Implementing a Machine-Readable Grammar of Uyghur (UIG)

Alexander Sugar
University of Washington

Abstract

Creating a machine-readable grammar for the Uyghur language presents various challenges. The portion of Uyghur grammar under discussion was based on code provided by the LinGO Grammar Matrix, which uses a syntactic formalism based on Head-Driven Phrase Structure Grammar (HPSG). HPSG sentences are considered "well formed" when they satisfy a series of phrasal and lexical constraints. Most information about a word—its semantic properties, potential for morphological changes, and the types of words or phrases with which it combines—is stored in lexical entries. A test suite based on authentic Uyghur texts and native speaker judgments led to the formulation of lexical types and lexical and phrasal rules, all organized into type hierarchies. Specific grammatical properties of Uyghur, including case marking and verb morphology, were modeled using HPSG.

Keywords: Uyghur, HPSG, grammar engineering

O. Introduction

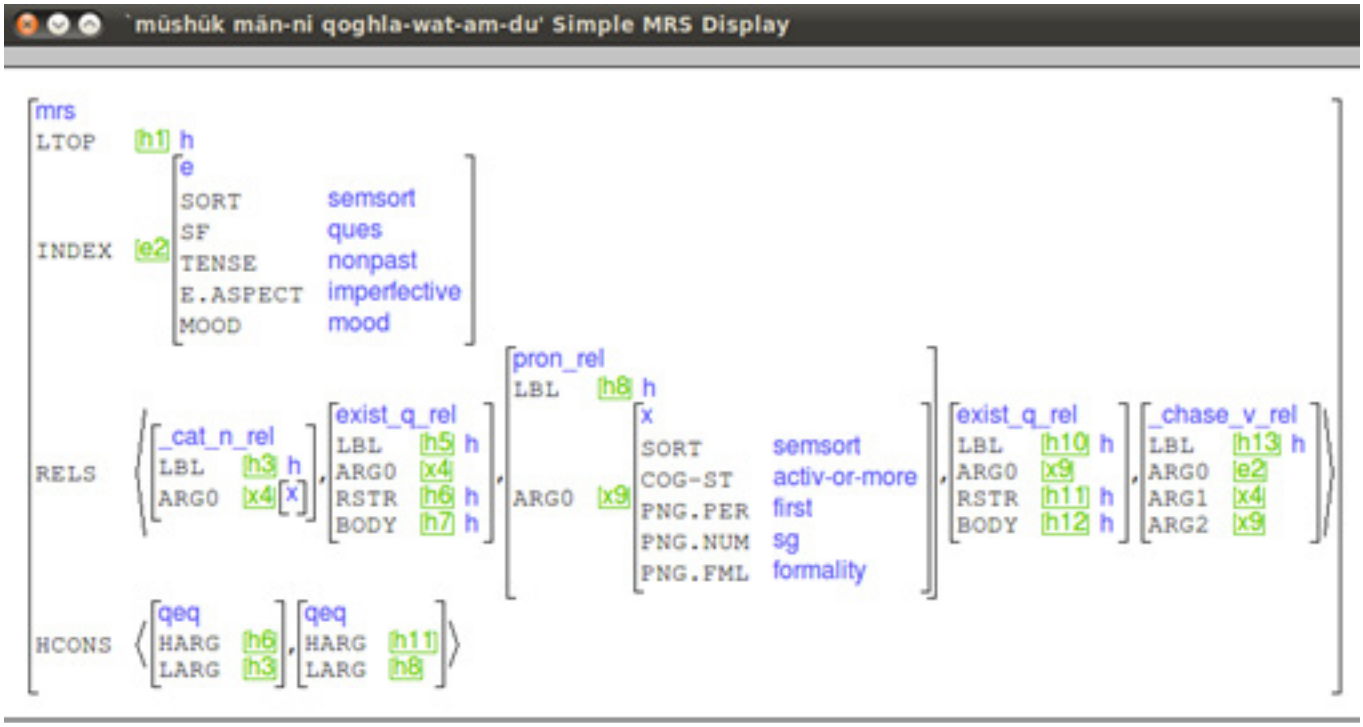
While the syntax of Uyghur has received some scholarly attention over the last decade (Asarina 2006, Bridges 2008, Engesæth et al 2010 among others), it has yet to have been studied, to this author's knowledge, through the formalism of Head-Driven Phrase Structure Grammar (HPSG). The Uyghur language has also been underrepresented in linguistic technology innovations like search engines and translation software. This paper describes the beginning of a project aiming to address both of these deficiencies by building a machine-readable or precision grammar of Uyghur. Section I describes the formalism and methodology of the project, including A) grammar engineering, B) the Grammar Matrix and C) HPSG. Section II includes examples of notable phenomena of Uyghur syntax and morphology and how they were dealt with in HPSG. Section III lists current coverage of the grammar and concludes with a look to future work.

