3

The SIL FieldWorks Language Explorer Approach to Morphological Parsing

H. ANDREW BLACK AND GARY F. SIMONS, SIL INTERNATIONAL

Natural language parsing can contribute to improving our knowledge of less-studied languages as the rigor involved in building a formal description allows the linguist's hypotheses to be tested and validated against actual data. However, linguistic field workers who are working to describe less-studied languages typically have different skills and needs than computational linguists. Thus, they do not have a good track record for taking advantage of parsers like those used by computational linguists. This paper addresses the parsing requirements of an ordinary working linguist and then describes a morphological parser we have developed to meet those requirements. The resulting parser is integrated into Language Explorer, the lexicon and text component of the SIL FieldWorks suite of tools which is designed to be used by a field linguist on a Windows laptop.

1 Why Another Parser?

For decades, SIL International has been analyzing and describing many of the world's less-studied languages. Back in the 1970s, SIL was aware of the potential that parsers offer for aiding linguistic analysis (Grimes 1975, Weber and Mann 1979, Weber and Mann 1981). In the 1980s we developed full-featured morphological parsing tools that were deployed in the field—

Texas Linguistics Society 10: Computational Linguistics for Less-Studied Languages.
Nicholas Gaylord, Stephen Hilderbrand, Heeyoung Lyu, Alexis Palmer and Elias Ponvert eds.

AMPLE (Weber, Black, and McConnel 1988; based on Weber and Mann 1979), and PC-KIMMO (Antworth 1990; based on Karttunen 1983). A syntactic parser was developed in the 1990s—PC-PATR (McConnel 1995; based on Shieber 1986). The results of these tools, however, were ultimately disappointing because the up-take was small. Our ordinary working linguists (henceforth OWLs) in the field found them too complicated and too abstract to deal with. Though AMPLE was conceived as "a morphological parser for linguistic exploration," it never lived up to its name because OWLs could not negotiate the learning curve to use it. A key problem was that the "theory" of morphology reflected by the features of the parser was not like what they had learned when they first learned morphology—it involved a lot of new computational concepts. Another problem was that it was one more program to learn. Most OWLs were already at their limit by the time they had learned a word processor and a lexicon management tool.

By contrast, another of our tools, Shoebox (Davis and Wimbish 1993), took a simplistic approach to morphological parsing and enjoyed almost universal adoption. It uses a limited set of pattern matching facilities to tokenize wordforms into morph sequences. It is very easy to use and get started, but runs out of steam with a complex morphology.

In both cases, whether the full-powered parsers that were too hard to use or the easy-to-use parser that did not have enough power, we observed another problem. The instructions given to the computer to teach it how to do parsing were not in a format or notation that was familiar to linguists. This meant that our experienced linguistic consultants could not review the OWL's work in order to give advice on improving the analysis. It also meant that the grammatical knowledge learned in the process was not in a form that could be published to share with others interested in learning about the language.

With the backdrop of these experiences, we launched (in the late 1990s) a project to develop a morphological parser that would address all of these shortcomings. The basic requirements were that it should:

- be fully integrated into lexicon management and interlinear text analysis so that users do not have to learn one more program;
- use an underlying model of morphology that is already familiar to linguists;
- be as easy to get started as Shoebox;
- result in a human-readable grammar sketch as well as a machine-interpretable parser;
- reuse our existing parsing software where practical.

This paper presents our solution to meeting these five requirements within a new product named the SIL FieldWorks Language Explorer. Each of the sections which follow deals with one of the requirements and de-

scribes the approach we have taken to meeting it. Each section also illustrates the approach through a screen shot or other sample.

2 An Integrated Approach

The first requirement is that the parser must be integrated with the other tools our users will use for the bulk of their work on language analysis and description. We do not want the user to have to learn yet another program just to do parsing. Nor do we want them to be concerned with the conversion and synchronization problems inherent in trying to use the same lexicon and texts with different programs.

The next generation of our tools for supporting field work is the SIL FieldWorks suite of programs. SIL FieldWorks is described on its home page (http://www.sil.org/computing/fieldworks/) as "a suite of software tools to help language teams manage language and cultural data, with support for complex scripts." The FieldWorks Language Explorer is the lexicon and text component of SIL FieldWorks. As described on its home page (http://www.sil.org/computing/fieldworks/flex/), "It can help you: elicit and record lexical information, create dictionaries, interlinearize texts, and study morphology."[1]

The parser is integrated into the Language Explorer component of FieldWorks; it uses information that has been entered into the lexicon to make suggested analyses for wordforms when interlinearizing text. Conversely, as new morphemes are encountered in analyzing texts, they are entered into the lexicon. The integration of components is based on the use of a single underlying database. That is, the data used by all tools are stored in a common relational database which is fully normalized.

Figure 1 shows a text being interlinearized. An analysis with a clear background (see *dia)* has been approved by the user for this occurrence in the text. An aqua background (see *Ivan)* indicates an analysis the user has seen and approved elsewhere, but which has not yet been confirmed for this occurrence.[2] A tan background (see *tikompensarovah)* indicates an analysis proposed by the parser,[3] but which the user has never approved. The words

[1] The tool also has facilities for importing Shoebox data and has an XML archival form for the entire database. One may also export interlinear text to a number of different formats, including XML, HTML and Open Office.

