Resolving relative time expressions in Dutch text with Constraint Handling Rules

Matje van de Camp[1] and Henning Christiansen[2]

[1] Tilburg Centre for Cognition and Communication
Tilburg University, The Netherlands
E-mail: M.M.v.d.Camp@uvt.nl
[2] Research group PLIS: Programming, Logic and Intelligent Systems
Department of Communication, Business and Information Technologies
Roskilde University, Denmark
E-mail: henning@ruc.dk

Abstract. It is demonstrated how Constraint Handling Rules can be applied for resolution of indirect and relative time expressions in text as part of a shallow analysis, following a specialized tagging phase. A method is currently under development, optimized for a particular corpus of historical biographies related to the development of the Dutch social movement between 1870 and 1940. It appears that CHR provides a modular language which is well-suited for tailoring a method optimized for the different linguistic constructs and the textual conventions applied in a specific corpus. We explain the principles and illustrate by sample rules. The project is currently in a phase of development and testing, and this paper focuses on the methodological issues, while a detailed evaluation is not relevant at present. Identified problems and possible extensions are discussed.

1 Introduction

Information regarding the social networks that underlie our society's history can provide new insights into known data for social historical research. Automatically extracting such a network from text involves finding and identifying the real world entities that are described in the text, such as persons, organizations, and locations. In order to connect the entity mentions to a timeline so that the evolution of the network can be examined, time expressions need to be detected, normalized, and put in their chronological order.

In this research, we study the problem of resolving the chronological ordering of time expressions found in a collection of biographies. Solving this task will facilitate the identification of events and relationships that involve two or more entities, including the time when the events took place and how long they lasted. Another point of interest is the positioning in time of general events and states of affairs.

In (historical) biographical texts, time may be indicated explicitly ("*June 12, 1876*","*in 1903*") or in a variety of indirect ways ("*on that day*","*May of*

that year"), depending on the writing style of the author and the availability of specific information. We are especially interested in resolving implicit time expressions and representing them as (perhaps imprecise) knowledge.

We use constraint solving techniques for this task, but instead of using a fixed constraint solver (e.g., a traditional finite domain solver) we use the language of Constraint Handling Rules (CHR). With this, we can write a constraint solver that incorporates linguistic knowledge and heuristics, including syntactic, semantic and pragmatic information extracted from the text.

Section 2 gives a short account on related work. Section 3 provides background on the chosen corpus and explains the goals for the analysis. Section 4 gives a gentle introduction to CHR and its application to language analysis. The preprocessing is briefly described in Section 5, and Section 6 explains some of the CHR rules that we have designed for time expressions. Considerations on tests and the current state of the implementation are given in Section 7, and finally we give some conclusions and indicate possible improvements of this technique.

2 Related work

Most standard methods for extraction of temporal information from text rely on machine learning, as exemplified by [1, 2]; see also [3] for an overview and more references. In our approach, we use a rule-based and transparent approach to tailor an extraction method optimized for a particular corpus in order to complete the suggestions produced by a rule-based tagger as the one explained in Section 5.

Applications of CHR for language processing have been demonstrated by different authors for a variety of tasks, such as parsing from Property Grammars [4], building UML diagram from use case text [5, 6], analyzing biological sequences [7, 8] and Chinese Word Segmentation [9]. The technique of using CHR for bottom up parsing is investigated in depth in [10] that also suggests a powerful notation of CHR Grammars on top of CHR; this notation was not used for the present work as CHR Grammars index only by positions in the text, whereas we apply an indexing that distinguishes between individual documents, paragraphs and sentences as well.

Temporal logic has been implemented in CHR [11] but we find that this, as well as other fully fledged calculi or logics concerning time, is far more general and computationally complex than we need here. A generic language processing system using typed feature structures implemented in CHR is described by [12] that demonstrates the use of event calculus for handling time. Instead, we have developed a constraint solver that reflects the particular linguistic constructions concerning time within our chosen corpus and their inherent, varying degrees of imprecision. The problem here does not seem not to be to reason about time, but to reason about context and discourse in natural language text, and here CHR provides a relevant tool [13]. We have not seen earlier work on CHR applied for extracting temporal information from text in a similar way.