I. Formalisn and Methodology

A. Grammar Engineering

Grammar engineering is the process of making formal rules of linguistic knowledge precise enough that they can be encoded on a computer (Bender et al 2011). In other words, the grammar engineer writes into a series of files all of the phrasal rules, lexical rules, lexical entries etc. that a computer will need to accurately process a given language. Processing here means both parsing and generating. A computer should be able to successfully parse grammatical sentences, connecting strings of words to only their correct semantic representations; it should fail to parse ungrammatical strings because they violate specific grammatical constraints. In the case of Uyghur, a computer should be able to take the string "müshük meni qoghliwatamdu" (is a cat chasing me?) and produce a semantic representation with a chasing event in progressive/imperfective aspect. The chaser is a cat, the thing being chased is first person singular, and the speaker is asking a yes-or-no question about the

event's occurrence. This semantic representation is shown below using a Minimal Recursion Semantics (MRS) graphic (Copestake et al 2005).



Conversely, a computer should be able to produce one or more sentences that convey the desired meaning in a language given a target semantic representation. A computer is able to perform these tasks by having a model of the linguistic knowledge possessed by a speaker of a given language.

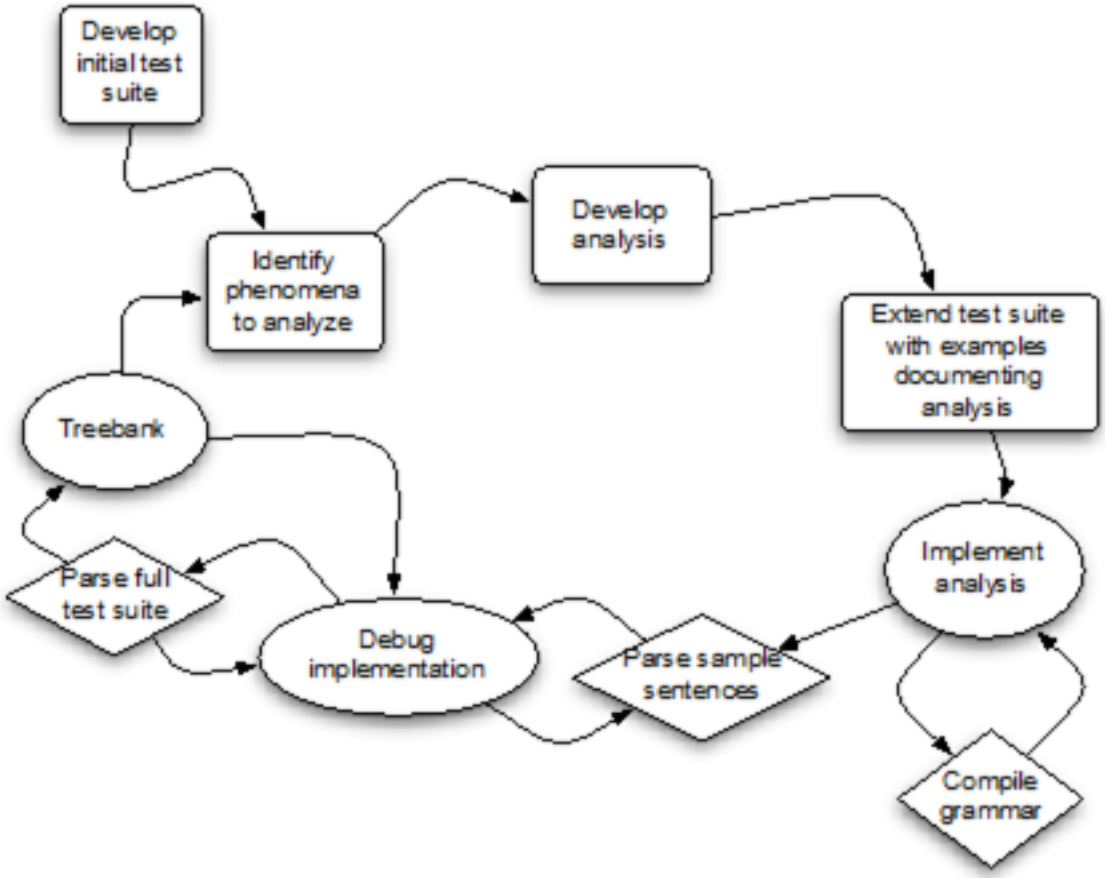
While there is an interest in comparing knowledge bases of distinct languages to see where they overlap and differ, the process of grammar engineering is bottom up. It begins by focusing only on the facts available about a specific language, with the collection of a test suite. The test suite consists of grammatical and ungrammatical sentences from primary documentation of the language, ideally vetted by native speakers. The test suite should cover a broad range of linguistic phenomena from basic word order, to tense and person marking, to embedded clauses. Sentences are listed in a file with information about their source, grammaticality, phenomena exhibited, English glosses and rough translations, as in the sample entry below.

```
#1 SOV word order
Source: elicited
Vetted: t
Judgment: g
Phenomena: wo
Mān kitab oquymān
Mān kitab oqu-y-mān
1SG.NOM book read-NPST-1SG
'I will read a book.'
```