[2] Such approved analyses may have originally been suggested by the parser or the user may have parsed the word manually. If the word has more than one analysis associated with it, then the analysis shown with aqua background is the analysis that the user has approved most often in the past. One of the other analyses may be selected via the context menu.

[3] The parser may actually propose several analyses. Only one is shown. The user does have the ability to see the other analyses proposed by the parser and select one of them.

with strings of asterisks below them are ones which the parser is not yet able to analyze.

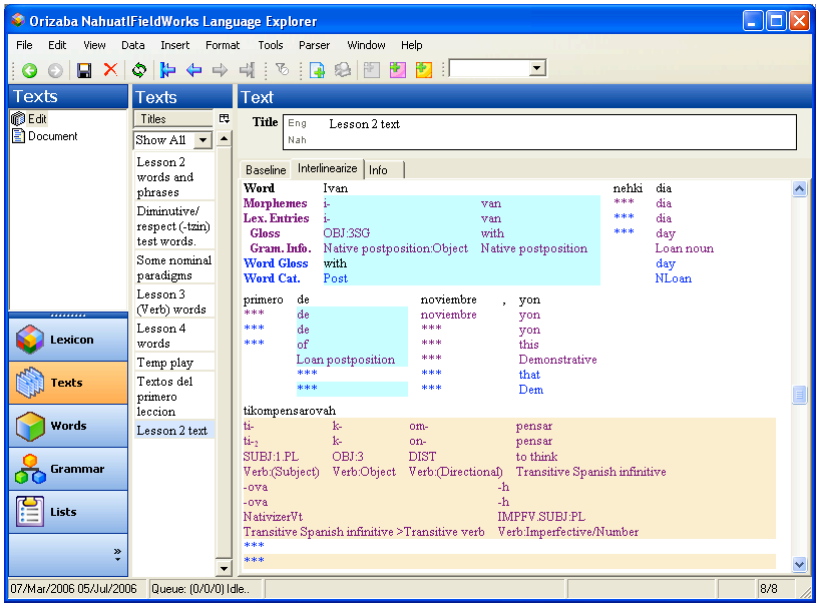

Figure 1. Integration of parsing with interlinear text analysis

The next screen shot in Figure 2 shows how a user may add a lexical entry from the Interlinear tool. In this example, clicking on the unanalyzed word *noviembre* allows the user to pop up the dialog box for entering lexical information that the parser will henceforth be able to use.

By including our parser in a tool within the FieldWorks suite, we have thus met our first requirement that the parser be integrated into a tool already being used for other functions.

3 A Conceptual-modeling Approach

The second requirement was to use notions and notations that are already familiar to linguists. By contrast, our earlier parsers used arcane notations and notions. To overcome this, we based our model of morphology on the concepts taught in the textbook that was most commonly used by our training schools (Bickford 1998). We built those concepts into a conceptual model of the objects in the problem domain, their attributes, and the rela-

tionships between them.[4] That model was formalized first in terms of the

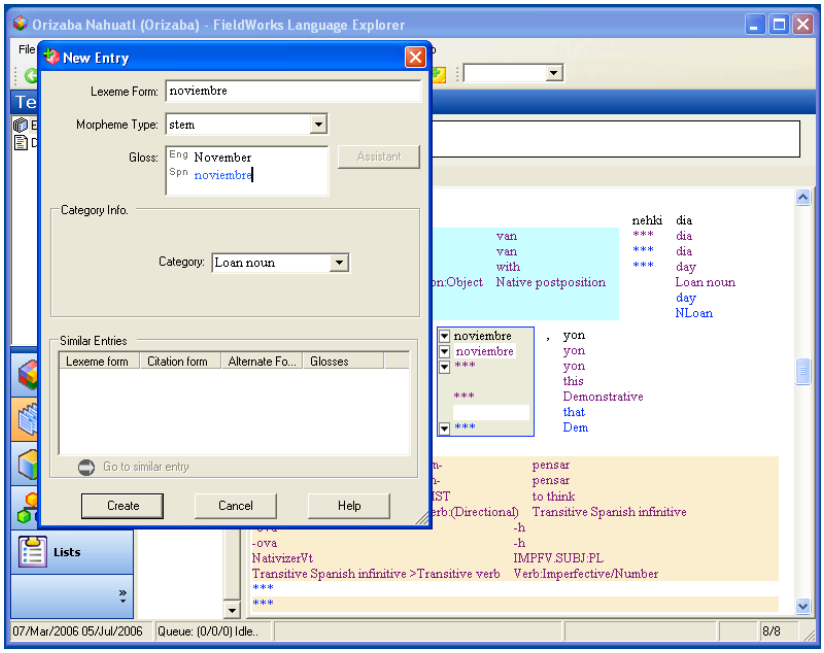

Figure 2. Adding a lexical entry during text analysis

CELLAR conceptual modeling language (Simons 2000) and more recently in terms of the Unified Modeling Language (Fowler 2003). That formal model then serves as input to a transformation process that automatically creates a SQL database that implements the model (Hayashi and Hatton 2001). Using this approach, the task of describing a language is that of creating records in the database to document the instances of the conceptual objects that are discovered in the language.

