Chinese in the Grammatical Framework: Grammar, Translation, and Other Applications

Aarne Ranta
University of Gothenburg
aarne@chalmers.se

Yan Tian
Shanghai Jiao Tong University
tianyan@sjtu.edu.cn

Qiao Haiyan
Sun Yat-sen University
qiaohy@mail.sysu.edu.cn

Abstract

Grammatical Framework (GF) is a grammar formalism based on type theory and functional programming. It is also a platform for multilingual applications such as translation, localization, and information retrieval. To enable non-linguist programmers to build linguistically precise applications, GF provides a Resource Grammar Library (RGL), which defines the basic syntax, morphology, and lexicon of languages in the form of easily usable software libraries. The RGL is an open-source collaborative project, which currently covers 30 languages with a shared tree structure. Chinese, in addition to basic RGL, has a translation lexicon of over 30,000 lemmas and a mobile translation app. This paper gives an overview of GF, emphasizing applications where Chinese is related to other languages. We also address the theoretical question how Chinese fits into the framework with a shared tree structure.

1 Introduction

Computer implementations of grammars used to be an important part of computational linguistics (e.g. TAG (Joshi, 1985), LFG (Bresnan, 1982), CCG (Steedman, 2000), and HPSG (Pollard and Sag, 1994)). But in the last couple of decades, they have been largely overshadowed by statistical methods and machine learning. However, handwritten grammars can still give valuable contributions to natural language processing. For instance, in machine translation (MT), grammars written with guidance from linguistic knowledge have the following advantages:

- Grammars **don't need so much data**, which is useful for language pairs with a lack of parallel texts.
- Systems using grammars are **predictable and programmable**, which is useful in mission-critical applications.
- Grammars are **compact representations** compared with e.g. phrase tables, which is useful in mobile applications.

In Information Retrieval (IR),

- Grammars enable a **precise logical analysis of content**, supporting detailed queries and powerful reasoning.

In Computer-Aided Language Learning (CALL),

- Grammars support **detailed error analysis and explanations**.

The main problems associated with grammars are their **limited coverage** and the **high cost** of building them. However, techniques of shallow parsing such as parsing by chunks (Abney, 1991) make it possible to overcome the limited coverage and, among other things, create robust MT systems based on grammars rather than statistics (Forcada et al., 2011).

The cost of grammar engineering can be reduced by modern software engineering techniques, which have made programming in the 2010's more productive than it used to be in the "golden age" of computational grammars, 1970's and 1980's. Such techniques form the basis of GF (Grammatical Framework, (Ranta, 2004; Ranta, 2011)), which is a programming language designed for multilingual grammar engineering:

- **Functional programming**, enabling powerful abstractions and generalizations;
- **Static type systems**, guaranteeing the consistency of the highly complex programs that grammars are;
- **Advanced module systems**, supporting collaborative work and maximal code reuse;
- **Libraries**, supporting division of labour and encapsulation of expert knowledge.

GF enables building a comprehensive grammar in a few months, e.g. as a Masters thesis project

Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing (SIGHAN-8), pages 100–109,
Beijing, China, July 30-31, 2015.

(Zimina, 2012). Adapting a GF grammar to a new setting, such as a dialogue system or a domain-specific translator, can be accomplished in a few days (Perera and Ranta, 2007; Ranta et al., 2012).

GF started at Xerox Research Centre Europe in 1998 as a part of a project on **multilingual document authoring** (Dymetman et al., 2000). Released open-source later, GF today is a collaborative project with over 150 developers around the world. In China, GF courses have been organized at Shanghai Jiao Tong University and at Sun Yat-Sen University in Guangzhou. The standard textbook on GF, (Ranta, 2011) has recently been translated to Chinese (Ranta, 2014b).

The GF software and grammar resources, including Chinese, are available from the GF homepage[1]. The licenses of the grammar reaources (LGPL and BSD) permit all kinds of uses, including commercial applications.

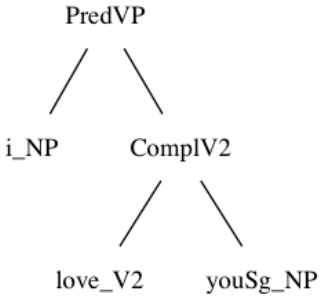
This paper gives an overview of the GF resources and applications available for Chinese. Section 2 introduces the idea of multilingual grammars. Section 3 describes the GF tool set and applications enabled by them. Section 4 summarizes the main issues encountered when adding Chinese to GF. Section 5 describes some controlled language applications. Section 6 shows how GF can scale up to wide-coverage translation. Section 7 discusses evaluation and Section 8 related work. Section 9 concludes.

