# CHAPTER ONE

# CONSTRAINT-BASED WORD SEGMENTATION FOR CHINESE

HENNING CHRISTIANSEN AND BO LI

## 1.1 INTRODUCTION

Written Chinese text has no separators between words in the same way as European languages use space characters, and this creates the Chinese Word Segmentation Problem, CWSP: given a text in Chinese, divide it in a correct way into segments corresponding to words. Good solutions are in demand for virtually any nontrivial computational processing of Chinese text, ranging from spellchecking over internet search to deep analysis.

Isolating the single words is usually the first phase in the analysis of a text, but as for many other language analysis tasks, to do that perfectly, an insight in syntactic and pragmatic content of the text is essentially required. While this parallelism is easy for competent human language user, computer-based methods tend to be separated into phases with little or no interaction. Accepting this as a fact, means that CWSP introduces a playground for a plethora of different ad-hoc and statistically based methods.

In this paper, we show experiments of implementing different approaches to CWSP in the framework of CHR Grammars (Christiansen, 2005), that provides a constraint solving approach to language analysis. CHR Grammars are based upon Constraint Handling Rules, CHR (Frühwirth, 1998, 2009), which is a declarative, high-level programming language for specification and implementation of constraint solvers. These grammars feature highly flexible sorts of context-dependent rules that may integrate with semantic and pragmatic analyses. The associated parsing method works bottom-up and is robust of errors and incomplete grammar specifications, as it delivers the chunks and (sub-) phrases that have been recognized also

when the entire string could not be processed. The main contribution of this paper is to demonstrate how different approaches to CWSP can be expressed in CHR Grammars in a highly concise way, and how different principles can complement each other in this paradigm. CHR Grammars may not be an ideal platform for high throughput systems, but can serve as a powerful and flexible system for experimental prototyping of solutions to CWSP.

Section 1.2 gives a brief introduction to the intricacies of CWSP and to CHR Grammars including a background of related work. Next, we begin the applications of CHR Grammars showing the representation of a lexicon in section 1.3, and section 1.4 demonstrates a rudimentary, lexicon-based CWSP method based on a maximum match principle. A splitting of a character sequence into into smaller portions, called maximum ambiguous segments, to be analyzed separately in shown in section 1.5. In section 1.6, we discuss further ideas for approaching CWSP that seem to fit into CHR Grammars, and section 1.7 gives a short summary and a conclusion. This paper is a revised version of (Christiansen and Li, 2011).

## 1.2 BACKGROUND AND RELATED WORK

### 1.2.1 THE CHINESE WORD SEGMENTATION PROBLEM

Chinese text is written without explicit separation between the different words, although periods are unambiguously delineated using the special character "○" which serves no other purpose. The Chinese Word Segmentation Problem, CWSP, is the problem of finding a correct or at least a good splitting of the text into units that, in some reasonable way, can be interpreted as words. However, the Chinese language does not possess the same clear distinction between syntax and morphology as European language normally are assumed to have, and what is considered a semantic unit, a word or a standard phrase is not always obvious.[1] A notion of "natural chunk" has been suggested by Huang *et al.* (2013) as a replacement for "word" together with machine learning techniques for identifying such chunks.

As for other languages, analysis is also made difficult by the fact that certain parts may be left out of a sentence; as opposed to most European languages, even the verb may be left out when obvious from the context, so for example, verbs corresponding to "have" or "be" are seldom needed in a

[1] It may be claimed that the main reason why CWSP is an apparent problem for natural language processing software may be that the current foundations for such software reflect traditional views of European languages, rooted in studies of Latin.

Chinese text. Also, Chinese has almost no inflectional markers. These facts make it even more difficult to use syntactic constraints or cues to guide or validate a given segmentation. The recent textbook Wong *et al.* (2010) contains a good introduction to these difficulties also for non-Chinese speakers; see also the analysis given by Li (2011).