These are the major morphological concepts we model:

- A feature system built with typed feature structures containing simple features with atomic values and complex features with embedded feature structures of a specified type as values. The system includes a feature catalog based on morphosyntactic properties taken from the online GOLD ontology (which can be found at this

[4] The original document delimiting and justifying our conceptual model was written by Mike Maxwell (Maxwell 2001).

URL: http://www.linguistics-ontology.org/gold.html).[5] Users select the features needed to describe the language from this catalog or add new ones when necessary.

- A category (part of speech) hierarchy. The system is initialized with a category catalog based on GOLD. Each category or subcategory may have:
  - Inflectional templates, including prefixing and/or suffixing slots. Each slot can have one or more inflectional affixes. Slots can be shared by multiple templates within the same category.
  - Inflection classes (also known as conjugation or declension classes).
  - Inflection features, bearable features, and exception features.
- Lexical entries for roots, stems, particles, and clitics.
- Lexical entries for affixes which come in three flavors:
  - Inflectional affixes. These are associated with one or more slots that appear in inflectional templates.
  - Derivational affixes. These map from one category to another. A derivational affix may change the inflection class or other features of the resulting stem. Other features may also be used to constrain where the derivation may apply.
  - Unclassified affixes. When a user does not yet know whether an affix is inflectional or derivational, it is marked as unclassified. The parser treats such affixes as relatively unconstrained.
- Compounding rules. These are for word-internal stem compounding. One can also use them to model things like noun incorporation.

The phonology component uses the classic item-and-arrangement approach (Hockett 1954). Thus, the user must list the various allomorphs a given entry may have. Here are the major phonological concepts we model:

- Phonemes
- Natural classes of phonemes
- Allomorph environments. Each allomorph in an entry may have one or more environments delimiting the segmental environment in which the allomorph is licit.

In addition, we have the capability to model full and partial reduplication and infixation.

The full conceptual model contains over 100 object classes. UML diagrams and prose definitions of the objects and attributes may be down-

[5] The current feature catalog is based on an early version of GOLD. We plan to update it when GOLD begins to stabilize (Simons and Hughes 2006).

loaded at http://Fieldworks.sil.org/ModelDoc/ModelDocumentation.chm.[6] By using this conceptual model-based approach, we have met the requirement that the parser use notions and notations that are already familiar to linguists.

Figure 3 is a sample from our UML modeling diagrams that includes the inflectional template portion. It indicates, for instance, that a syntactic category (PartOfSpeech) may define inflectional templates (MoInflAffixTemplate) and inflectional slots (MoInflAffixSlot), and that instances of the former reference instances of the latter. Figure 4 is a screen shot illustrating the interface for creating and updating instances of MoInflAffixTemplate.

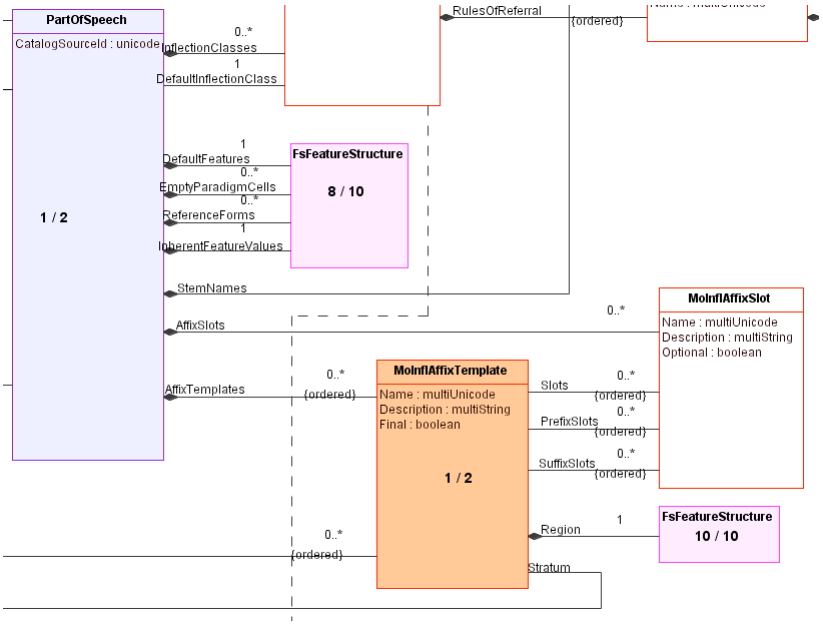

Figure 3. Conceptual model of inflectional templates

4 A Stealth-to-Wealth Approach

The third requirement is that the parser should be as easy to get started with as the Shoebox parser, yet, at the same time, the user needs to be able to do a complete job of modeling all aspects of the morphology of a given language. That is, the idea here is to enable a researcher to start quickly and yet be able to finish well. We have called this notion, "stealth-to-wealth."