The grammar engineer will then formulate phrasal and lexical rules that account for the grammatical sentences while ruling out the ungrammatical sentences. Progress in accurately covering test suite sentences can be monitored using software like [incr tsdb()] (Oepen 2001). Finally, the grammar engineer will evaluate the coverage of the grammar over a test corpus, a collection of authentic text in a given language held out from the initial test suite.

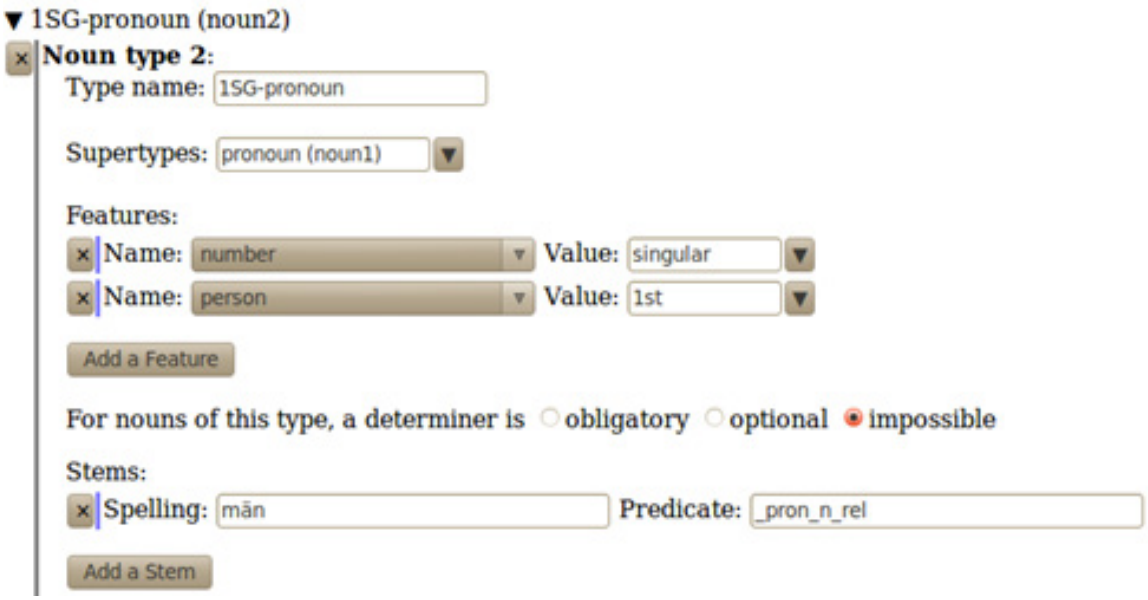
Testing a grammar over sentences in test suites and corpora, the linguist will inevitably need to add or remove constraints from existing rules, and even write new ones. Because grammars are written to cover all the linguistic knowledge of languages, rather than targeting certain aspects of language, any change made to a part of a grammar can affect the way it treats other aspects of a language. Thus, an important part of the grammar engineering process is reanalyzing coverage after each change is made to the grammar, to see whether a change made with one problem in mind has caused a new problem or even solved what seemed to be an unrelated problem. A single rule governing the relationship between heads and complements, for example, can apply to both verb phrases and preposition phrases.