3 The case study: Historical biographies in Dutch

The data used in this study is a collection of biographies, in Dutch, detailing the lives of the 575 most notable actors related to the rise of socialism in the Netherlands (± 1870–1940). The biographies describe a coherent group of people and organizations, with many connections between them.

The data set is provided to us by the International Institute of Social History (IISH) in Amsterdam.[3] IISH hosts the biographies online.[4] The documents are presented as separate pages, accessible either through an alphabetized index of person names, or via keyword search using simple Boolean operators. Links between documents are minimal: only the first mention of a person who also has a biography in the collection links to that biography, other entities are not linked. The biographies are accompanied by a database that holds personal information regarding the 575 main characters. It includes their first names, surname, dates of birth and death, and a short description. These details are used to generate the index on the website.

The overall goal is to enhance the biographies in such a way, that new entry points are created into the data, connecting all relevant pieces, and allowing for more serendipitous research to be performed on the data set. We believe that by interpreting the entity mentions as a co-occurrence network, which includes entities of types organization and location, and attaching it to a timeline, new perspectives are opened up that are not necessarily person-centric, as is the case now.

Part of the overall task is to identify how certain phenomena are related to time, e.g., particular events that are related to a specific date or entity attributes that hold for some period of time. The data contains time expressions of various types and formats. We restrict ourselves to resolving dates and intervals spanning days, months, or years. Time expressions with a finer granularity are not considered.

Globally, two types of temporal expressions are found in the biographies: specific dates, where day, month, and year are all known, and non-specific dates, where at least one of the attributes is unknown. Besides the distinction between specific and non-specific, another form of ambiguity exists in the expression of time in text. Dates can be given explicitly, as for instance "14 November 1879", or implicitly, as "14 November of that year" or even "on that day". In the latter case, the time expression can only be resolved in reference to one or more (non-) specific dates mentioned elsewhere in the text. Also, for each instance, we need to determine if it concerns a reference to a single point in time, or if it denotes the start or end of (or even some point during) an interval.

Ultimately, we aim to represent all explicit and implicit references to specific and non-specific dates or pairs thereof as points and intervals anchored to the timeline underlying all biographies, where we leave room for imprecisely defined dates.

[3] http://www.iisg.nl/
[4] http://www.iisg.nl/bwsa/

4 Constraint Handling Rules and text analysis

Constraint Handling Rules [14, 11], for short CHR, is an extension to Prolog that introduces bottom-up, forward chaining computations. Operationally, CHR is defined as rewriting rules over a constraint store, which can be seen as a global resource to be used by a Prolog program for storing, manipulating and consulting different hypotheses.

CHR inherits the basic nomenclature and syntactic conventions of Prolog such as variables written with capital letters and terms as a generic representation. The constraints of CHR can be called from Prolog programs and whenever a constraint is called, it is added to the constraint store, and the CHR rules supplied by the programmer apply as long as possible. CHR provides the following sorts of rules.

$$\text{Simplification rules: } c_1, \ldots, c_n \texttt{ <=> } \mathit{Guard} \mid c_{n+1}, \ldots, c_m$$
$$\text{Propagation rules: } c_1, \ldots, c_n \texttt{ ==> } \mathit{Guard} \mid c_{n+1}, \ldots, c_m$$

The c's are atoms that represent constraints, possibly including variables. What is to the left of the arrow symbols is called the *head*, and what is to the right of the guard the *body*.

A simplification rule works by replacing in the constraint store, a possible set of constraints that matches the pattern given by $c_1, \ldots c_n$, with the constraints given by $c_{n+1}, \ldots, c_m$, however, only if the condition given by *Guard* holds. A propagation rule executes in a similar way, but it does not remove the head constraints from the store. The declarative semantics is hinted by the applied arrow symbols (bi-implication, resp., implication formulas, with variables assumed to be universally quantified) and it can be shown that the indicated procedural semantics agrees with this.