[6] This documentation is a "Compiled HTML Help" file and opens with Microsoft's HTML Help application.

This approach allows linguistically aware users to begin at the level where they are and to use the tool to help them successively improve their analysis and the resulting model of it. That is, the user gradually tells the system what s/he knows about the grammar, receiving as a reward increasingly automated analysis of text.

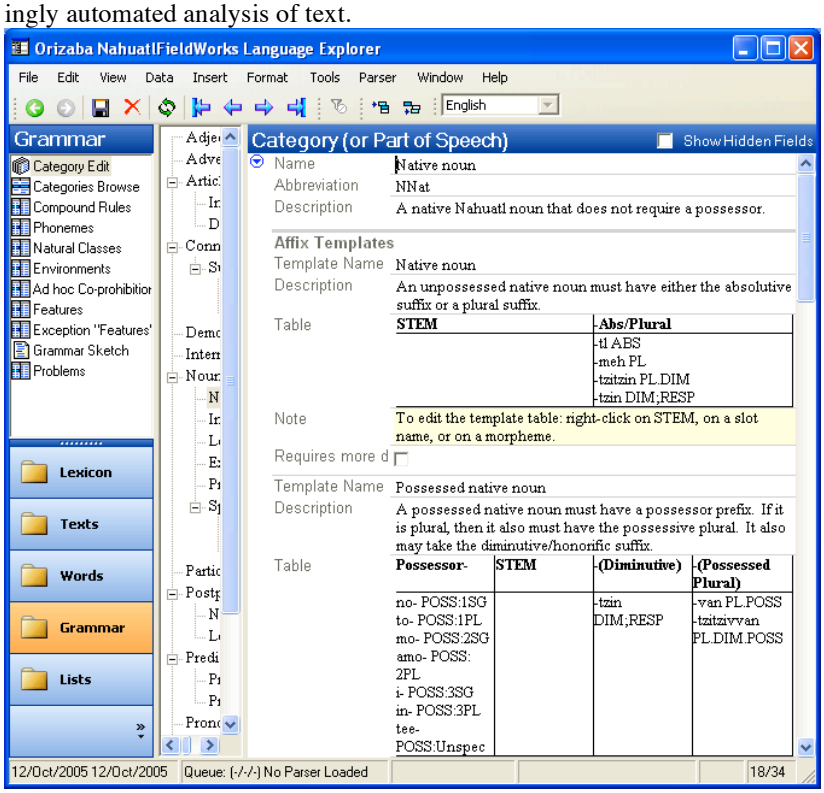

Figure 4. User interface for entering inflectional templates

"Stealth" refers to the parser hiding behind the interlinearizer and lexicon and parsing with no user setup at all. As the user performs word analysis by hand in the interlinearizer tool, the system creates minimal lexical entries behind the scenes. The parser then uses this information to offer up possible parses on new words encountered in texts. In the early stages, the parser knows just allomorph forms and the only constraint on where they may occur is given by their classification as prefix, root, suffix, and the like. At first this gives satisfying results as the parser is able to propose correct analyses using roots and affixes already encountered in the text. But as more allomorphs are added to the lexicon, the parser begins coming up with

many incorrect parses as it finds more and more combinations of allomorph strings that will cover the wordform. At some point users become frustrated enough by the incorrect parses to want to get rid of them.[7] In the process, a description of the morphology gradually (and stealthily) unfolds.

"Wealth" refers to the end result of a powerful, linguistically satisfying, highly productive system for both processing texts and describing language. It allows our users at all levels of linguistic understanding to describe the grammar of a language to the level they are comfortable with. It provides a migration path from simple initial observations up to detailed, linguistically satisfying descriptions that take advantage of all the concepts supported in the conceptual model of morphology.

The model contains one concept that is motivated not by good practice in morphology but by the stealth-to-wealth strategy. This is the concept of an ad hoc co-occurrence prohibition. When users are so frustrated by incorrect parses that they want to get rid of them, but are not able to figure out the best way to model the needed constraints, they may resort to an ad hoc prohibition. This has the immediate effect of giving the desired improvement to parsing results; it has the long-term effect of recording the use of an ad hoc prohibition as part of the description in the database. The next time a linguistic consultant works with the user, s/he will retrieve all the instances of ad hoc prohibitions and work with the user on recasting the constraints in terms of the concepts in the morphology model, if possible.

Another example of stealth-to-wealth in action is the Morphological Glossing Assistant (Maxwell, Simons and Hayashi 2002). When a grammatical morpheme is glossed in interlinear text analysis, the MGA presents a view of the complete feature catalog as a choice list for possible glosses. As glosses are selected, they are added to a language-specific feature system which is being automatically constructed behind the scenes. When the user wants to use morphosyntactic features to describe constraints on parses, that draft feature system is already available as a starting point.

Figure 5 is a sample screen shot for the MGA. In this example, selecting *plural number* causes a feature specification for

[noun-agreement: [number: plural]]

to be created behind the scenes.