At various stages of achieving coverage, the grammar engineer may wish to mark the preferred analyses of linguistic phenomena by treebanking before finding new phenomena to analyze. This cyclic workflow is summarized below (from Bender et al 2011).



B. The Grammar Matrix

Bender et al (2010) devised a starter kit known as the Grammar Matrix to assist in the building of precision grammars. For the linguist user, the Matrix takes the form of an online questionnaire (located at http://www.delph-in.net/matrix/customize/matrix.cgi) to be filled out with information about a specific language. The Matrix attempts to capitalize on cross-linguistic commonality by allowing many encoded linguistic rules to be shared across grammars (Bender 2008). Where languages may vary, the Matrix is divided into "libraries" each devoted to a different linguistic phenomenon. So far phenomena addressed by Matrix libraries include negation, adjectives and coordination. The linguist user is given a series of options from which to select and in some cases may be asked to provide specific instances of lexical types within a library to best represent their language. The image below shows a first person singular pronoun being defined in the matrix.



After completing the questionnaire, the user can download files documenting their choices and the rules produced by the Matrix in Type Description Language (TDL) (Copestake 2002). While hand editing of the TDL may be needed as more facts emerge about the language one wishes to model, the Grammar Matrix provides a significant jump-start to the grammar engineering process.

C. HPSG

The TDL files produced by the Grammar Matrix and used in the grammar engineering process follow the formalism of HPSG (Pollard and Sag, 1994). HPSG is fundamentally a constraint-based grammar, in which each word is stored as a lexical entry inheriting a specific set of constraints according to its type. Lexical types are organized into type hierarchies so that constraints shared by multiple types need not be restated on each entry. In order to be realized as a word in a sentence, a lexical entry must go through one or more lexical rules, which add information about the word and its behavior in a sentence, and may also add an affix.

A noun in the Uyghur grammar like *alma* (apple) is a stem of a type called normal-noun-lex. This lexeme type inherits constraints from a broader type noun-lex (including the constraints that it needs case inflection and can take a determiner), and adds the constraints that it is always third person and must be inflected for number via a lexical rule.

```
alma := normal-noun-lex &
  [ STEM < "alma" >,
    SYNSEM.LKEYS.KEYREL.PRED "_apple_n_rel" ].
```

```
normal-noun-lex := noun-lex &
  [ SYNSEM.LOCAL.CONT.HOOK.INDEX.PNG.PER 3rd,
    INFLECTED.NUM-FLAG - ].
```

Alma must go through a number lexical rule before entering a sentence as a word, and it has two options. The singularnoun-lex-rule and the pluralnoun-lex-rule will mark the noun's NUM value as singular or plural, respectively. Both rules inherit the instruction to set the NUM-FLAG as +. While the singularnoun-lex-rule inherits from a rule licensing constants (adding no morphemes to the stem), the pluralnoun-lex-rule inherits from a rule licensing inflections, and is associated with a rule adding the plural suffix *-lar*. Both rules are shown below.

```
singularnoun-lex-rule := number-lex-rule-super & const-lex-rule &
  [ SYNSEM.LOCAL.CONT.HOOK.INDEX.PNG.NUM singular ].

pluralnoun-lex-rule := number-lex-rule-super & infl-lex-rule &
  [ SYNSEM.LOCAL.CONT.HOOK.INDEX.PNG.NUM plural ].
```