CHR provides a third kind of rules, called *simpagation rules*, which can be thought of as a combination of the two or, alternatively, as an abbreviation for a specific form of simplification rules.

$$\text{Simpagation rules: } c_1, \ldots, c_i \setminus c_{i+1}, \ldots c_n \texttt{ <=> } \mathit{Guard} \mid c_{n+1}, \ldots, c_m$$
$$\text{which can be though of as: } c_1, \ldots, c_n \texttt{ <=> } \mathit{Guard} \mid c_1, \ldots, c_i, c_{n+1}, \ldots, c_m$$

When this rule is applied, $c_1, \ldots, c_i$ stay in the constraint store and $c_{i+1}, \ldots, c_n$ are removed. In practice, the body of a rule can be any executable Prolog code.

It is straightforward to write bottom-up parsers in CHR when the strings to be analyzed, as well as recognized phrases, are represented as constraints with their position in the string represented by constraints carrying word boundaries as attributes. The following sample rule could be part of a simplistic natural language parser.

```
nounPhrase(N0,N1), verbPhrase(N1,N2) ==> sentence(N0,N2).
```

Informally it reads: if a noun phrase has been recognized between positions `N0` and `N1`, and a verb phrase between `N1` and `N2`, then we recognize a sentence

between `N0` and `N2`. These `N`x variables are assigned natural numbers that represents a numbering of the spaces between the words. It is possible to extend this principle to model various context sensitive dependencies, as rules can also refer to non-consecutive elements and other constraints that represent semantic and pragmatic information. Rules can describe both shallow and deep analyses and flexible combinations thereof, and it is also possible to control in detail, the order in which rules are applied using special constraints that serve as triggers.

5 The starting point: Tagged text

The data is preprocessed in a number of ways. First, the text is lemmatized and tagged with part-of-speech information using Frog, a morpho-syntactic parser for Dutch [15]. Next, proper nouns are automatically identified and classified into types person, organization, and location, using the Stanford Named Entity Recognizer [16], which we have retrained for Dutch named entities. The entity mentions are then clustered to separate the distinct real world entities that they refer to. Finally, temporal expressions are identified and normalized using HeidelTime [17], a rule-based temporal tagger. An in depth analysis of the preprocessing stage will be given in a forthcoming article. For now, we briefly expand on the detection and normalization of temporal expressions.

5.1 Normalization of temporal expressions

HeidelTime was originally developed for German and English by Strötgen & Gertz (2012) [17]. Because the system is rule-based, its functionality can easily be extended by adding new sets of rules. We have demonstrated so by adding a rule set for the extraction and normalization of Dutch time expressions. HeidelTime extracts the temporal expressions by matching them and their immediate context to predefined patterns of regular expressions and converting them to a uniform format (TimeML[5]), making it easier to reason over them.

HeidelTime recognizes four types of temporal expressions: *date*, *time*, *duration*, and *set* [18]. We consider only types *date* and *duration* for this experiment. Both include normalization rules for expressions with various degrees of specificity. For example, "halverwege de jaren negentig" (mid 90's) is initially normalized to `'XX95-XX-XX'`, where day, month, and century are left undefined. A single mention of "juni" (June) is normalized to `'XXXX-06-XX'`. In a next step, HeidelTime applies some basic reasoning to these cases, trying to resolve them by relating them to either the document creation time, or the time expressions that have occurred previously in the text. If these do not provide sufficient information, the system will look at linguistic cues, such as verb tense, to determine how the expression is related to its context.

However, this is not a guaranteed method, especially not when dealing with historic documents. The document creation time in this case cannot be trusted,

[5] http://www.timeml.org

since there is an obvious gap between the creation time and the time when the described events occurred. `'XX95-XX-XX'` might mistakenly be normalized to `'1995-XX-XX'`, while in light of the domain under consideration, it is far more likely to refer to the year 1895.

A downfall of adhering to the TimeML format is that some information is lost in translation. A specific mention of the year "95" is normalized to the exact same tag as the example above, losing the distinction between a more or less exact point in time and a slightly more ambiguous version thereof. As the ambiguity of the expression increases, so does the severity of the information loss. Expressions such as "een paar dagen later" (a few days later) or "enkele jaren daarna" (several years later), are normalized to descriptive, imprecise tags. In this case both expressions are normalized to `'FUT_REF'`, which does not reflect the most important difference: their unit of calculation. This may not form such a huge problem in the analysis of, for instance, news articles, where the described events are likely to span only a short period. For the current task of resolving and *ordering* time expressions that span decades, however, this information is crucial.