[7] We have prototyped help pages on "How do I get rid of incorrect parses?" which will describe how to use the features of the morphology model listed in the preceding section to constrain away incorrect parses.

Note that in our tool, users never write overt rules that are specific for the parser engine itself.[8] Rather, they merely enter as much information

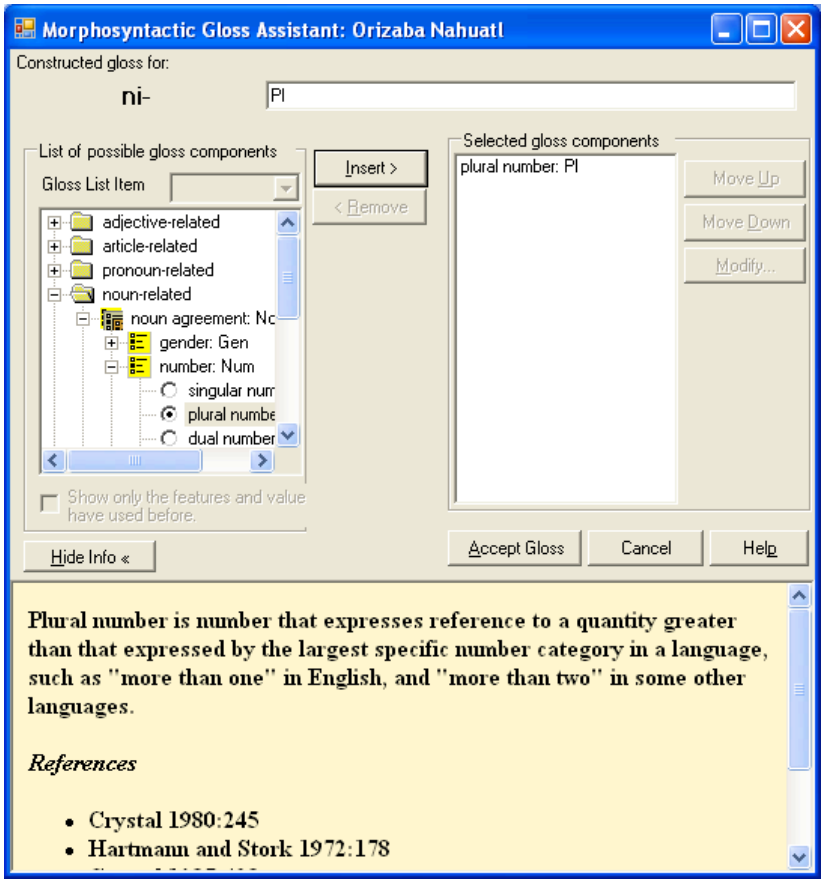

Figure 5. Morphological Glossing Assistant

about each lexical item, inflection, etc. as they know at the time into the conceptually-modeled database. The system translates that knowledge to rules. By having our parser allow for the full range of partial specification up to full specification, we have thus met the third requirement.

[8] In fact, we have designed the tool so that we could plug in more than one parsing engine. Therefore, we would not want the user to write any rules which were specific to only a given parsing engine.

5 A Descriptive Approach

The fourth requirement was that the user must be able to generate not only the information needed for the parser, but also a grammatical write-up of the description. We do this by auto-generating a grammatical sketch as well as the files needed by the parser. In both cases, the generation is from the modeled knowledge stored in the database.

The linguist's input is declarative knowledge and as such can be mapped to a human-readable description. We do this by extracting the relevant information from the SQL database in XML form and then transform this into HTML.[9] The resulting HTML page is displayed via an embedded Internet Explorer browser. Figure 6 shows the opening of a generated sketch for Orizaba Nahuatl.

[9] We actually transform the XML extracted from the database into an in-house XML for writing linguistic papers. We then transform this intermediate representation into HTML.