2 Multilingual grammars

A grammar defines a language: a set of strings and the analyses assigned to them, typically trees. In the usual view, every language has its own grammar, and trees in different grammars are distinct objects. Grammar-based translation systems such as (Rayner et al., 2000) typically map the trees of one language into trees of another language.

Monolingual grammars can also be written in GF. But its real power comes with **multilingual grammars**, where several languages use the same trees, called **abstract syntax trees** (ASTs). An AST expresses **pure constituency**, for instance, that a sentence has a certain subject, verb, and object. But it does not specify the actual words, their inflection forms, or the order in which they appear.

To give a simple example, consider the sentence *I love you*. A possible AST is



more conveniently represented as a LISP-like term

```
(PredVP i_NP
  (ComplV2 love_V2 youSg_NP))
```

An AST has a function F and 0 or more argument ASTs. In the example, the function is `PredVP`, marking predication. Its arguments are `i_NP`, marking the noun phrase *I* and `(ComplV2 love_V2 youSg_NP)`, which is a verb phrase built from the two-place verb *love* and the pronoun *you* in the singular sense.

The 0-place functions `i_NP`, `love_V2`, and `youSg_NP` have names formed from English words, but they stand for interlingual word senses, so that for instance the plural and singular *you* have distinct functions. A more accurate analysis might also distinguish genders and politeness levels of pronouns.

The AST above corresponds to different strings in different languages. For example:

- English: *I love you*
- Chinese: *wo ai ni* ("I love you", just changing the words)
- Dutch: *ik houd van je* ("I hold of you", adding a preposition)
- French: *je t'aime* ("I you-love", the object pronoun before the verb)
- Italian: *ti amo* ("you love(1st person singular)", dropping the subject pronoun)

(We use Pinyin for most Chinese examples in this paper, but the actual GF implementation uses simplified Chinese characters in UTF-8 encoding.)

Even more variation is shown when question formation is applied to the clause:

```
(QuestCl (PredVP i_NP
  (ComplV2 love_V2 youSg_NP)))
```

Languages use widely different mechanisms to express this:

- English: *do I love you* (auxiliary verb)
- Chinese: *wo ai ni ma* (particle) or *wo ai bu ai ni* (reduplication)

[1] www.grammaticalframework.org

- Dutch: *houd ik van je* (inversion)
- French: *t'aime-je* (inversion)
- Italian: *ti amo* (intonation in spoken question)

Nonetheless, the AST can be shared.

The ASTs have types and are thus terms in type theory. Each function has a type that indicates the types of its arguments and its value. Basic types (**categories**) are introduced by rules such as

```
cat NP
```

Functions are introduced by rules such as

```
fun PredVP : NP -> VP -> Cl
```

stating that `PredNP` takes two arguments, of types `NP` and `VP`, and returns a `Cl` (clause).

Each function has, for each language in the grammar, a **linearization rule**, which specifies how trees are converted to strings. Thus the rule

```
lin PredVP np vp = np ++ vp
```

says that the first argument (i.e. the linearization of the first subtree) is concatenated (`++`) to the second argument.

The `fun` and `lin` rules together correspond to the context-free rule

```
Cl ::= NP VP
```

and decompose it to a tree-building rule and a string-producing rule. The decomposition makes it possible to build multilingual grammars with shared trees and different strings.

However, to deal with the differences of languages, we need linearization rules that don't just operate on strings but also on **tables** that encode the inflectional forms of words and phrases, and on **records** that store different kinds of grammatical information. We don't want this kind of information enter the abstract syntax, because it is language-specific.

Thus in Chinese, it is enough to linearize noun phrases to strings,

```
lin i_NP = "wo"
```

But in English, noun phrases are linearized to records that have two fields: one labelled `s` ("string"), which contains an inflection table with nominative and accusative cases, and one labelled `a` ("agreement"), which contains a record that in turn contains a number `n` and a person `p`):

```
lin i_NP = {
  s = table {Nom => "I" ; Acc => "me"} ;
  a = {n = Sg ; p = Per1}
  }
```

The linearization rule of `PredVP` uses the information in the record to select the nominative case for the subject and guarantee that the verb phase agrees to the subject:

```
lin PredVP np vp =
  np.s ! Nom ++ vp ! np.a
```

(The notation `np.s` computes the `s` part from the record `np`, and `vp ! np.a` computes the value for `np.a` from the table `vp`.)

