A constraint-based bottom-up counterpart to DCG

Henning Christiansen
Roskilde University, Computer Science Dept.
P.O.Box 260, DK-4000 Roskilde, Denmark
henning@ruc.dk

Abstract

A new grammar formalism, CHR Grammars (CHRG), is proposed that provides a constraint-solving approach to language analysis, built on top of the programming language of Constraint Handling Rules in the same way as Definite Clause Grammars (DCG) on Prolog. CHRG works bottom-up and adds the following features when compared with DCG: – An inherent treatment of ambiguity without backtracking. – Robust parsing; do not give up in case of errors but return the recognized phrases. – A flexibility to produce and consume arbitrary hypotheses making it straightforward to deal with abduction, integrity constraints, operators *à la* assumption grammars, and to incorporate other constraint solvers. – Context-sensitive rules that apply for disambiguation, coordination in natural language, and tagger-like rules.

1 Introduction

Definite Clause Grammars (Colmerauer 75; Pereira & Warren 80) (DCG) have been appreciated for their declarative nature and execution environment inherited from the logic programming language Prolog. Where Prolog and DCG work top-down, the language of Constraint Handling Rules (Frühwirth 98) (CHR) provides a logic programming framework for bottom-up computations that implies several advantages for language processing (Abdennadher & Schütz 98; Abdennadher & Christiansen 00). In fact, any context-free grammar or DCG can be rewritten in a straightforward way as a set of propagation rules of CHR that serves as an error robust parser with an inherent treatment of ambiguity without backtracking, and no artificial nonterminals are necessary as often in a DCG. Restrictions are that empty productions and loops among nonterminals cannot be handled.

A new and powerful grammar formalism, called CHRG for CHR Grammars, is proposed with a form of context-sensitive rules that can take into account arbitrary grammar symbols to the left and right of a sequence supposed to match a given nonterminal. This allows tagger-like grammar rules, it can be used for disambiguating simple and ambiguous context-free grammar rules, and provides also a way to handle coordination in natural language as shown by an example: The following rules are excerpt of a CHR grammar for sentences such as *"Peter likes and Mary detests spinach"*.

```
sub(A), verb(V), obj(B) ::> sent(s(A,V,B)).
subj(A), verb(V) /- [and], sent(s(_,_,B))
    ::> sent(s(A,V,B)).
```

The first rule is to be understood in the usual way that a complete `sub-verb-obj` sequence can be reduced to a `sent` node. The second rule is an example of a context-sensitive rule: It applies to a `subj-verb` sequence only when followed by terminal symbol "`and`" and another `sent` node, and in this case the incomplete sentence takes its subject, matched by variable `B`, from this following `sent` node. The marker "`/-`" separates the `subj-verb` sequence from the required right context; a similar marker may indicate left context. In contrast to most other notations, CHRG mentions the constituents before the whole to emphasize the bottom-up nature.

The CHRG notation includes the full expressive power of CHR, including the ability to integrate with arbitrary constraint solvers and a highly flexible way to handle sets of extra-grammatical hypotheses. For example, abduction for context comprehension can be characterized in CHRG in a surprisingly simple way that requires no meta-level overhead as do other approaches to abduction in language processing. Elements of linear logic as in Assumption Grammars (Dahl *et al.* 97) are included in a similar way.

2 Background and related work

The notion of constraints, with slightly different meanings, is often associated with language processing. "Constraint grammars" and "unification grammars" are often used for feature-structure grammars, and constraint programming techniques have been applied for the complex constraints that arise in natural language processing; see, e.g., (Allen 95; Duchier 00) for introduction and overview. One approach using CHR for this purpose in HPSG is (Penn 00). See also (Blache 00; Duchier & Thater 99; Maruyama 94; Schröder *et al.* 00) for similar approaches.