Fully satisfactory solutions to CWSP have not been seen yet. The state of the art among "fairly good" systems use lexicon-based methods, complemented by different heuristics and statistically based methods as well as specialized tools such as name entity recognizers; see, e.g., Wong *et al.* (2010) and Li (2011) for a more detailed overview. A good source of primary literature is the web repository containing all proceedings from the CIPS-SIGHAN Joint Conferences on Chinese Language Processing and previous workshops (CIPS-SIGHAN repository, 2000–). Controlled competitions between different Chinese word segmentation systems have been arranged together with the CIPS-SIGHAN conferences. Reports from the 2010 and 2012 competitions (Zhao and Liu, 2010; Duan *et al.*, 2012) indicate precision and recall figures up to around 0.95 for tests on selected corpora, but it is unlikely that these results will hold on arbitrary unseen (types of) text.

Some general systems for Internet search such as Google[2] and Baidu[3] use their own word segmentation algorithms which are not publicly available; Li (2011) provides some tests and discussion of these approaches.

A few more details of related work are discussed in section 1.6, below.

### 1.2.2 CHR GRAMMARS

CHR Grammars (Christiansen, 2005) add a grammar notation layer on top of Constraint Handling Rules (Frühwirth, 1998, 2009), CHR, analogous to the way Definite Clause Grammars (Pereira and Warren, 1980) are added on top of Prolog. CHR itself was introduced in the early 1990es as a rule-based, logical programming language for writing constraint solvers for traditional constraint domains such as integer or real numbers in a declarative way, but has turned out to be a quite general and versatile forward-chaining reasoner suited for a variety of applications; see Frühwirth (1998); Christiansen (2009, 2014b). The CHR Grammar system and a comprehensive Users' Guide are available on the internet (Christiansen, 2002). We assume the terminology and basic concepts of CHR and Prolog to be known, but

[2] http://www.google.com

[3] http://www.baidu.com; the biggest Chinese web search engine.

the following introduction to CHR Grammars may also provide sufficient insight to readers without this detailed background.

Grammar symbols (terminals and nonterminals) are represented as constraints, decorated with integer numbers that refer to positions in the text, although these are normally kept invisible for the grammar writer. Rules work bottom up: when certain patterns of grammar symbols are observed in the store, a given rule may apply and add new grammar symbols in the same way as a constraint solver may combine and simplify a group of constraints into other constraints. Consider the following example of a grammar given as its full source text.

```
:- chrg_symbols noun/0, verb/0, sentence/0.
[dogs] ::> noun.
[cats] ::> noun.
[hate] ::> verb.
noun, verb, noun ::> sentence.
end_of_CHRG_source.
```

Notice that the information on the left and righthand sides of the rules are opposite to the usual standard for grammar rules. This is chosen to indicate the bottom-up nature of CHR Grammars and to resemble the syntax of CHR. Symbols in square brackets are terminal symbols, and nonterminals are declared as shown in the first source line above and can be used in the grammar rules as shown. Given the query

```
?- parse([dogs,hate,cats])
```

constraints corresponding to the three terminal symbols are created and entered into the constraint store; the three "lexical" rules will apply and add new grammar symbols representing the recognition of two `nouns` and a `verb` in a suitable order, such that the last rule can apply and report the recognition of a `sentence`. The answer is given as the final constraint store which includes the following constraint; notice that the positions (or boundaries) in the string are shown here.

```
sentence(0,3)
```

The rules shown above are *propagation* rules, that work by adding new instances of grammar symbols to those already existing in ehe constraint store. When the arrow in a rule is replaced by `<:>`, the rule becomes a *simplification* rule which will remove the symbols matched on the letfhand side; there is also a form of rules, called *simpagations*, that allow to remove only some of the matched symbols. When simplification and simpagation rules

are used, the actual result of a parsing process may depend on the procedural semantics of the underlying CHR system (i.e., its principles for which rules are applied when), and a knowledge about this is recommended for the grammar writer who wants to exploit the full power of CHR Grammars. A strict use of propagation rules implies a natural handling of ambiguity as all possible analysis are generated in the same constraint store, while simplification rules may be applied for pruning or sorting out among different (partial) solutions.