Just like ASTs, linearizations thus have types, but these types are dependent on language. The linearization type of the category `NP` in Chinese is defined by the rule

```
lincat NP = Str
```

whereas in English a more complex type is needed,

```
lincat NP = {
  s : Case => Str ;
  a : {n : Number ; p : Person}
  }
```

marking a record that holds a table and the agreement features.

Linearization types vary greatly from one language to another, partly because of morphology; for instance, Finnish noun phrases have 15 cases. But even Chinese, which has no morphological variation, is not entirely context-free (string-based). If we want to keep the common abstract syntax, we need to use records to encode **discontinuous constituents**, that is, phrases in which later functions insert new words. An example is question formation by verb reduplication. The linearization types involved are

```
lincat
  QCl = Str
  Cl = {subj : Str ; vp : VP}
  VP = {verb : Str ;
        neg : Str ; obj : Str}
```

The `neg` part of the `VP` is *bu* or *mei*, depending on verb. The question forming function is linearized as follows with reduplication:

```
lin QuestCl cl =
  cl.subj ++ cl.vp.verb ++ cl.vp.neg ++
  cl.vp.verb ++ cl.obj
```

(Questions with particle *ma* could be given as an alternative linearization.)

Multilingual grammars are a generalization of **synchronous grammars** (Aho and Ullman,

1969), originally defined for context-free grammars but later generalized to tree-adjoining grammars (TAG) (Shieber and Schabes, 1990). GF adds to synchronous grammars the explicit notion of abstract syntax, which has replaced the direct transfer of synchronic grammars in modern compiler construction (Appel, 1998). Tables and records are related to unification grammars (Shieber, 1986), but the expressive power of GF is lower: it is equivalent to PMCFG (parallel multiple context-free grammars) (Seki et al., 1991), which enjoys polynomial parsing. The word "parallel" in PMCFG means that an expression may be duplicated in linearization. This is not needed in all languages, but Chinese reduplication questions are an example of it.

3 The GF toolset

The GF set of tools has several components:

- The **GF programming language** and its compiler (Ranta, 2010).
- **PGF, Portable Grammar Format**, the "machine language of GF" generated by the GF compiler (Angelov et al., 2009).
- **Runtime interpreters** for PGF, enabling mobile and web applications (Ranta et al., 2010; Angelov et al., 2014).
- The **Resource Grammar Library** (RGL), currently comprising 30 languages (Ranta, 2009).
- A **wide-coverage translation system** (Hallgren, 2014 2015).
- **Controlled language applications** (Angelov and Ranta, 2009).
- **Conversions** of GF grammars and trees to other formats, such as speech recognition grammars (Bringert, 2007), finite state automata in the Xerox format (Beesley and Karttunen, 2003), dependency trees in the CoNLL format (Eisner, 2007), and phrase tables in the Giza++ format (Och and Ney, 2003).

The last item, conversions, guarantees that GF is not a closed world, but that GF grammars can be reused in other ecosystems. The advantage of GF is that it enables programming on a higher level than e.g. hand-written speech recognition grammars (Perera and Ranta, 2007). This is essential in order for grammar writing to be competitive with machine learning and statistics. Even in statistical systems, writing grammars can be a way to compensate for the lack of data (Jonson, 2006).

Figure 1: Languages in GF RGL.

The key for a language to enter the GF ecosystem is an RGL implementation. The RGL has a **core abstract syntax** consisting of 86 categories and 216 functions. In addition to this, it a test lexicon of 524 word senses. A language implementation with linearizations for all these functions is accessible in all parts of the "GF ecosystem", via the common abstract syntax.

Figure 1 shows the languages currently available in the RGL. The 14 innermost languages, connected with lines with the abstract syntax, have a large lexicon enabling wide-coverage translation (see Section 6). The layer around them contains 16 languages, which also have complete RGL implementations. The outermost 6 languages have partial RGL implementations and could be completed in weeks or a couple of months.

4 The Chinese grammar

The Chinese resource grammar was started in 2012, as the third East-Asian language of the RGL, after Thai and Japanese, and as the 25th language altogether. (Peng, 2013) gives some details of the first version of the grammar.