Words are put together through phrase structure rules, which specify which types may combine, and what each constituent word contributes to the phrase it helps form. Like lexical types, lexical and phrase structure rules are organized into type hierarchies such that the most widely shared constraints are located in higher nodes, while lower nodes contain constraints specific to a smaller set of instances. Let us suppose the noun *alma* has gone through the pluralnoun-lex-rule to become *almilar* (alma + lar, 'apples') and is ready to go through a phrase structure rule. *Almilar* can be part of a verb phrase if we combine it with the verb *yäydu* (*yä* + *y* + *du*, '(s) he/they eat(s)'), which will have been licensed through a different set of lexical rules applying to verbs. These two words will be put together by the comp-head-phrase rule, which specifies that the second of its two daughters becomes the head of the phrase: the daughter whose semantic index is shared with its mother's. The result is *almilar yäydu* ('(s)he/they eat(s) apples').

An HPSG grammar also records a specific list of constraints that a sentence must satisfy in order to be considered well formed. The ultimate goal of HPSG is to map syntactic strings to semantic representations, using MRS.

II. Sample Phenomena

A. Accusative Case Definiteness

In Uyghur, direct objects may appear either with or without a visible accusative case suffix (*-ni*). The presence or absence of this suffix affects the definiteness of the object. Consider the following sentences:

(a) Kitabni oquymän
Kitab-ni oqu-y-män
book-ACC read-PF-1SG
"I will read the book."

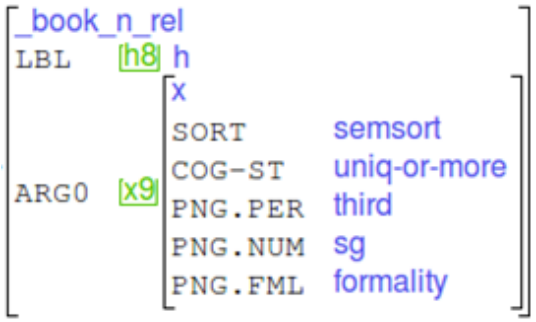
(b) Kitab oquymän
Kitab oqu-y-män
book.ACC read-PF-1SG
"I will read a book."

With (*-ni*) present in (a), the speaker asserts that they will read a specific book. Without this suffix in (b), the speaker merely asserts that they will read a book, without a specific book in mind.

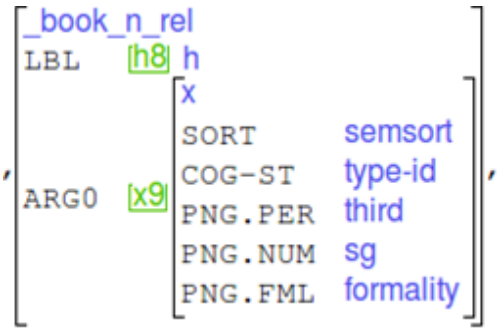
Every noun in the Uyghur HPSG grammar must go through a case-assigning lexical rule, which will mark it as eligible for playing certain roles in a sentence. The first question here is what case a bare indefinite object like 'kitab' should be assigned. I opt to consider 'kitab' as having accusative case without a suffix, which clarifies its role as the direct object in sentences like (b) and avoids the unwanted ambiguity of possibly being the sentential subject (if it were marked with nominative case).

To encode this analysis, two separate accusative case rules must be created: one which adds the suffix (*-ni*) and specifies that the noun refers to a definite object, and the other which adds no suffix and specifies that the noun refers to a general type of object. These two rules can inherit all other information associated with accusative case from the same parent lexical rule. The definiteness contrast is recorded in the MRS through a feature of nouns called COG-ST (cognitive status). The accusative case rule which ads the (-ni) suffix gives 'kitab' a COG-ST value of uniq-or-more (that there is a unique item being referenced), as shown in (a); while the rule which does not add a suffix gives 'kitab' a COG-ST value of type-id (that its type is identifiable), as shown in (b).

a.



b.