CHR has been applied for diagram parsing by Meyer (Meyer 00) but not elaborated into a grammar formalism; Morawietz (Morawietz 00) has implemented deductive parsing (Shieber *et al.* 95) in CHR and shown that a specialization of a general bottom-up parser leads to rules similar to those produced by our translator; none of these consider context in the sense we do. Abduction in CHR has been applied by (Christiansen & Dahl 02) for diagnosis and correction of grammatical errors. An attempt to characterize the grammar of ancient Egyptian hieroglyph inscriptions by means of context-sensitive rules in CHRG is given by (Hecksher *et al.* 02). CHR is available as extension of, among others, SICStus Prolog (Swedish Institute of Computer Science 03) which is the version applied in the present work. For more details, refer to a forthcoming journal paper (Christiansen 04) and the website (Christiansen 02b) for CHRG with source code, user's guide and example grammars.

Finally, we mention important advances based on soft or graduated constraints and statistically based parsing, representing important advances for robustness and disambiguation, see, e.g., (Collins 96; Charniak 97; Heinecke *et al.* 98; Abney *et al.* 99). We have not tried to integrated such ideas in CHRG but (Bistarelli *et al.* 02) have shown how soft constraints can be handled in CHR so a combination seems possible.

3 Syntax, semantics, and implementation of CHRG

A *CHR Grammar*, or *CHRG* for short consists of finite sets of *grammar* and *constraints symbols* and a finite set of *grammar rules.*

An *attributed grammar symbol*, for short called a *grammar symbol*, is formed as a logical atom whose predicate symbol is a grammar symbol; a grammar symbol formed by `token`/1 is called a *terminal*, any other grammar symbol a *nonterminal.* Sequences of terminal symbols `token(`a_1`)`, ..., `token(`a_n`)` may also be written `[`a_1`, ...,` a_n`]`; if ground, such a sequence is called a *string.*

A *propagation (grammar) rule* is of the form

$$\alpha \ \texttt{-\textbackslash}\ \beta \ \texttt{/-}\ \gamma \ \ \texttt{::>}\ \ G \ \texttt{|}\ \delta.$$

The part of the rule preceding the arrow `::>` is called the *head*, G the *guard*, and δ the body; $\alpha, \beta, \gamma, \delta$ are sequences of grammar symbols and constraints so that β contains at least one grammar symbol, and δ contains exactly one grammar symbol which is a nonterminal (and perhaps constraints); α (γ) is called *left (right) context* and β the *core* of the head; G is a guard as in CHR that may test properties for the variables in the head of the rule. If left or right context is empty, the corresponding marker is left out and if G is empty (interpreted as `true`), the vertical bar is left out. The convention from DCG is adopted that non-grammatical constraints in head and body of a rule are enclosed in curly brackets.

The implemented system combines CHRG with rules of CHR and Prolog which is convenient for defining behaviour of non-grammatical constraints. CHRG includes also notation for gaps and parallel match not described here.

In Chomskian grammars, derivations are defined over strings of symbols and this suffices also for a large class of CHRGs. Several aspects of CHRG make this too restrictive: Context-sensitive rules of CHRG extend those of Chomsky by the possibility to refer to any grammar symbol which has been created at some stage during derivation (not only the "current" stage); extra-grammatical hypotheses created during a derivation serve as a common resource for all subsequent derivation steps; CHRG includes other sorts of rules (below) inherited from CHR which need to be specified in a bottom-up fashion.

The most obvious way to define derivations in CHRG seems to be to represent sequencing by means of word boundaries (e.g., integer numbers) and each stage in the derivation as a constraint store. For each grammar symbol N of arity n, we assume a corresponding constraint also denoted by N of arity $n + 2$ called an *indexed grammar symbol* with the two extra arguments referred to as phrase (or word) *boundaries.*

For a grammar symbol $S = N(\bar{a})$, the notation S^{n_0,n_1} refers to the indexed grammar symbol $N(n_0,n_1,\bar{a})$ with integers $n_0 < n_1$; in case of a terminal, $n_0 + 1 = n_1$ is assumed. For any sequence σ of grammar symbols $S_1, \ldots, S_k$ and increasing integers $n_0, n_1, \ldots, n_k$, we let σ^{n_0,n_k} refer to the set $\{S_1^{n_0,n_1},\ldots,S_k^{n_{k-1},n_k}\}$ with the existence of $n_1,\ldots,n_{k-1}$ understood. This extends so that for a sequence of grammar symbols and extra-grammatical constraints, we remove all constraints from the sequence, put indexes on the remaining grammar symbols, and add again the constraints in their original position.