5.2 From tags to constraints

After preprocessing, the tagged text is converted into constraints of the following form, one for each recognized token in a text:

$$\begin{aligned}&\texttt{token(}\mathit{Doc}\texttt{,}\mathit{Par}\texttt{,}\mathit{Sent}\texttt{,}\mathit{WordStart}\texttt{,}\mathit{WordEnd}\texttt{,}\\&\quad\mathit{NamedEntity}\texttt{,}\mathit{TimeML}\texttt{,}\mathit{TimeId}\texttt{,}\\&\quad\mathit{Token}\texttt{,}\mathit{PosTag}\texttt{,}\mathit{Lemma}\texttt{)}\end{aligned}$$

The *WordStart* and *WordEnd* attributes are word boundaries, as explained above, relative to a specific sentence within a specific paragraph of a specific document (i.e., a biography) as indicated by the first three attributes. The attributes *Token*, *PosTag*, and *Lemma* indicate a token, its part-of-speech tag, and its stem. The actual tense, mood, etc., are ignored since we are only interested in extracting specific pieces of information and not a full syntactic and semantic analysis of the text. The *NamedEntity* attribute represents whether the token is part of a named entity and, if so, which type of entity. The *TimeML* attribute contains the output received from HeidelTime, if the token is part of a temporal expression. The *TimeId* attribute contains, when relevant, the TimeML representation converted to a Prolog term of the form `date(`*Century*`,`*Decade*`,`*Year*`,`*Month*`,`*Day*`)`, using a `null` value to represent components that are not specified.

Since the biographies have already been processed automatically, we may expect occasional parsing or tagging errors. In the present experiment, we take the processed data as they are.

6 Examples of CHR rules for resolving time expressions

We are interested in identifying expressions that refer – either explicitly or indirectly through ana- or even kataphora of different sorts – to time points and

time periods (or intervals), and resolving them as reliably as possible to actual time. The relationship between these time expressions and particular events or states of affairs mentioned in the surrounding text is a different issue that we do not consider in the present paper.

Biographies have some specific features that we can anticipate and rely on. For time expressions, we may in some cases encounter a degree of uncertainty, either due to lack of precise information, or because the biographer judges the exact date to be irrelevant. On the other hand, we can expect that there is real history underneath, i.e., references to real time points and intervals. This implies that we can count on some general properties of time, such as temporal linearity, which might not exist in other genres of text, such as fiction.

All temporal expressions could be translated into constraints that treat time as points on a numerical axis and apply standard constraint solving techniques for equations and inequalities. However, this would result in the same loss of information that we see in the normalization to TimeML provided by HeidelTime. To bypass this problem, we find it more fitting to use constraints that reflect the linguistic expressions retaining their uncertainty. For example, to represent the meaning of *"in de late jaren zeventig"* (in the late seventies) it seems inappropriate to select an interval with hard, exact boundaries, for instance, from January 1, 1876, until New Year's eve 1879. A more intuitive solution is to replace the exact start date with a fuzzy representation. Finding the best way to do this is an issue for future research. Here we show some rules that are used in ongoing experiments.

6.1 Explicit dates and intervals

The tagger has already done an effort to recognize exact dates. The following rule converts an exact date recognized in the text into a single, more manageable constraint. Notice that the tagger via the *NamedEntity* attribute also indicates a relative position of each token in the assumed time expression, so that `'B-TIME'` indicates the beginning of a time expression, and `'I-TIME'` any immediately following tokens that are part of the same expression.

```
token(Bio,P,S,N0,N1,'B-TIME',_,date(C,De,Y,M,Da),_,'TW',Da),
token(Bio,P,S,N1,N2,'I-TIME',_,date(C,De,Y,M,Da),_,'SPEC',MonthName),
token(Bio,P,S,N2,N3,'I-TIME',_,date(C,De,Y,M,Da),_,'TW',_)
   ==> isPossibleDate(C,De,Y,M,Da), isMonthName(MonthName,M)
   | exactDate(Bio,P,S,N0,N3,date(C,De,Y,M,Da)).
```

It matches `token` constraints originating from a text such as *"12 februari 1889"*. In this rule we also perform a check that ensures the output from the tagger is valid, although this is fairly trivial for exact dates. The `isPossibleDate` and `isMonthName` predicates in the guard are part of a small lexical toolbox implemented for this application. The `'TW'` indicates a numeral.