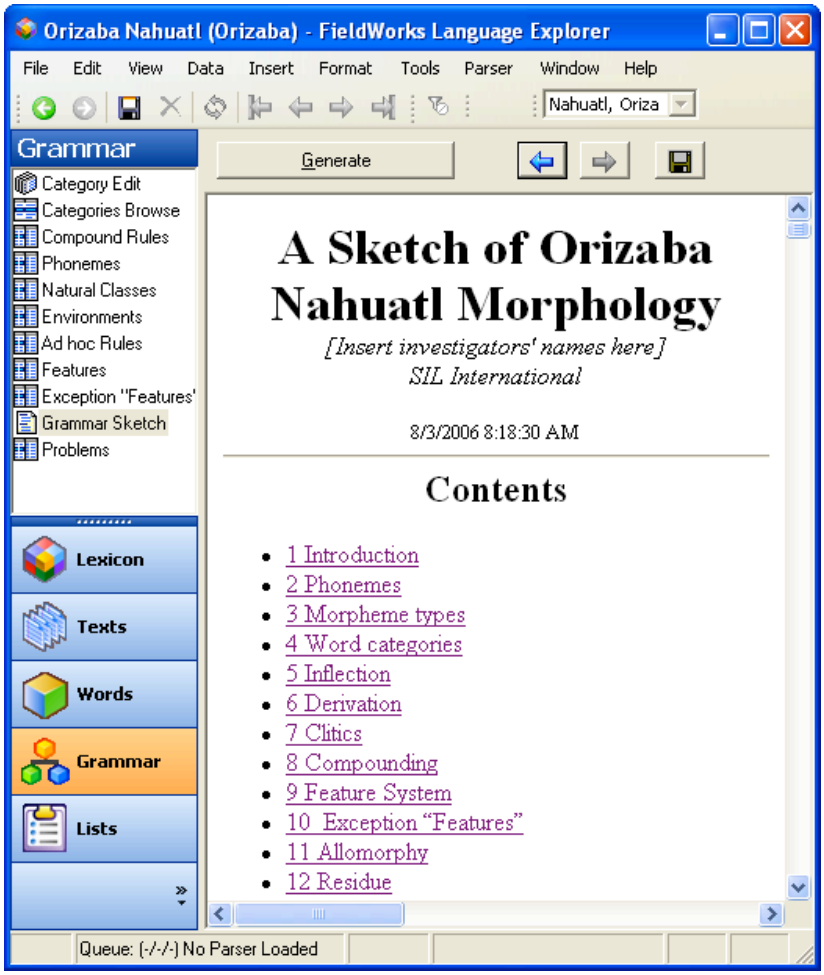

Figure 6. Contents of the generated morphology sketch

The sketch includes sections on the following items:

- Introduction covering brief information on the language and the nature of the sketch.
- A phoneme inventory in chart form. The user can provide a description for each phoneme.
- The set of morpheme types (e.g. prefix, infix, root, bound stem, suffix) used in the language description. It includes a count of the number of lexical entries for each type.

- A listing of the word categories used, including a count of the number of lexical entries for each category.
- A description of inflection in the language, including all the inflectional templates (by word category) and their slots. Figure 7 shows a section from the Orizaba Nahuatl sketch that describes one of the inflectional templates.
- A chart of the derivational affixes, showing the categories they attach to and the resulting category.
- A chart of any clitics.
- A listing of word-internal stem compounding rules.
- The feature system used, including the features and their values.
- Any exception "features" used to control non-productive affixation.
- A discussion of the allomorphy in the language, including any natural classes, phonological environments, and inflection classes used.
- A residue section which may include:
    - A listing of any roots or affixes which have been left underspecified for type or category (e.g. an affix that has not yet been classified as being inflectional or derivational or a root which has not yet been assigned to a category).
    - A listing of any ad hoc rules (designed to prevent the occurrence of unwanted combinations of morphemes or allomorphs).
    - A listing of any exception "features" which have been defined, but not used.

This sketch not only serves as a way to automatically document the user's description of the morphology of the language, it can also serve as a diagnostic device. For instance, when one user looked at the derivation section, he immediately noted that he had inadvertently left out some deriva-

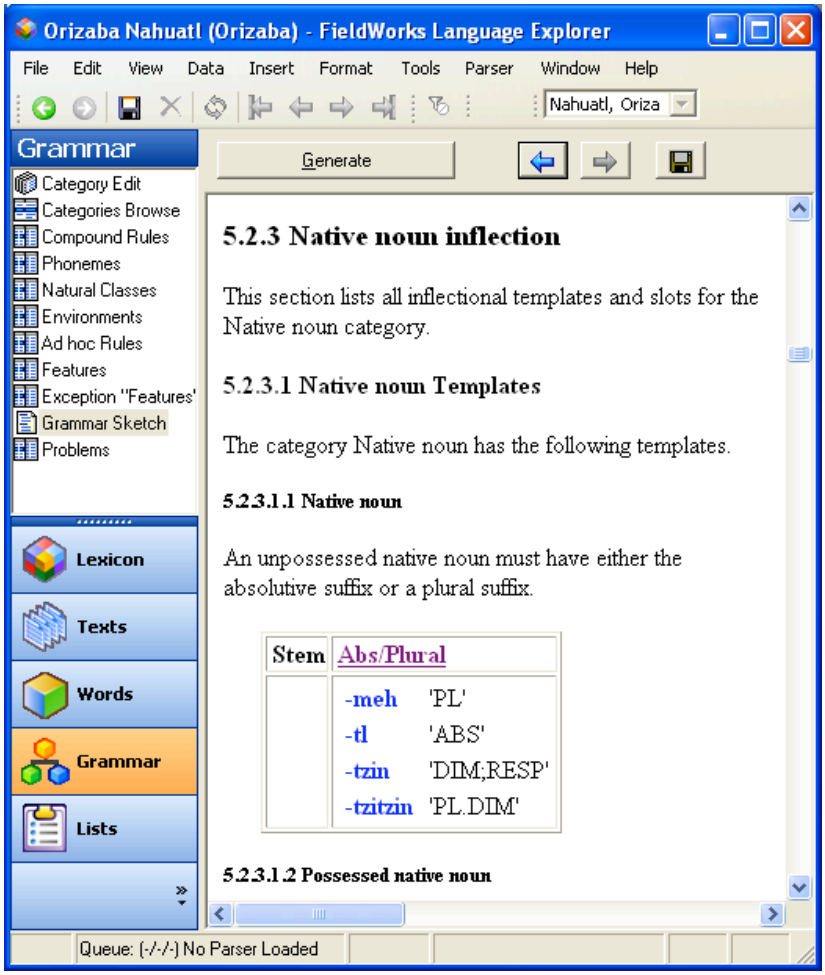

Figure 7. Portion of the generated morphology sketch

tional affixes: he knew that the language had derivational affixes mapping from categories that were not listed. The sketch also makes it possible for experienced consultants to review the work of inexperienced field workers and give timely help that will improve their understanding and description of the language. By producing this morphological sketch, we have met the fourth requirement.