Since the abstract syntax of the RGL was originally designed for European languages (English, French, Russian, German, Swedish, Finnish), the question was how well this structure fits on an East-Asian language. Due to the expressive power of GF's linearization rules, it is usually possible to tweak the grammar to work. But if the abstract syntax does not fit well, the grammar needs lots of artificial parameters that make the code more complex than with a more native tree structure. In this respect, Japanese has turned out to be one of the most difficult languages (Zimina, 2012).

language	LoC	CF rules	CF/GF
Chinese	1200	317	7
English	1800	18998	432
Finnish	3000	137558	3126
French	3600	152632	3469
Japanese	3700	4521	103

Table 1: The complexity of some RGL implementations. LoC = lines of GF source code in core RGL (216 functions). CF rules = number of context-free rules generated from a set of 44 functions from the resource grammar.

4.1 How complex is Chinese grammar?

In the case of Chinese, the fear of complexity turned out to be unnecessary. The total core RGL implementation for Chinese has 1,200 lines of code, which is less than for most other languages; see Table 1 for some examples. The amount of code reflects both the inherent complexity of the language (in particular morphology) and the fit of the abstract syntax.

The common abstract syntax of the RGL hides the morphological variation completely, but linearization rules have to address it with tables and records. But even on this level, the abstractions provided by functional programming keep the code sizes similar for different languages, as shown in Table 1.

To get another view on the complexity, one can look at the size of the context-free expansions of the languages. Table 1 gives the number of rules in context-free expansions for the core 44 rules of the resource grammar, together with the context-free/GF rule ratio. As the table also shows the source code size for each language, it gives an idea of the compression that GF grammars achieve in comparison to actual language data. The expansion algorithm is defined in (Bringert, 2007); the result is still only approximative because GF is not context-free. The figures say that every Chinese GF rule can be approximated by just 7 context-free rules, whereas French needs over 3000 context-free rules on the average! The explosion is a multiplicative effect of the parameters involved in the argument and value types of syntactic combination functions. Nevertheless, the source code for French is just 3 times the source code for Chinese.

4.2 Linguistic phenomena

We have already mentioned reduplication as a feature of Chinese that needs attention. Another characteristic feature are **classifiers** attached to common nouns (`CN`) and used in combinations with determiners (`Det`). They can be controlled by a linearization type that has a field for the classifier,

```
lincat CN = {s : Str ; c : Str}
```

The determination rule,

```
fun DetCN : Det -> CN -> NP
```

places the classifier between the determiner and the noun,

```
lin DetCN det cn = det ++ cn.c ++ cn.s
```

Since adjectives and even relative clauses are prefixed to the noun, the classifier can end up arbitrarily far from the noun that it depends on. This is a problem in chunk-based approaches to translation (see Section 5), but not in a proper grammar.

Another feature of Chinese that needed attention in the RGL is the position of adverbials, which need a parameter classifying them to time, place, and manner. Each of the classes has a different place in a sentence.

4.3 Segmentation

Since Chinese sentences are written without spaces between words, word segmentation is an important task in Chinese NLP (see e.g. (Wong et al., 2009)), needed as a preprocessor for almost any application. The Chinese RGL grammar solves this in the simplest possible way: with no preprocessing at all. Thus the GF parser reads Chinese input character by character, treating each character as a token, and tries to build the AST from this input. When the AST is constructed, word boundaries can be read out from it as a by-product.

One advantage of the method is that only grammatically possible word segmentations are returned. Another advantage is that all grammatically possible segmentations are accessible to the parser, while pre-processing segmentation, typically based on less information, might throw away grammatically correct segmentations.

Random testing with grammar-generated data suggests that the method does not slow down the parser significantly, and that different segmentation are not very frequent if they are required to be grammatically possible. However, this by-product

of the Chinese GF grammar remains to be evaluated with real data.

5 Controlled language applications

GF was originally designed as a tool for CNL (Controlled Natural Language). In our sense of the word, a CNL is any fragment of natural language that has a precise grammar and can therefore be processed mechanically. The abstract syntax of a CNL is typically built on semantic grounds, so that the ASTs are more like logical formulas than linguistic syntax trees. In this way the grammars can be made more precise and also more idiomatic, because logical meanings are often expressed by different syntactic means in different languages.

Since the RGL takes care of linguistic details such as inflection and word order, GF is a productive way to implement CNLs: linearization rules are written in terms of RGL trees instead of records and tables. A typical CNL can in this way be implemented in a few days (Hallgren et al., 2012). Porting it to new languages is even quicker, because the same RGL functions can be reused most of the time. A new language can often be added to a CNL system in a few hours, which makes it easy to build multilingual systems.