In some cases, it may be relevant to order the application of rules into phases such that firstly all rules of one kind apply as much as possible, and then a next sort of rules is allowed to apply. This can be done by embedding non-grammatical constraints in the lefthand side of a grammar rule, declared as ordinary CHR constraints. We can illustrate this principle by a modification of the sample grammar above.

```
...
:- chr_constraint phase2/0.

{phase2}, noun, verb, noun ::> sentence.
```

Notice the special syntax with curly brackets, which is inspired by Definite Clause Grammars Pereira and Warren (1980). This means that, in this rule, the constraint `phase2` does not depend on positions in the text, but must be present for the rule to apply. The query for analysis should then be changed as follows.

```
?- parse([dogs,hate,cats]), phase2.
```

This means that first, the lexical rules will apply as long as possible (as they are not conditioned by the constraint `phase2`), and when they are finished, the `sentence` rule is allow to be tried. In this particular example, this technique not change the result, but we give examples below where it is essential.

CHR Grammar rules allow an extensive collection of patterns on the lefthand side for how grammar symbols can be matched in the store: context-sensitive matching, parallel matching, gaps, etc.; these facilities will be explained below when they are used in our examples. As in a Definite Clause Grammar Pereira and Warren (1980), grammar symbols may be extended with additional arguments that may store arbitrary information of syntactic and semantic kinds.

CHR, often in the shape of CHR Grammars, have been used for a variety of language processing tasks until now, but to our knowledge, not to Chinese until the work reported here. Hecksher *et al.* (2002) used a predecessor of

CHR Grammars for analysing hieroglyph inscriptions, Christiansen *et al.* (2007a,b) used it for interpreting use case text and converting it into UML diagrams; van de Camp and Christiansen (2012) and Christiansen (2014a) have used CHR for resolving relative and other time expressions in text into absolute calendric references; Christiansen and Dahl (2003) have made grammatical error detection with error correction; Bavarian and Dahl (2006) have analyzed biological sequence data.

## 1.3 A LEXICON IN A CHR GRAMMAR

We begin the applications of CHR Grammars introducing a lexicon. As in most other grammar formalisms, a lexicon for testing can be represented by a collection of small rules, one for each lexeme. The following sample lexicon are used in the examples to follow.

```
[中]    ::> word([中]).              % centre, middle
[中,华] ::> word([中,华]).           % Chinese (adjective), China
[华,人] ::> word([华,人]).           % Chinese (people)
[人]    ::> word([人]).              % people, human
[人,民] ::> word([人,民]).           % people
[国]    ::> word([国]).              % country
[国,中] ::> word([国,中]).           % high-school
[共,和]  ::> word([共,和]).          % republic
[共,和,国] ::> word([共,和,国]).     % republic, country
[中,华,人,民,共,和,国]
   ::> word([中,华,人,民,共,和,国]).  % People's-republic-of-China
[中,央] ::> word([中,央]).           % central
[政,府] ::> word([政,府]).           % government
[民,政] ::> word([民,政]).           % civil-administration
[中,央,人,民,政,府]
   ::> word([中,央,人,民,政,府]). % People's central government
```

Notice that the grammar contains two rather large words that look like compounds, but which will be included in any dictionary as words as they are known and fixed terms with fixed meanings.

The `word` grammar symbol may be extended with syntactic tags, but for now we will do with the simplest form as shown.

## 1.4 MAXIMUM MATCHING

A first naive idea for approaching CWSP may be to generate all possible words from the input, followed by an assembly of all possible segmenta-

tions that happens to include the entire input, and then a final phase selecting a best segmentation according to some criteria. Obviously, this is of exponential or worse computational complexity, so more efficient heuristics have been developed. One such heuristics is the maximum matching method, which has been used in both forward and backward versions; here we show the forward method; see Wong *et al.* (2010) for background and references. The sentence is scanned from left to right, always picking the longest possible word; then the process continues this way until the entire string has been processed.

Three CHR Grammar rules are sufficient to implement this principle. The first one, which needs some explanation, will remove occurrences of words that are proper prefixes of other word occurrences.

```
!word(_) $$ word(_), ... <:> true.
```

The "`$$`" operator is CHR Grammar's notation for parallel match: the rule applies whenever both of the indicated patterns match grammatical constraints in the store for the same range of positions (i.e., substring). The symbol "`...`" refers to a *gap* that may match any number of positions in the string, from zero and upwards, independently of whatever grammar symbols might be assiciated with those positions.[4] In other words, the pattern "`word(_), ...`" matches any substring that starts with a word. So when this is matched in parallel with a single word, it applies in exactly those cases where two words occur, one being a (not necessarily proper) prefix of the other. The exclamation mark in front of the first `word` indicates that the grammar matched by this one is not removed from the store as is the standard for simplification rules. This is an example of a so-called simpagation rule having the `<:>` arrow (which otherwise signifies simplification), in which all grammar symbols and constraints appearing on the lefthand side marked with "`!`" are kept in the store and all others removed. The `true` on the righthand side stands for nothing, meaning that no new constraints or grammar symbols are added. Thus, when a string is entered, this rule will apply as many times as possible, each time a lexicon rule adds a new word, and thus keeping only longest words.