B. Verb Morphology

Verbal suffixes in Uyghur occur in a relatively fixed order (e.g. negation, tense, aspect, person), but only person needs to be overtly present on every verb. To track a verb's morphological realization, a flag system is used (Goodman 2013). Suffix morphemes are organized into position classes based on their appearance in relation to the stem and other suffixes; each position class is represented by a flag. Flags have *luk* values (ternary and disjunctive) indicating that a given position class needs to be realized (-), has been realized (+), is not relevant in the given context (*na*), or is either *na* or one of + or -. In order for a word to be considered properly "inflected" for participation in a phrasal rule, all of its flags must be set to *na-or-*+; i.e., a word with a position class that needs filling is not considered properly inflected.

For a concrete example, consider the following paradigm of tense morphology in Uyghur. A standard nonpast verb with no special aspect marking will be marked by the -y suffix, as shown in (a).

(a) Itlar uxlaydu
It-lar uxla-y-du
Dog-PL.NOM sleep-NPST-3
"Dogs will sleep."

When the progressive aspect suffix -*wat* is added, however, then the -*y* suffix is not added, although the sentence is still interpreted as nonpast (Engesæth et al 2010).

(b) Itlar uxliwatidu
It-lar uxla-0-wat-du
Dog-PL.NOM sleep-NPST-PROG-3
"Dogs are sleeping."

Yet when the negation suffix -*ma* is added along with the progressive suffix -*wat*, then the nonpast -*y* suffix becomes overt once again.

(c) Itlar uxlimaywatidu
It-lar uxla-ma-y-wat-du
Dog-PL.NOM sleep-NEG-NPST-PROG-3
"Dogs are not sleeping."

This peculiar contrast can be modeled in HPSG using flags. Relevant to the above examples, the flags TENSE-FLAG, COVERTNONPAST-FLAG, PROG-FLAG and NEG-FLAG are all defined under INFLECTED. When verb-lex is instantiated, TENSE-FLAG is set to - (since every verb must be inflected for tense), while the other three flags are set to *na* (since it is not necessary that a verb in a vacuum have unmarked nonpast tense, progressive aspect or negation.) For sentence (a), the verb *uxla* will simply go through the nonpast-lex-rule, setting its TENSE-FLAG to +, and the word will then be ready to go through another lexical rule assigning it a person suffix.

A verb that has received the -y suffix will not be eligible to receive the progressive -wat suffix, because

the progressive-lex-rule requires its daughter to have a COVERTNONPAST-FLAG + value. Thus the verb in (b) first goes through the covertnonpast-lex-rule, which does not add a suffix, sets the COVERTNONPAST-FLAG and the TENSE-FLAG values to +, and also sets the PROG-FLAG to -, requiring that the verb now go through the progressive-lex-rule.

In (c), the verb *uxla* first goes through the neg-lex-rule, which adds the suffix *-ma* and sets the NEG-FLAG to +. The verb is now eligible for the negprogovert-nonpast-lex-rule, which adds the *-y* suffix but sets the COVERTNONPAST-FLAG value to +, "tricking" the progressive-lex-rule into adding the -wat suffix.

The solution provided here to reproducing this inflectional paradigm is only one of many possible ways to model the morphological reality observed in the Uyghur language.

C. Nominal Predicate

Unlike English, Uyghur allows nouns to serve as predicates without the linking of a copula verb. Thus Uyghur can have complete sentences like,

Bu zhurnal
DEM magazine
"This is a magazine."

To model this reality, a noun-predicate-rule was created that takes a noun in nominative case as a daughter and converts it into a verb. The rule also adds a "be" predication, linking the first argument with the sentential subject and the second argument with the index identifying the rule's head daughter noun. This rule allows sentences that seemingly "don't have a verb" to be licensed while still observing the requirement that a well-formed Uyghur sentence have a verb.