Two major CNLs in GF have been ported to Chinese:

- Attempto Controlled English, a CNL for knowledge representation and reasoning, also available as a multilingual semantic wiki system, (Kaljurand and Kuhn, 2013).
- The MOLTO Phrasebook, a CNL supporting idiomatic translation of tourist phrases, also available as a mobile app (Ranta et al., 2012).

Many other CNLs have been created in the European MOLTO project (Caprotti, 2010 2013) and in other academic and commercial projects. This line of work might be the commercially most promising use of GF, since it can satisfy the needs of companies having to produce multilingual information rapidly and accurately, for instance for e-commerce purposes. For this purpose, the vocabulary and syntax may be restricted enough to support a CNL, and GF can easily make them multilingual. The grammar that is used for translation can also be easily converted to a query interpreter, and the abstract syntax is easy to link with other semantic information, e.g. web ontologies (Damova et al., 2014).

A formalized multilingual grammar can also help learners of foreign languages. For learning a language with a different word order and morphology (or the lack of morphology, as in Chinese), a multilingual grammar can be just the right thing. The grammar need not cover the whole language, but just the key structures in an accurate way. The grammar can support learning by translation, which has been proved an efficient way to learn and to teach foreign languages (Cook, 2010). The teacher can use the grammar to produce an infinite variation of sentence pairs as examples or exercises and show their correspondences and contrasts accurately by using alignments produced from the common AST. Figure 2 shows the word alignments between two different PP attachments: one with the bread, the other with the act of eating; Chinese places the PP to front the element that it modifies. Alignment illustrations like this are automatically generated by GF.

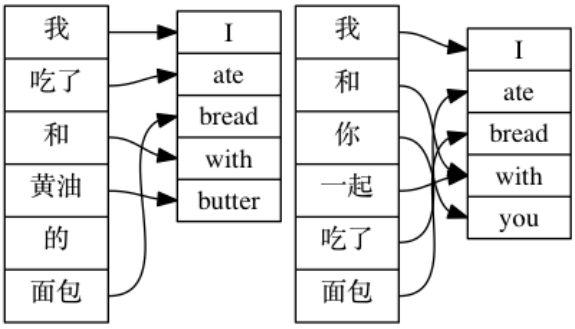


Figure 2: Word alignments for two PP attachments, automatically generated in GF.

6 Wide-coverage lexicon and translation

Unlike a typical CNL, the RGL is not domain-specific but tries to cover the whole language. Thus it is interesting to check if it can be used for "translating anything", like main-streams translation tools do. The GF Wide Coverage Translator, WCT (Hallgren, 2014 2015), is based on the RGL and the following additions:

- **Large lexicon**, with 66,000 word senses.
- **Syntax extensions**, structures not covered by the core RGL.
- **Back-up strategy for parsing**, to guarantee that the system always yields a result.
- **Disambiguation strategy**, to select from a potentially large number of syntax trees.

The current WCT covers 14 languages (Hallgren, 2014 2015). As it uses the ASTs as an interlingua, it enables the translation for $13 \times 14 = 182$

language pairs, 26 of which include Chinese; the languages are shown in Figure 1. Each languages implements at least 20,000 of the word senses.

Extending an RGL implementation to a baseline wide-coverage translator is a small task compared to building the RGL itself: a working system can be built in a few days, if a suitable word list is available. Only the lexicon needs to be implemented separately for each language: the rest is done on the level of the common abstract syntax.

The first version of the Chinese dictionary in WCT was built manually by a class of Chinese undergraduate students, covering 15,000 word senses. It was later completed by 20,000 more words from the Wiktionary. The quality of the automatically added words is lower than the manual words, and continuous checking and revision is a part of the workflow of improving the translator.

In addition to a dictionary, wide coverage translation needs syntax extensions in order to cover structures that are not in the core RGL. This could be done through precise linguistically analysis, like the RGL itself. But a cheaper way to increase coverage is to introduce a **chunking grammar**: a set of rules that enable chunk-by-chunk translation in cases where the entire sentence cannot be covered by a syntax tree.

The quality of chunk-based translation is generally lower than fully syntactic translation. Since there are by definition no grammatical dependencies between chunks, two kinds of errors arise:

- **Agreement**: a chunks cannot determine the features of another chunk.
- **Word order**: the syntactic roles of the chunks are not defined.