An explicit year which is not part of a date is extracted as follows.

```
token(Bio,P,S,N1,N2,'B-TIME',_,Y,'TW',Year)
  ==> isPossibleExactYear(Year,C,De,Y)
   | exactYear(Bio,P,S,N1,N2,date(C,De,Y,null,null)).
```

The `isPossibleExactYear` predicate checks that the observed number, for example 1889, is a reasonable year and splits it into century, decade and year, for the example 18, 8, 9. The `null` values for month and date indicate that it is not relevant (at this level of analysis) to insert further information, as it is not recognized whether this mention of a year is intended as some exact, but not specified, date in that year, or a time interval spanning most of that year.

We have similar rules for identifying exact months in a given year and months without an explicitly given year. The latter are expected to be resolved using their context in a later phase of the analysis.

Exact time intervals are indicated in different ways in Dutch, one of the most common is to use the word *tot* (until) between two exact dates. This gives rise to the following CHR rule. In a simpagation rule like the following, the order of the constituents in the rule do not match the textual order, which is controlled by the `N`x attributes.

```
token(Bio,P,S,N1,N2,'O',_,null,tot,'VZ',tot)
\
exactDate(Bio,P,S,N0,N1,date(C1,D1,Y1,M1,D1)),
exactDate(Bio,P,S,N2,N3,date(C2,D2,Y2,M2,D2))
   <=>
     timeInterval(Bio,P,S,N0,N3, date(C1,D1,Y1,M1,D1),
                                 date(C2,D2,Y2,M2,D2)).
```

Notice that the rule removes the exact dates from the constraint store as they are no longer relevant after this pattern has been applied.

6.2 Indirect time points

We often encounter time expressions where part of it is not expressed directly. There may be a pronoun as in *"september van dat jaar"*, or simply *"september"*. The following rule attempts to identify a relevant year for such cases.

```
token(Bio,P,S,N1,N2,'O',_,null,dat,'VG',dat),
token(Bio,P,S,N2,N3,'O',_,null,jaar,'N',jaar)
\
month(Bio,P,S,N0,N1,date(null,null,null,M,null))
   <=> nearestBeforeYear(Bio,P,S,N0, NBY),
       isPossibleExactYear(NBY,C,De,Y)
    |
       month(Bio,P,S,N0,N3,date(C,De,Y,M,null)).
```

The `nearestBeforeYear` device is a special constraint that searches backwards through the text to match the first constraint that contains a reasonable year.

This constraint is then supplied to the new `month` constraint that replaces the one matched in the head.

When the nearest, previous mention of a year is part of a reference to literature, the system might make a wrong guess. To correct this, a preprocessing phase neutralizes such citation years before we apply the normalization rules. This phase is implemented using CHR rules that match the conventions for references used in this corpus.

6.3 Time intervals with partly implicit information

Consider the fragment *"... mei tot september 1904"* (May until September 1904), for which it seems reasonable to assume that *"mei"* refers to *"mei 1904"*. For this particular pattern, we implement the following rule.

```
token(Bio,P,S,N1,N2,'O',_,null,tot,'VZ',tot)
\
month(Bio,P,S,N0,N1,date(null,null,null,M1,null)),
month(Bio,P,S,N2,N3,date(C2,De2,Y2,M2,null))
   <=>
     timeInterval(Bio,P,S,N0,N3, date(C2,De2,Y2,M1,null),
                                 date(C2,De2,Y2,M2,null)).
```

However, this rule will be incorrect if applied to *"december tot september 1904"*, where (at least in the lack of other strong indicators) *"december"* most likely refers to *"december 1903"*. There are yet other cases where the year is given for the first and not the last given month and so on. We could write a set of almost identical CHR rules for the different cases, but instead it is more convenient to write one general rule as follows (thus also subsuming the rule above).

```
token(Bio,P,S,N1,N2,'O',_,null,tot,'VZ',tot)
\
month(Bio,P,S,N0,N1,date(C1,De1,Y1,M1,null)),
month(Bio,P,S,N2,N3,date(C2,De2,Y2,M2,null))
   <=> canResolveMonthInterval(C1,De1,Y1,M1, C2,De2,Y2,M2,
                               C1r,De1r,Y1r,M1r, C2r,De2r,Y2r,M2r)
    |  timeInterval(Bio,P,S,N0,N3, date(C1r,De1r,Y1r,M1r),
                                    date(C2r,De2r,Y2r,M2r)).
```