6 An Eclectic Approach

After developing the conceptual model for the system, we realized that our current parsers could not implement it. But we also realized that we did not have the resources to implement a new parser from scratch. Thus we set the goal of reusing our existing parsing code as much as possible.

The best morphological parser we had was AMPLE. It, though, had a sequential right branching bias, which made it impossible in general to correctly infer the word category of a form involving both prefixes and suffixes or to correctly percolate features within the derivation. We also had PC-PATR, a unification-based syntactic parser. We realized that it could address the limitations of AMPLE if we merely treated the sequence of morphemes produced by AMPLE as the tokens to be input to a word grammar for PC-PATR.

Therefore, we chose to marry our AMPLE morphological parser with PC-PATR and produced a version we call XAMPLE.[10] When we generate the source files for XAMPLE, in addition to the lexicon and analysis control files traditionally required by AMPLE, we create a word grammar file for the embedded PC-PATR parser. After the AMPLE component has broken the wordform into a possible sequence of allomorphs, this preliminary result is passed to the word grammar which does the lion's share of the work, including:

- Handling inflectional templates
- Controlling inflection classes
- Constraining and percolating inflectional features
- Constraining exception "features"
- Constraining stem compounding
- Constraining derivation
- Constraining clitic attachment
- Allowing for unspecified or underspecified affixes or roots
- Percolating morphosyntactic features
- Allowing derivational affixes to override morphosyntactic features of the stem
- Constraining derivation outside of inflection appropriately
- Constraining derivational and inflectional circumfixes

Figure 8 is a portion of the automatically generated word grammar that is for the inflectional template illustrated in Figure 7. It consists of a phrase-structure rule followed by all the applicable feature unification constraints.

[10] The "X" in XAMPLE originally stood for "eXperimental," but now we like to think of it as meaning "eXtended." See http://www.sil.org/computing/catalog/show_software.asp?id=1 and http://www.sil.org/pcpatr/manual/pcpatr.html.

The numbers are SQL database IDs which are looked up in the database before results are displayed back to the user. For instance, 13150_0 is the identifier for the "Abs/Plural" slot and 13146 represents the "Native noun" category.

```
rule {Fully analyzed stem with a final inflectional template 13149}
  Full = Stem 13150_0
                                          | percolation
  <Full synCat> = <Stem synCat>
  <Full morphoSyntax> = <Stem morphoSyntax>
  <Full inflected> = +
  <Full requiresInflection> = -
                    | constraints
  <Stem blocksInflection> = - | prevent a non-final template from
                              | immediately being inflected without any
                              | intervening derivation or compounding
  <Stem synCat> = 13146
  <Stem morphoSyntax> = <13150_0 morphoSyntax>
  <Stem synCat> = <13150_0 envCat> | allomorph
  <Stem morphoSyntax> = <13150_0 envMorphoSyntax>
  <Stem exception> = <13150_0 fromException>
  <Stem inflectionClass> = <13150_0 inflectionClass>
```

Figure 8. Portion of generated word grammar for XAMPLE

We have thus accomplished the goal of having the user control the parser primarily by describing the inflection, derivation, and compounding components of the morphology, along with delineating allomorphy. The user does not have to write any rules specific to XAMPLE. Rather, the system "compiles" the descriptive knowledge in the conceptually modeled database into the input files needed by XAMPLE.

A note on performance is in order. Benchmarks run on a 3.2GHz Pentium 4 with 1GB of memory show that AMPLE parses 140 words per second on a full description of Southeastern Puebla Nahuatl, a morphologically complex language of south central Mexico [ISO 639-3 code: nhs]. By contrast, an XAMPLE description of closely related Orizaba Nahuatl [ISO 639-3 code: nhv] parses only 4.9 words per second. Clearly there is considerable room for improving performance in the PC-PATR component of the parser. Nevertheless, given the fact that parsing takes place in the FieldWorks Language Explorer on a word-by-word basis as the linguist works through interlinearizing a text, the performance is adequate for the present purposes.

7 Conclusion

Field workers seeking to describe less-studied languages have different skills and needs than computational linguists. As we have sought to show, the parser in the SIL FieldWorks Language Explorer meets their requirements as outlined above in section 1:

- Since the parser is incorporated into this tool, it is fully integrated into lexicon management and interlinear text analysis so that users do not need to learn one more program.
- The underlying model of morphology is familiar to linguists.
- It is easy to get started, yet one can finish well.
- It not only creates the files that the parser needs, it also creates a descriptive grammar sketch as well.