The agreement problem is not so visible in Chinese as in European languages, because of the lack of morphology. But a related problem is the choice of classifiers: if *five* and *cats* end up in different chunks when translating

I have five black cats

the result is likely to be

wo you wu ge hei mao

using the most frequent classifier *ge*, rather than

wo you wu zhi hei mao

using the proper cat classifier *zhi*. These effects are familiar from phrase-based statistical translation, where sentences are also built from chunks. The former translation is actually the result from Google translate on the date of writing this, whereas GF produces the latter one due to complete syntactic analysis.

As for word order, a typical problem is the placement of adverbs. In many European languages, adverbs such as the place are at the end of the sentence, but in Chinese, before the verb. A full syntactic analysis is able to "move" the adverb to the right place, but mere chunking cannot do this.

Since Chinese places prepositional phrases in front of they modify (Figure 2), English PP attachment is an ambiguity that cannot be solved by syntax alone: parsing provides both analyses and their linearizations, but it cannot select the correct one, even in clear cases like those in Figure 2. For the final disambiguation, either deeper semantic analysis or an accurate statistical model is needed.

Semantic analysis is easy to implement in a CNL but hard to scale up. Thus the WCT uses statistical disambiguation based on probabilities estimated from the Penn treebank (Marcus et al., 1993). The Penn trees are converted to abstract syntax trees of the RGL, and the frequencies of functions are computed (Angelov, 2011). As the trees are common to all the 30 languages of the RGL, the same model can be used for all of them. But a more adequate model would of course be expected from native treebanks, such as the Chinese Penn treebank (Xue et al., 2005), which remains as future work.

The WCT can be optimized for a special domain by combining it with an **Embedded CNL** (Ranta, 2014a). This means that CNL analyses are given priority over syntactic and chunk-based analyses, whenever available. The translator then generates high quality whenever the input matches the CNL; when not, the other analyses work as a back up that makes the translation robust. The mobile app (Angelov et al., 2014) and the web application (Hallgren, 2014 2015) mark the translations with colours, using green for CNL translations, yellow for syntactic translations, and red for chunk translations. Figure 3 shows the differences between them in the current system: the uppermost, green translation is perfect and idiomatic (using the MOLTO Phrasebook); the middle, yellow translation is syntactically correct but does not capture the meaning of the idiom; the third, red translation results from grammatically incor-

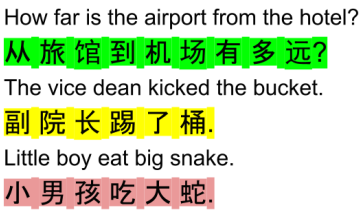


Figure 3: Embedded CNL translation with syntactic and chunk-based back-ups.

rect input manages to render it chunks of intelligible Chinese.

The clearest advantage of grammars in translation is perhaps their compact size. The whole mobile app for 14 languages and 182 language pairs fits in a 35-megabyte binary file, which runs off-line in a mobile phone. Statistical translators, such as Google, are usually run over the internet; downloading stripped-down versions on Android phones is possible, but requires 200 megabytes per language pair.

7 Evaluation

The wide range of applications of GF creates several things to evaluate. Let us address what is perhaps the most frequent question: translation quality with the usual metrics BLEU and TER. Table 2 shows the first results from an evaluation campaign for English to Chinese translation on two different levels: semantic CNL (the MOLTO phrasebook) and wide-coverage GF translation (WCT). In these evaluations, we have machine-translated a set of sentences and created the reference translations by human post-editing. The CNL sentences come from a MOLTO test suite, whereas the WCT sentences are from news (50%), Europarl (25%), and fiction (25%). The table shows comparisons with systems (in WCT, Moses trained with United Nations data). For the CNL, is also shows Swedish and English comparisons.

As expected, GF outperforms the general-purpose Google translate in the CNL, even though Google can be quite good at idiomatic tourist phrases. The Finnish and Swedish CNLs get better scores because more work has been put to them. In the WCT, Moses is better than GF. It is too early to say how competitive GF can be made in this scenario, but an interesting case would be the translation between Chinese and some other language than English, with less parallel data available to build statistical systems from. GF translation is not affected by this problem.

task	BLEU	TER
CNL en-zh, GF	**84**	**9.5**
CNL en-zh, Google	50	35
CNL en-sv, GF	**96**	**1.7**
CNL en-sv, Google	61	19
CNL en-fi, GF	**89**	**5.3**
CNL en-fi, Google	44	33
WCT en-zh, GF	21	62
WCT en-zh, Moses	**36**	**43**

Table 2: First evaluation results for CNL (MOLTO Phrasebook) and WCT (the GF wide-coverage translator).