The predicate `canResolveMonthInterval` in the guard is a Prolog predicate that covers all the different cases of known and unknown values. For the example *"december 1903 tot mei"*, the variables `C2r`, `De2r`, `Y2r` would be instantiated to indicate the year 1904. The guard fails (which prevents the rule from being applied) when there is no sensible way of filling in the missing information.

6.4 Open intervals

Biographical text often indicates time intervals by giving only a start or an end time, where the missing point is either obvious according to the timeline or

not considered important. Following the recognition of closed intervals, we can define rules that resolve expressions such as *"tot september 1904"*, when it is not immediately preceded by another time expression.

```
token(Bio,P,S,N1,N2,'O',_,null,tot,'VZ',tot)
\
exactDate(Bio,P,S,N2,N3,date(C2,De2,Y2,M2,Da2))
   <=>
       timeInterval(Bio,P,S,N1,N3, DATE, date(C2,De2,Y2,M2,Da2)),
       DATE <=< date(Y2,M2,D2).
```

The symbol `<=<` is a constraint that represents time order. Current experiments investigate how to optimally combine such basic algebraic constraints with the principle of searching in the surrounding text for a reasonable start or end time.

7 Expected results and considerations about test

The method presented so far allows us to introduce new rules optimized for the chosen corpus, where, in some cases, the rules are so specific that they are only applied once. These tailored rules allow us to obtain maximal recall and precision measures on the given corpus.

However, this conclusion is not very interesting in itself, and we still need to design tests that provide a fair measurement for comparison with related methods. This could be based on a traditional splitting between training and validation, and allowing only rules of a certain generality, measured by the number of times each rule is applied to the training set. The rule set exemplified above is still under development and not yet at a stage that justifies a proper evaluation.

However, some initial tests have shown incorrect results caused by the CHR rules, for instance, when they fail to recognize a time expression reported by the temporal tagger. A possible cause of this problem could be the fact that the tagger was developed independently before the use of CHR was considered. To get the best out of the combination of a smart tagger and CHR matching, there needs to be a clearer separation of the tasks performed at each level and, thus, a more consistent interface between the two levels. The experiment reported here required 132 different rules, and we judge that about 100 could do if tagger and rule set were designed together. The rule set is highly modular, as each rule can be designed and tested individually, and it is easy to add specialized rules for named events and idiomatic expressions.

8 Conclusions

An approach to extract and resolve time expressions from natural language text is proposed. Time in natural language is often expressed in indirect and/or underspecified ways that reflect lack of exact information. We suggest a rule- and

constraint-based approach that can be tailored carefully to the pragmatics of a particular class of texts, here exemplified by interrelated personal biographies written in Dutch concerning a given historical period and context. We start from text that has been processed by a fairly advanced, but not flawless, temporal tagger. Our rules can both rely on and correct for consistent mistakes in the tagging.

We expect that the principle described here can be developed into a general method for resolution of time expressions, that can be applied to many specific cases in a modular way, and provides a flexibility for long distance references in any direction.

The explicit manipulation of `null` values for the resolution of implicit information, as described above, is only an intermediate solution. It seems more appropriate to represent unspecified time values as constrained logical variables and develop a full constraint solver to handle the relevant collection of temporal representations, thus resolving unknown values in an automatic way. This will reduce the need for explicit search for missing information, as the inference from context is done via relationships between time expressions, thus providing a symmetrical flow of information forwards, or backwards, through the text. In case of inconsistent or over-specified time constraints embedded in a text, the application of soft constraint techniques will be appropriate, as studied in CHR by [19]. Automatic extraction of rules from trusted, time tagged corpora may be another interesting continuation of this work; the CHRiSM system [20] that combines CHR with probabilistic machine learning techniques seems suited for such experiments.