In addition, we succeeded in reusing our existing parsing software to deliver this new functionality.

The SIL FieldWorks Language Explorer runs in a Windows environment and is currently in a final beta version pending full release later this year. It may be freely downloaded by any interested party from this URL: http://www.sil.org/computing/fieldworks/FW_downloads.htm.

Acknowledgements

We are indebted to a host of colleagues who have been part of this project which began to take shape eight years ago. During the formative years of the project, Mike Maxwell was a critical member with us on the conceptual modeling design team. Larry Hayashi has served as the keeper of the formal models. John Hatton and Randy Regnier (with the first author) comprised the team that brought the designs to proof-of-concept implementation. In the last couple years, as these results have been incorporated into the SIL FieldWorks suite, many more colleagues have played a significant role in bringing the Language Explorer into production: Mike Cochran, Curtis Hawthorne, Dan Hinton, Marlon Hovland, Alistair Imrie, Susanna Imrie, Michael Lastufka, Rick MacLean, Stephen McConnel, Steve Miller, Ron Moe, Eric Pyle, Rob Scebold, Howard Sheldon, John Thomson, Dennis Walters, and Ken Zook, among others.

References

Antworth, E. L. 1990. *PC-KIMMO: A Two-Level Processor for Morphological Analysis*. Occasional Publications in Academic Computing No. 16. Dallas: Summer Institute of Linguistics. (http://www.sil.org/computing/catalog/show_software.asp?id=33).

Bickford, J. A. 1998. *Tools for Analyzing the World's Languages*. Dallas: Summer Institute of Linguistics.

Davis, D. W. and J. S. Wimbish. 1993. *The Linguist's Shoebox: An Integrated Data Management and Analysis Tool (version 2.0)*. Waxhaw, NC: Summer Institute of Linguistics. (http://www.sil.org/computing/shoebox/index.html).

Fowler, M. 2003. *UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition*. Reading, MA: Addison Wesley.

Grimes, J. E. 1975. *Network Grammars*. Summer Institute of Linguistics Publications in Linguistics and Related Fields, 45. Norman: Summer Institute of Linguistics of the University of Oklahoma.

Hayashi, L., and J. Hatton. 2001. Combining UML, XML and relational database technologies – the best of all worlds for robust linguistic databases. *Proceedings of the IRCS Workshop on Linguistic Databases, December 11-13*, University of Pennsylvania, Philadelphia, USA, 115-124. (http://www.ldc.upenn.edu/annotation/database/papers/Hayashi_Hatton/23.3.hayashi.pdf).

Hockett, C. 1954. Two models of grammatical description. *Word* 10:210-231.

Karttunen, L. 1983. KIMMO: a general morphological processor. *Texas Linguistic Forum* 22:163-186.

Maxwell, M. 2001. M3 Morphology: Description of Classes. SIL International ms.

Maxwell, M., G. Simons, and L. Hayashi. 2002. A morphological glossing assistant. *Proceedings of the Workshop on Resources in Tools and Field Linguistics, Third International Conference on Language Resources and Evaluation (LREC), May 26-27*, Las Palmas, Canary Islands, Spain, 25-1 to 25-10. European Language Resources Association. (http://www.sil.org/~simonsg/preprint/Morphological%20Glossing%20Assistant.pdf).

McConnel, S. 1995. PC-PATR Reference Manual. SIL International. (http://www.sil.org/pcpatr/manual/pcpatr.html).

Shieber, S. M. 1986. *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes, 4. Stanford, CA: Center for the Study of Language and Information.

Simons, G. F. 2000. CELLAR: A data modeling system for linguistic annotation. *Proceedings of the Workshop on Data Architectures and Software Support for Large Corpora* Second International Conference on Language Resources and Evaluation, Athens, Greece, 10-15. European Language Resources Association. (http://www.sil.org/~simonsg/local/LREC%202000%20Simons.pdf).

Simons, G. F. and B. Hughes. 2006. GOLD as a standard for linguistic data interoperation: A road map for development. *Proceedings of the EMELD'06 Workshop on Digital Language Documentation: Tools and Standards: The State of the Art*, Lansing, MI. June 20-22, 2006. Electronic Metastructures for Endangered Language Data. (http://emeld.org/workshop/2006/papers/SimonsHughes.doc).

Weber, D J., H. A. Black, and S. R. McConnel. 1988. *AMPLE: A Tool for Exploring Morphology*. Occasional Publications in Academic Computing No. 12. Dallas,

Texas: Summer Institute of Linguistics. (http://www.sil.org/computing/catalog/show_software.asp?id=1).

Weber, D. J. and W. C. Mann. 1979. *Prospects for computer-assisted dialect adaptation*. Notes on Linguistics Special Publication, 1. Dallas: Summer Institute of Linguistics.

Weber, D. J. and W. C. Mann. 1981. Prospects for computer-assisted dialect adaptation. *American Journal of Computational Linguistics* 7:165-177.