In a second phase, we compose a segmentation from left to right, starting from the word starting after position 0. The first rule applies an optional notation in "`:(0,_)`", which makes the word boundaries explicit, here used

[4] Gaps are not implemented by matching, but affect how its neighbouring grammar symbols are matched, putting restrictions on their word boundaries. In the example shown, it must hold that $r_1 \geq r_2$ for the rule to apply where $r_1$ and $r_2$ designate the right boundary of the first, resp., the second `word` in the rule head.

to indicate that this rule only applies for a leftmost word. The `compose` constraint is used as described above to control that these rules cannot be applied before all short words have been removed by the rule above.

```
{!compose}, word(W):(0,_) <:> segmentation(W).
{!compose}, segmentation(Ws), word(W) <:> segmentation(Ws/W).
```

Assuming the lexicon given above, we can query this program as follows, shown also with the answer found (with constraints removed that are not important for our discussion).

```
?- parse([中,华,人,民,共,和,国,中,央,人,民,政,府]), compose.
segmentation(0,13,[中,华,人,民,共,和,国]/[中,央,人,民,政,府])
```

Here the method actually produces the right segmentation, meaning "The Central People's Government of the People's Republic of China"; the "of" being implicit in the Chinese text. Notice that there is actually a word spanning over the split, namely the word for high-school that happens to be in the lexicon, cf. section 1.3. This example showed also the advantage of combining the maximum match principle with having common terms or idioms represented as entries in the lexicon.

We can show another example that demonstrates how maximum matching can go wrong. We extend the lexicon with the following rules.

```
[明,确] ::> word([明,确]).          % definitude, clearly
[确,实] ::> word([确,实]).          % really, actually
[实,在] ::> word([实,在]).          % honest, actually
[考,虑] ::> word([考,虑]).          % consider
[将,来] ::> word([将,来]).          % future
[将,来,的] ::> word([将,来,的]).    % future-related
[李]    ::> word([李]).             % Li (family name)
[李,子] ::> word([李,子]).          % plum
[明]    ::> word([明]).             % bright, Ming (given name)
[将]    ::> word([将]).             % will
[在]    ::> word([在]).             % at
[来]    ::> word([来]).             % come
[事]    ::> word([事]).             % thing
```

The sample sentence we want to check is "李明确实在考虑将来的事", which can be translated into English as "Li Ming is really considering the future things" corresponding to the correct segmentation as follows.

```
[李]/[明,确]/[实,在]/[考,虑]/[将,来,的]/[事]
```

Querying the maximum matching program as shown above for this sentence gives the segmentation

```
[李,明]/[确,实]/[在]/[考,虑]/[将,来,的]/[事]
```

that does not give sense to a Chinese reader. The problem is that the first two characters, which together represent a person's name that is not included in the lexicon. Thus, the first character is taken as a word and thus the second and third second character are taken as the next word, and so on. In the middle of the sentence, the program accidentally gets on the right track again and gets the remaining words right. Due to the high frequency of two-character words in Chinese, it is easy to produce quite long sentences where one wrong step in the beginning makes everything go wrong for the maximum matching method.

If instead, in the example above, the two characters for the personal name Li Ming are treated as one unit, everything would go right. This could suggest that a specialized algorithm for identifying personal names will be useful as an auxiliary for CWSP, as it has been suggested among others by Chen *et al.* (2010). We can simulate such a facility by adding a rule for this specific name as follows.

```
[李,明] ::> word([李,明]).  % Li Ming (person name)
```

Finally, we mention that combinations of forward and backward maximum segmentation have been used, and in those regions where the two disagree, more advanced methods are applied; see, e.g., Zhai *et al.* (2009).