D. Tranformation Rules

Since machine translation is a valuable natural language processing (NLP) application, it is important to engineer a grammar that will be mutually intelligible with other grammars. Transfer rules ensure that components of one grammar can be interpreted in another. Their role is to match potential input from other grammars with a grammar's own types. Most languages have a unique way of encoding tense and aspect, for example. A language like Chinese has no tense morphology, but aspectual morphemes allow the interlocutor to infer tense. The perfective marker *guo*, for example, indicates that an event occurred in the past. A transfer rule that maps perfective markers to past tense would then be a useful transfer rule to include in an English grammar. Transformation rules are also used when predicates don't exactly match between languages. The meaning of the verb *hurt* in English is morphologically expressed as *cause pain* in Uyghur (*aghir-t*, lit. pain-CAUS). Transfer rules can encode these relationships. It would be preferable to develop a systematic way of identify these predicate correspondences, but doing so is beyond the scope of this project.

III. Conclusion

The process of building a precision grammar of Uyghur has been described with the accusative object definiteness contrast, one verb morphology paradigm and nominal predicates given as examples of specific properties of Uyghur grammar that can be treated in a machine-readable HPSG approach.

As measured using [incr tsdb()], this Uyghur grammar currently provides 75.3% coverage over a test suite containing 253 items. Many of the sentences lacking a parse exhibit grammatical phenomena (such as dative case) yet to be defined in the grammar. Ambiguity (in the form of multiple parses per string) often arises

because a Uyghur speaker can switch the order of subject and object in some sentences to focus on the latter. As the grammar expands to cover more phenomena, so too will the test suite grow to exhibit a greater variety of phenomena, prompting additional modifications to the grammar in a cycle of discovery.

Grammar engineering provides the ability to parse and generate sentences in real time, allowing for quick testing of linguistic hypotheses. Precision grammars can also easily be compared to determine what properties are or are not common to language families or even universally shared by languages. Additionally, machine-readable grammars can be used in a variety of NLP tasks from machine translation to information retrieval. This project represents early steps in the vast process that is building up a precision grammar of a language. This Uyghur grammar is currently being used to model a class of Uyghur auxiliaries that contribute aspectual information rather than an independent predicate.

References

Asarina, Alevtina. (2006). *Case in Uyghur and Beyond*. PhD Diss., Massachusetts Institute of Technology.

Bender, Emily M. (2008). "Evaluating a Crosslinguistic Grammar Resource: A Case Study of Wambaya." In *Proceedings of ALC08:HLT*. Columbus, OH: Association for Computational Linguistics.

Bender, Emily M., Scott Drellishak, Antske Fokkens, Laurie Poulson, and Safiyyah Saleem. (2010). "Grammar Customization." *Research on Language and Computation* 8(1):23-72.

Bender, Emily M., Dan Flickinger and Stephen Oepen. (2011). "Grammar Engineering and Linguistic Hypothesis Testing: Computational Support for Complexity in Syntactic Analysis." In *Language from a Cognitive Perspective*, ed. Emily M. Bender and Jennifer E. Arnold, pp. 5-30. Stanford: CSLI Publications.

Bridges, Michelle. (2008). *Auxiliary Verbs in Uyghur*. MA Thesis, Kansas University.

Copestake, Anne. (2002). "Definitions of Typed Feature Structures." In Oepen, Stephan, Dan Flickinger, Junichi Tsujii and Hans Uszkoreit, eds., *Collaborative Language Engineering*. Stanford: CSLI Publications. pp. 227-230.

Copestake, Ann, Dan Flickinger, Carl Pollard and Ivan Sag. (2005). "Minimal Recursion Semantics: An Introduction." *Research on Language and Computation* 3(4), 281-332.

Engesæth, Tarjei, Mahire Yakup and Arienne Dwyer. (2010). Greetings from the Teklimakan: a Handbook of Modern Uyghur. Lawrence, KS: University of Kansas Scholarworks.

Goodman, Michael Wayne. (2013). "Generation of Machine-Readable Morphological Rules from Human-Readable Input." In *University of Washington Working Papers in Linguistics* (13).

Oepen, Stephan. (2001.) [*incr tsdb()*] — *Competence and performance laboratory. User manual*. Technical report. Saarbrücken, Germany.

Pollard, Carl and Ivan A. Sag. (1994). *Head-Driven Phrase Structure Grammar.* Studies in Contemporary Linguistics. Chicago, IL and Stanford, CA: The University of Chicago Press and CSLI Publications.