References

1. Costa, F., Branco, A.: Extracting temporal information from portuguese texts. In de Medeiros Caseli, H., Villavicencio, A., Teixeira, A.J.S., Perdigão, F., eds.: PROPOR. Volume 7243 of Lecture Notes in Computer Science., Springer (2012) 99–105
2. Hovy, D., Fan, J., Gliozzo, A.M., Patwardhan, S., Welty, C.A.: When did that happen? - Linking events and relations to timestamps. In Daelemans, W., Lapata, M., Màrquez, L., eds.: EACL, The Association for Computer Linguistics (2012) 185–193
3. Verhagen, M., Gaizauskas, R.J., Schilder, F., Hepple, M., Moszkowicz, J., Pustejovsky, J.: The tempeval challenge: identifying temporal relations in text. Language Resources and Evaluation **43**(2) (2009) 161–179
4. Dahl, V., Blache, P.: Implantation de grammaires de propriétés en CHR. In Mesnard, F., ed.: Actes des 13èmes Journées Francophones de Programmation en Logique avec Contraintes, Hermes (2004)
5. Christiansen, H., Have, C.T., Tveitane, K.: From use cases to UML class diagrams using logic grammars and constraints. In: RANLP '07: Proc. Intl. Conf. Recent Adv. Nat. Lang. Processing. (2007) 128–132
6. Christiansen, H., Have, C.T., Tveitane, K.: 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. (2007) 40–52

7. Bavarian, M., Dahl, V.: Constraint based methods for biological sequence analysis. The Journal of Universal Computer Science **12**(11) (2006) 1500–1520
8. Dahl, V., Gu, B.: A CHRG analysis of ambiguity in biological texts. In: CSLP '07: Proc. 4th Intl. Workshop on Constraints and Language Processing. Volume 113 of Roskilde University Computer Science Research Report. (2007) 53–64
9. Christiansen, H., Li, B.: Approaching the chinese word segmentation problem with CHR grammars. In: Proceedings of CSLP 2011, the 6th International Workshop on Constraints and Language Processing. A workshop at CONTEXT '11: The 7th International and Interdisciplinary Conference on Modeling and Using Context 2011. Volume 134 of Roskilde University Computer Science Research Report. (2011) 21–31
10. Christiansen, H.: CHR Grammars. Theory and Practice of Logic Programming **5**(4-5) (2005) 467–501
11. Frühwirth, T.: Constraint Handling Rules. Cambridge University Press (2009)
12. Penn, G., Richter, F.: The other syntax: Approaching natural language semantics through logical form composition. In Christiansen, H., Skadhauge, P.R., Villadsen, J., eds.: Constraint Solving and Language Processing. First International Workshop, CSLP 2004, Roskilde, Denmark, September 1-3, 2004, Revised Selected and Invited Papers. Volume 3438 of Lecture Notes in Computer Science., Springer (2005) 48–73
13. Christiansen, H., Dahl, V.: Meaning in Context. In Dey, A., Kokinov, B., Leake, D., Turner, R., eds.: Proceedings of Fifth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-05). Volume 3554 of Lecture Notes in Artificial Intelligence. (2005) 97–111
14. Frühwirth, T.W.: Theory and practice of Constraint Handling Rules. Journal of Logic Programming **37**(1-3) (1998) 95–138
15. van den Bosch, A., Busser, B., Daelemans, W., Canisius, S.: An efficient memory-based morphosyntactic tagger and parser for dutch. In van Eynde, F., Dirix, P., Schuurman, I., Vandeghinste, V., eds.: Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting (CLIN17), Leuven, Belgium (2007) 99–114
16. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005). (2005) 363–370
17. Strötgen, J., Gertz, M.: Multilingual and cross-domain temporal tagging. Language Resources and Evaluation (2012)
18. Strötgen, J., Gertz, M.: Heideltime: High quality rule-based extraction and normalization of temporal expressions. In: Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval-2010). (2010) 321–324
19. Bistarelli, S., Frühwirth, T.W., Marte, M., Rossi, F.: Soft constraint propagation and solving in constraint handling rules. Computational Intelligence **20**(2) (2004) 287–307
20. Sneyers, J., Meert, W., Vennekens, J., Kameya, Y., Sato, T.: Chr(prism)-based probabilistic logic learning. Theory and Practice of Logic Programming **10**(4-6) (2010) 433–447