## 1.5 MAXIMUM AMBIGUOUS SEGMENTS

Another principle that may be used in algorithms for CWSP is to run a first phase, identifying the maximum ambiguous segments of a text. We have distilled the principle from a variety of methods that apply similar principles; we have not been able to trace it back to a single source, but Wong *et al.* (2010) may be consulted for a detailed review.

An *ambiguous segment* is defined as a contiguous segment *s* in which

- any two contiguous characters are part of a word,
- there are at least two words that overlap, and
- the first and last character are each part of a word entirely within *s*.

For example, if *abcd* and *def* are words, then the substring *abcdef* will form an ambiguous segments, but not necessarily *cdef* or *abcdefg*. An ambiguous segment is *maximal*, a MAS, whenever it cannot be expanded in any

direction to form a larger ambiguous segment. For example, if *abc*, *cde*, *def*, *defg* are words, then the substring *abcdefg* may form a maximal ambiguous segment. Thus, if no unknown words occur in a text, the splits between the MASs will be definitive. Except in construed cases, the length of the MASs are reasonable, which means that we can apply more expensive methods subsequently within each MAS, perhaps even with exponential methods that enumerate and evaluate all possible segmentations.

Identifying these MASs can be done by very few CHR rules. For simplicity, we introduce a grammar symbol `maxap` which covers MASs as well as single words that can only be recognized as such. The following two CHR Grammar rules and an additional Prolog predicate are sufficient to identify the `maxap`s.

```
word(W) ::> maxap.
maxap:R1, ... $$ ..., maxap:R2 <:> overlap(R1,R2) | maxap.

overlap((A1,B1),(A2,B2)):- A1 < B2, A2 < B1.
```

The second rule uses the auxiliary Prolog predicate `overlap` as a guard. The presence of a guard, between the arrow and the vertical bar, means that the rule can only apply in those cases where the guard is true. When this rule applies, the variables `R1` and `R2` will be instantiated to pairs of indices indicating beginning and end of the two given `maxap`s. The `overlap` predicate tests, as its name indicates, whether the two segments in the string occupied by the two input `maxap`s do overlap. This grammar rule will gradually put together ambiguous segments and, via repeated applications, merge together so only maximum ones remain.

We can test this program for the previous example, “Li Ming is really ...” as follows.

```
?- parse([李,明,确,实,在,考,虑,将,来,的,事]).
maxap(0,5)
maxap(5,7)
maxap(7,10)
maxap(10,11)
```

This corresponds to splitting the sequence into the substrings

李明确实在, 考虑, 将来的事,

which then can be analyzed separately.

## 1.6 DISCUSSION

Our main sources on CWSP research (CIPS-SIGHAN repository, 2000–) report also statistically based methods of different sorts, possibly combining with part-of-speech tagging. Part-of-speech tagging can be implemented in a CHR Grammar, but for realistic applications with a huge lexicon, the right solution may be to preprocess the text by a part-of-speech tagger, and the let CHR Grammar use the tags. Named entity recognizers can be integrated in a similar way.

CHR Grammars do not themselves support machine learning, but it is straightforward to integrate probabilities or other weighting schemes (found by other means) into a CHR Grammar: each constituent has an associated weight, and when a rule applies, it calculates a new weight for the compound. Additional rules can be added that prune partial segmentations of low weight. A recent approach to CWSP (Zhang *et al.*, 2013) maps first a text into a binary three that represents alternative segmentations based on a lexicon, and then this tree is pruned based on statistically learned weigths. Comprehensive statistics concerning ambiguity phenomena in Chinese text is reported by Qiao *et al.* (2008), which appears to be very useful for further research into CWSP.

More refined analyses involving particular knowledge about the Chinese language may be incorporated in a CHR Grammar approach to CSWP. For example, the sign "的" (pronounced "de") normally serves as a marker that converts a preceding noun into an adjective; in fact, most adjectives are constructed in this way from nouns which often have no direct equivalent in European languages, e.g., adjective "red" is constructed from a noun for "red things". Thus, what comes before "的" should preferably be a noun.[5] There are of course lots of such small pieces of knowledge that can be employed and should be employed, and we may hope that the modular rule-based nature of CHR can make it possible to add such principles in an incremental way, one by one.