It can be objected that the comparison between GF and Google translate is not fair in the CNL case, because the GF grammar was specifically tailored for the domain. But this is in fact the very point: since GF grammars are easy to adapt to specific domains, they are a useful technique when high quality is expected and the coverage can be limited. This way of using grammars has also shown commercial potential (Ranta et al., 2015).

8 Related work

The Chinese Penn Treebank (Xue et al., 2005) has been used for building grammars. In particular, (Yu et al., 2010) measures the accuracy and coverage of a generated HPSG grammar, and also lists smaller HPSG projects on Chinese. (Zhang et al., 2012) reports on a more comprehensive HPSG grammar and treebank. As for translation, several systems exist between English and Chinese, but for some of the languages in the GF WCT, e.g. Bulgarian and Finnish, only partially documented commercial systems (such as Google translate) are available. As for CNL, (Cardey et al., 2004) makes a suggestion for medical English-Chinese translation, but we haven't found complete CNL systems for Chinese other than those in GF.

9 Conclusion

We have shown the main ideas of GF and how they can be applied in NLP. The most mature applications are controlled-language tasks such as dissemination translation, language teaching, and natural language queries. Such task have commercial potential, and grammars gives full control on quality. GF makes the use of grammars fea-

sible with its engineering tools and its library of 30 languages. The abstract structures originally created for European languages have proven to work for Chinese as well. GF also scales up to wide-coverage translation, but is not yet competitive with statistical methods. The main advantage in this task is the compact size of the system, making it possible to use 182 language pairs off-line in a mobile device.

References

Steven P. Abney. 1991. Parsing by chunks. In *Principle-Based Parsing*, pages 257–278. Kluwer Academic Publishers.

Alfred V. Aho and Jeffrey D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1):37 – 56.

Krasimir Angelov and Aarne Ranta. 2009. Implementing Controlled Languages in GF. In Norbert Fuchs, editor, *Workshop on Controlled Natural Language, CNL 2009*, volume 5972 of *LNCS/LNAI*.

Krasimir Angelov, Björn Bringert, and Aarne Ranta. 2009. PGF: A Portable Run-Time Format for Type-Theoretical Grammars. *Journal of Logic, Language and Information*.

Krasimir Angelov, Björn Bringert, and Aarne Ranta. 2014. Speech-enabled hybrid multilingual translation for mobile devices. *EACL'14*, pages 41–44.

Krasimir Angelov. 2011. *The Mechanics of the Grammatical Framework*. Ph.D. thesis, Chalmers University of Technology.

Andrew Appel. 1998. *Modern Compiler Implementation in ML*. Cambridge University Press.

Ken Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Publications.

Joan Bresnan, editor. 1982. *The Mental Representation of Grammatical Relations*. MIT Press.

Björn Bringert. 2007. Speech Recognition Grammar Compilation in Grammatical Framework. In *SPEECHGRAM 2007: ACL Workshop on Grammar-Based Approaches to Spoken Language Processing, June 29, 2007, Prague*.

Olga Caprotti. 2010–2013. MOLTO: Multilingual Online Translation. `http://www.molto-project.eu`

Sylviane Cardey, Peter Greenfield, and Xiaohong Wu. 2004. Designing a controlled language for the machine translation of medical protocols: The case of english to chinese. In *Machine Translation: From Real Users to Research, 6th Conference of the Association for Machine Translation in the Americas, AMTA 2004, Washington, DC, USA, September 28-October 2, 2004, Proceedings*, pages 37–47.

Guy Cook. 2010. *Translation in Language Teaching*. Oxford University Press.

Mariana Damova, Dana Dannélls, Ramona Enache, Maria Mateva, and Aarne Ranta. 2014. Multilingual natural language interaction with semantic web knowledge bases and linked open data. In *Towards the Multilingual Semantic Web*, pages 211–226. Springer Berlin Heidelberg.

Marc Dymetman, Veronika Lux, and Aarne Ranta. 2000. XML and multilingual document authoring: Convergent trends. In *Proc. Computational Linguistics COLING, Saarbrücken, Germany*, pages 243–249. International Committee on Computational Linguistics.

Jason Eisner, editor. 2007. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Association for Computational Linguistics, Prague, Czech Republic, June.

Mikel L Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144.

