilr:langData>
            <ilr:langData lang="pus" script="IPA">-əm(a)</ilr:langData>
            <ilr:gloss />
          </ilr:inlineExample>
```

A more user-friendly view...

2.2 Personal suffixes

The personal suffixes in T_PersonalEndings are those that reflect the categories of person, number, and gender; for example, *second person plural* or *third person feminine singular*. We refer to them as PNG for short.

Verbs; personal endings Person; of verbs PNG suffixes can be attached to the bases described in sxn_RootClasses to form finite verbs, with some exceptions to be described below. The spelling of these endings is common to both present and past tenses in the first and second person; however, present tense verbs are stem-stressed while past-tense verbs are end-stressed. Stress; word; verbs On the other hand, the third person suffixes differ depending on tense: the present tense does not distinguish gender or number in the third person, Gender; lack of gender distinction; in present tense Number; lack of number distinction; in present tense Tense; present; lack of gender distinction in Tense; present; lack of number distinction in while the past tense third person endings distinguish both, as can be seen in T_PersonalEndings.

Personal Endings

colspec colspec colspec colspec

		Singular	Plural
1st		م -əm مه -əm(a) (SW,M)	و -u ی -i (M)
2nd		ی -e	ئی -əy است -āst (verb bases ending in ی y (SW)) وی -oy (M)
3rd - present	M	ی -i	
	F		
3rd - past	M	ه -ə ئی -əy (SW)	ه -ə ∅
	F	ه -a	ی -e

As seen in XMLMind

The formal grammar

```
<src:fragment xml:id  = 'fragVerbPersonSuffixes'
              xmlns:src = 'http://casl.umd.edu/LiterateProgramming'
              xmlns:Ln  = 'http://casl.umd.edu/LinguisticSchemas/Language.rng'>
<Ln:InflectionalAffix gloss='-1Sg' id='afVerb1Sg'>
    <Ln:AffixAllomorph>
        <!-- -əm (all dialects) -->
        <Ln:Form spelling='+əm' script='xlit'/>
        <Ln:Form spelling='+م' script='Arabic'/>
    </Ln:AffixAllomorph>
    <Ln:AffixAllomorph dialect="Western">
        <!-- -əma (Western dialects) -->
        <Ln:Form spelling='+əma' script='xlit'/>
        <Ln:Form spelling='+ما' script='Arabic'/>
    </Ln:AffixAllomorph>
    <Ln:fs description='Inflectional Features'>
        <Ln:f name='Person'>
            <Ln:symbol value='first'/>
        </Ln:f>
        <Ln:f name='Number'>
            <Ln:symbol value='Sg'/>
        </Ln:f>
    </Ln:fs>
</Ln:InflectionalAffix>
<Ln:InflectionalAffix gloss='-1Pl' id='afVerb1Pl'>
```

Admittedly, not real user-friendly...

…a little more user-friendly:

```
<src:fragment xml:id="fragExPhonologicalRule1">
<Ln:PhonologicalRule>
    <Ln:subrules>
        <Ln:PhonologicalSubrule>
            <Ln:input><Ln:refPhoneme idref="phB" DisplayName="b"/></Ln:input>
            <Ln:output><Ln:refPhoneme idref="phP" DisplayName="p"/></Ln:output>
        </Ln:PhonologicalSubrule>
        <Ln:PhonologicalSubrule>
            <Ln:input><Ln:refPhoneme idref="phD" DisplayName="d"/></Ln:input>
            <Ln:output><Ln:refPhoneme idref="phT" DisplayName="t"/></Ln:output>
        </Ln:PhonologicalSubrule>
        <Ln:PhonologicalSubrule>
            <Ln:input><Ln:refPhoneme idref="phG" DisplayName="g"/></Ln:input>
            <Ln:output><Ln:refPhoneme idref="phK" DisplayName="k"/></Ln:output>
        </Ln:PhonologicalSubrule>
    </Ln:subrules>
    <Ln:environments>
        <Ln:Environment wordFinal="true"/>
    </Ln:environments>
</Ln:PhonologicalRule>
</src:fragment>
```

…and as seen in XMLmind:

```
▽ XXXProgram fragment 'fragExPhonologicalRule1'
b --> p   d --> t   g --> k   / _#
```

...typeset ("woven"):

		Singular	Plural
1st		ـم -əm ـمه -əm(a) SW,M	ـو -u ـي -i M
2nd		ـي -e	ـئ -əy ـاست -āst (verb bases ending in ى y SW) ـوى -oy M
3rd - present	M	ـي -i	
	F		
3rd - past	M	ـه -ə ـئ -əy SW	ـه -ə ∅
	F	ـه -a	ـي -e

Table 7.2: Personal Endings

The formal grammar of person suffixes appears below:

```
<Ln:InflectionalAffix gloss='-1Sg' id='afVerb1Sg'>
  <Ln:AffixAllomorph>
    <!-- -əm–all dialects -->
    <Ln:Form spelling='+əm' script='xlit'/>
    <Ln:Form spelling=م+'' script='Arabic'/>
  </Ln:AffixAllomorph>
  <Ln:AffixAllomorph dialect="Western">
    <!-- -əma (Western dialects) -->
    <Ln:Form spelling='+əma' script='xlit'/>
    <Ln:Form spelling=ەم+'' script='Arabic'/>
  </Ln:AffixAllomorph>
  <Ln:fs description='Inflectional Features'>
    <Ln:f name='Person'>
      <Ln:symbol value='first'/>
```

Using dblatex and XeTeX

As a "yapa," the grammar provides parser test data

- Every inline example (including paradigm tables) and interlinear example is a test case.
  - To the extent that the descriptive grammar is complete, we have a complete set of test cases for every paradigm and declension class.
  - If we include a morpheme breakdown in the examples, we even know what the correct parse is.
  - An unparsed example may be an error in the grammar, a word not in the dictionary, or a misspelling or other mistake in the example.
- This gives us *reproducible results*: finally, other linguists can verify that our grammar really does what it claims!
- Of course corpora have additional test data: common misspellings, non-standard spellings, dialectal variants, loan words, proper names, place names, numbers, words not in the dictionary... maybe even constructions we missed in the grammar!

Now and Future

- We have written two complete (descriptive + formal) morphological grammars, for Bangla and Urdu. The parsers have been built and tested.
- We have a complete descriptive grammar and partial formal grammar for Pashto.
- We are beginning work on Dhivehi and Punjabi.
- To-do:
  - Finish parser converter (e.g. reduplication).
  - Add lexicons of words used in examples of descriptive grammars.
  - Publish formal grammar documentation, so other linguists can use it. (Ask me!)
  - Try unrelated languages (Javanese? Arabic?)
  - Develop mechanism to display formal grammar as linguistic objects (inflectional templates → tables of affixes, phonological rules in standard linguistic format,...).
  - Develop better tools to edit the formal grammar.
  - Syntax?

The values again

- Grammatical description is a valu*able* form of language documentation
- Grammars should be understand*able* (at least by other linguists!)
- Grammars should be reli*able*; in conjunction with lexicons, they should generate all and only the relevant structures of the language
- Research should be reproduc*ible*, and therefore grammars should be test*able* by anyone (not just reported as accurate by the original researchers)
- Grammars (like lexicons and texts) should be archived, and should therefore be archiv*able*
- Grammars should be multi-purpos*able*:
  - For understanding texts
  - For language learning (in some sense)
  - For computational text analysis
  - For understanding related languages
  - For theorizing about language
- And ~~maybe~~ grammars <u>should</u> be enjoy*able*!

From millennia ago:

...and also from thousands of years ago:

Will your grammars be relevant millennia from now?

UNIVERSITY OF MARYLAND
CASL
CENTER FOR ADVANCED STUDY OF LANGUAGE