The out-of-vocabulary (OOV) problem, that we did not approach here, may be the obstacle that makes any method, that works well for an isolated and controlled corpus, useless in practice. OOV words are often proper names and it is obvious that a module for recognizing proper names should be included. We have already referred to Chen *et al.* (2010) that suggests an approach to recognize person names, and Wong *et al.* (2010) list several

[5]There are few additional usages of "的" (where it is pronounced "di"), but these are in special words that are expected to be included in any Chinese dictionary.

characteristics than may be applied in identifying also place names, transcription of foreign names, etc. We may also refer to an interesting approach to OOV in CWSP that incorporate web searches (Qiao and Sun, 2009). Li (2011) suggests a method that involves web searches to evaluate alternative suggestions for segmentations which also may improve performance in case of OOV. A recent proposal by Tian *et al.* (2013) applies machine learning techniques to produce a sort of abstract grammar for Chinese words, which thus also handle OOVs. To reduce the complexity induced by the large character sets, characters are mapped into classes based on semantic features, and then the "word grammar" is expressed in terms of identifiers for those classes.

## 1.7 CONCLUSION

It has been demonstrated how different approaches to the Chinese Word Segmentation Problem can be realized in a concise way in the framework of CHR Grammars, that may serve as a flexible platform for experimenting with and testing new approaches to the proble, There is a high demand for efficient and precise solutions due to the vast presence of the Chinese language on the Internet, as well as for Chinese language processing in general. It is also an interesting test case for the versatility of CHR Grammars.

The straightforward lexicon-as-grammar-rules approach that we have applied here, which is perfect for small prototypes, does not scale well to full dictionaries. However, it is easy to get around this problem using an external dictionary and other resources such as a named entity recognizer as preprocessors. So in addition to entering the texts as a character sequence as shown in our examples, it may be accompanied with constraints that represent all possible word occurrences in the text.

With this extension in mind, CHR Grammar based approaches to CWSP may scale reasonably to larger texts due to the unambiguous indication of periods which can be analyzed one by one. CHR Grammars' flexibility may be utilized to incorporate handling of lots of special cases based on a linguistic insight. An important next step is to incorporate methods for handling OOV words.

## BIBLIOGRAPHY

ACL Anthology: A Digital Archive of Research Papers in Computational Linguistics (2000–2013). Special Interest Group on Chinese Language Processing (SIGHAN). *Webarchive with all articles of CIPS-SIGHAN Joint Conference on Chinese Language Processing 2010, Proceedings of the nth SIGHAN Workshop on Chinese Language Processing, $n = 1, \ldots, 7$, 2002–2013, Second Chinese Language Processing Workshop, 2000.* http://www.aclweb.org/anthology/sighan.html. *Link checked March 2013*.

Bavarian, M. and Dahl, V. (2006). Constraint based methods for biological sequence analysis. *Journal of Universal Computing Science*, **12**(11), 1500–1520.

Chen, Y., Jin, P., Li, W., and Huang, C.-R. (2010). The Chinese persons name disambiguation evaluation: Exploration of personal name disambiguation in Chinese news. In *CIPS-SIGHAN Joint Conference on Chinese Language Processing 2010*. Online proceedings, http://aclweb.org/anthology/W/W10/W10-4152.pdf.

Christiansen, H. (2002). CHR Grammar web site; released 2002. http://www.ruc.dk/~henning/chrg.

Christiansen, H. (2005). CHR Grammars. *Int'l Journal on Theory and Practice of Logic Programming*, **5**(4-5), 467–501.

Christiansen, H. (2009). Executable specifications for hypothesis-based reasoning with Prolog and Constraint Handling Rules. *J. Applied Logic*, **7**(3), 341–362.

Christiansen, H. (2014a). Constraint logic programming for resolution of relative time expressions. In A. Beckmann, E. Csuhaj-Varjú, and K. Meer, editors, *Computability in Europe 2014*, Lecture Notes in Computer Science. Springer. To appear.

Christiansen, H. (2014b). Constraint programming for context comprehension. In P. Brézillon and A. Gonzalez, editors, *Context in Computing*. To appear.

Christiansen, H. and Dahl, V. (2003). Logic grammars for diagnosis and repair. *International Journal on Artificial Intelligence Tools*, **12**(3), 227–248.