Thomas Hallgren, Aarne Ranta, John Camilleri, Grégoire Détrez, and Ramona Enache. 2012. Grammar Tools and Best Practices. MOLTO Deliverable D2.3, June.

Thomas Hallgren. 2014–2015. GF Wide Coverage Translation Demo. `cloud.grammaticalframework.org/wc.html`

Rebecca Jonson. 2006. Generating statistical language models from interpretation grammars in dialogue system. In *Proceedings of EACL06, Trento, Italy*.

Aravind Joshi. 1985. Tree-adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press.

Kaarel Kaljurand and Tobias Kuhn. 2013. A Multilingual Semantic Wiki Based on Attempto Controlled English and Grammatical Framework. In Philipp Cimiano, Oscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph, editors, *The Semantic Web: Semantics and Big Data. 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, volume

7882 of *Lecture Notes in Computer Science*, pages 427–441. Springer Berlin Heidelberg.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Chen Peng. 2013. Implementation of a chinese resource grammar in grammatical framework. *International Journal of Knowledge and Language Processing*, 4(1):26–34.

Nadine Perera and Aarne Ranta. 2007. Dialogue System Localization with the GF Resource Grammar Library. In *SPEECHGRAM 2007: ACL Workshop on Grammar-Based Approaches to Spoken Language Processing, June 29, 2007, Prague*.

Carl Pollard and Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Aarne Ranta, Krasimir Angelov, and Thomas Hallgren. 2010. Tools for multilingual grammar-based translation on the web. In *Proceedings of the ACL 2010 System Demonstrations*, pages 66–71. Association for Computational Linguistics.

Aarne Ranta, Ramona Enache, and Grégoire Détrez. 2012. Controlled Language for Everyday Use: the MOLTO Phrasebook. In Tobias Kuhn and Norbert Fuchs, editors, *Controlled Natural Language*, volume 7427 of *LNCS/LNAI*. Springer.

Aarne Ranta, Christina Unger, and Daniel Vidal Hussey. 2015. Grammar engineering for a customer: a case study with five languages. In *GEAF-2015: ACL 2015 workshop on Grammar Engineering across Frameworks*. Association for Computational Linguistics.

Aarne Ranta. 2004. Grammatical Framework: A Type-Theoretical Grammar Formalism. *The Journal of Functional Programming*, 14(2):145–189.

Aarne Ranta. 2009. The GF Resource Grammar Library. *Linguistics in Language Technology*, 2:1–65.

Aarne Ranta. 2010. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications. to appear.

Aarne Ranta. 2011. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford.

Aarne Ranta. 2014a. Embedded controlled languages. In *Controlled Natural Language - 4th International Workshop, CNL 2014, Galway, Ireland, August 20-22, 2014. Proceedings*, volume 8625 of *LNCS*.

Aarne Ranta. 2014b. *Yufa kuangjia wei duo zhong ziran yuyan yufa biancheng (Grammatical Framework: Programming with Multilingual Grammars). Chinese translation by Yan Tian*. Shanghai Jiao Tong University Press.

Manny Rayner, David Carter, Pierrette Bouillon, Vassilis Digalakis, and Mats Wirén. 2000. *The Spoken Language Translator*. Cambridge University Press, Cambridge.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *COLING*, pages 253–258.

Stuart Shieber. 1986. *An Introduction to Unification-Based Approaches to Grammars*. University of Chicago Press.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press.

Kam-Fai Wong, Wenjie Li, Ruifeng Xu, and Zhengsheng Zhang. 2009. *Introduction to Chinese natural language processing*, volume 2 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Kun Yu, Yusuke Miyao, Xiangli Wang, Takuya Matsuzaki, and Jun ichi Tsujii. 2010. Semi-automatically Developing Chinese HPSG Grammar from the Penn Chinese Treebank for Deep Parsing. In Chu-Ren Huang and Dan Jurafsky, editors, *COLING 2010*, pages 1417–1425.

Yi Zhang, Rui Wang, and Yu Chen. 2012. Joint grammar and treebank development for mandarin chinese with hpsg. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12). International Conference on Language Resources and Evaluation (LREC-2012), May 23-25, Istanbul, Turkey*. European Language Resources Association (ELRA), 5.

Elizaveta Zimina. 2012. Fitting a round peg in a square hole: Japanese resource grammar in gf. In Hitoshi Isahara and Kyoko Kanzaki, editors, *Advances in Natural Language Processing*, volume 7614 of *Lecture Notes in Computer Science*, pages 156–167. Springer Berlin Heidelberg.