Christiansen, H. and Li, B. (2011). Approaching the chinese word segmentation problem with CHR grammars. In *CSLP 2011: Proc. 4th Intl. Workshop on Constraints and Language Processing*, volume 134 of *Roskilde University Computer Science Research Report*, pages 21–31.

Christiansen, H., Have, C. T., and Tveitane, K. (2007a). From use cases to UML class diagrams using logic grammars and constraints. In *RANLP '07: Proc. Intl. Conf. Recent Adv. Nat. Lang. Processing*, pages 128–132.

Christiansen, H., Have, C. T., and Tveitane, K. (2007b). Reasoning about use cases using logic grammars and constraints. In *CSLP '07: Proc. 4th Intl. Workshop on Constraints and Language Processing*, volume 113 of *Roskilde University Computer Science Research Report*, pages 40–52.

Duan, H., Sui, Z., Tian, Y., and Li, W. (2012). The cips-sighan clp 2012 chineseword segmentation onmicroblog corpora bakeoff. In *Proceedings of the Second CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 35–40, Tianjin, China. Association for Computational Linguistics.

Frühwirth, T. (2009). *Constraint Handling Rules*. Cambridge University Press.

Frühwirth, T. W. (1998). Theory and practice of Constraint Handling Rules. *Journal of Logic Programming*, **37**(1-3), 95–138.

Hecksher, T., Nielsen, S. T., and Pigeon, A. (2002). A CHRG model of the ancient Egyptian grammar. Unpublished student project report, Roskilde University, Denmark.

Huang, Z., Xun, E., Rao, G., and Yu, D. (2013). Chinese natural chunk research based on natural annotations in massive scale corpora - exploring work on natural chunk recognition using explicit boundary indicators. In Sun *et al.* (2013), pages 13–24.

Li, B. (2011). *Research on Chinese Word Segmentation and proposals for improvement*. Master's thesis, Roskilde University, Computer Science Studies, Roskilde, Denmark. Available at http://rudar.ruc.dk/handle/1800/6726.

Pereira, F. C. N. and Warren, D. H. D. (1980). Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, **13**(3), 231–278.

Qiao, W. and Sun, M. (2009). Incorporate web search technology to solve out-of-vocabulary words in Chinese word segmentation. In *Proceedings of 11th Pacific Asia Conference on Language, Information and Computation (PACLIC'2009)*, pages 454–463.

Qiao, W., Sun, M., and Menzel, W. (2008). Statistical properties of overlapping ambiguities in Chinese word segmentation and a strategy for their disambiguation. In P. Sojka, A. Horák, I. Kopecek, and K. Pala, editors, *TSD*, volume 5246 of *Lecture Notes in Computer Science*, pages 177–186. Springer.

Sun, M., Zhang, M., Lin, D., and Wang, H., editors (2013). *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data - 12th China National Conference, CCL 2013 and First International Symposium, NLP-NABD 2013, Suzhou, China, October 10-12, 2013. Proceedings*, volume 8202 of *Lecture Notes in Computer Science*. Springer.

Tian, L., Qiu, X., and Huang, X. (2013). Chinese word segmentation with character abstraction. In Sun *et al.* (2013), pages 36–43.

van de Camp, M. and Christiansen, H. (2012). Resolving relative time expressions in Dutch text with Constraint Handling Rules. In D. Duchier and Y. Parmentier, editors, *CSLP*, volume 8114 of *Lecture Notes in Computer Science*, pages 166–177. Springer.

Wong, K.-F., Li, W., Xu, R., and Zhang, Z.-S. (2010). *Introduction to Chinese Natural Language Processing*. Morgan and Claypool publishers.

Zhai, F.-W., He, F.-W., and Zuo, W.-L. (2009). Chinese word segmentation based on dictionary and statistics. *Journal of Chinese Computer Systems,*, **9**(1), (No page numbers given).

Zhang, K., Wang, C., and Sun, M. (2013). Binary tree based chinese word segmentation. *CoRR*, **abs/1305.3981**.

Zhao, H. and Liu, Q. (2010). The CIPS-SIGHAN CLP 2010 Chinese Word Segmentation Bakeoff. In *Proceedings of the Joint Conference on Chinese Language Processing*, pages 199–209. Association for Computational Linguistics.