A *constraint store* is a set of constraints and indexed grammar symbols and the *initial store* for a terminal string σ is the store $\sigma^{0,k}$ where k is the length of σ. An *instance* (and *ground instance*) of a grammar rule is defined in the usual way. A *derivation step* from one constraint store $\mathbf{S}_1$ to another $\mathbf{S}_2$ by an instance of a propagation grammar rule α `-\` β `/-` γ `::>` G `|` δ is defined whenever

- $\vdash \exists\bar{x}G$ where $\bar{x}$ are the variables in G not in α,β,γ,
- $\alpha^{i,j}\cup\beta^{j,k}\cup\gamma^{k,\ell}\subseteq\mathbf{S}_1$, and $\mathbf{S}_2=\mathbf{S}_1\cup\delta^{j,k}$.

Useful for optimization purposes, CHRG includes two other sorts of rules that reflect the underlying CHR system. A *simplification (grammar) rule* is similar to a propagation rule except that the arrow is replaced by `<:>`; a *simpagation (grammar) rule* is similar to a simplification except that one or more grammar symbols or constraints in the core of the head are prefixed by an exclamation mark "`!`". Derivation with these rules is defined as above, except that the new state is given $\mathbf{S}_2=\mathbf{S}_1\cup\delta^{j,k}\setminus\beta^{j,k}\cup\beta'^{j,k}$ where $\beta'^{j,k}$ are those elements of $\beta^{j,k}$ prefixed by exclamation marks.

A *parsing derivation* for terminal string σ (and given grammar) is defined as a sequence of steps starting with state $\sigma^{0,n}$. A *final* constraint store is one in which no further step can apply; for any grammar symbol N with $N^{0,n}$ in the final store, we say that N is *accepted* from σ.

CHRG is implemented by a compiler that translates a grammar into a CHR program which, when executed, realizes the semantics defined above. Basically, it extends grammar symbols with arguments for word boundaries.

For propagation rules, the final state is independent of the order in which rules apply but in general, this order does matter. The implemented system inherits a specific order from CHR defined roughly as follows: Enter the `tokens` into the store in sequence and apply as many rules as possible before entering the next one; details are spelled out in (Christiansen 04).

With respect to efficiency, we can rely on the indexing techniques applied in CHR and expected future optimizations of the CHR system involving control and data flow analyses. Referring to results of (McAllester 00; Ganzinger & McAllester 01; Ganzinger & McAllester 02), complexity results have been shown that also are verified empirically: A locally unambiguous grammar executes linearly in terms of the length of the input string. For arbitrary grammar without attributes (including any context-free grammar without single productions and loops), complexity is cubic similarly to classical algorithms such as Early and Cocke-Younger-Kasami.

4 Examples

The following shows the syntax used in the implemented system. The "`handler`" command is a reminiscent of the CHR system; grammar symbols are declared by the `grammar_symbols` construct as shown. The final command has no effect in the present example, but it adds extra rules needed for the extensions described below.

```
handler my_grammar.
grammar_symbols np/0, verb/0, sentence/0.
np, verb, np ::> sentence.
[peter] ::> np.
[mary] ::> np.
[likes] ::> verb.
end_of_CHRG_source.
```

For the string *"Peter likes Mary"*, the `np`, `verb`, and `np` are recognized, and then the `sentence` rule applies. This grammar consists of propagation rules, so the `tokens`, `nps`, and `verb` are not consumed. If a rule were added, say `np, [likes] ::> sentence1`, a `sentence` as well as a `sentence1` would be recognized. If all rules were changed into simplification rules (changing `::>` to `<:>`), only one would be recognized. For an ambiguous grammar of propagation rules, a sentence node is generated for each different reading.

Context-sensitive rules were shown in the introduction for coordination handling. In the following, left and right contexts are applied in a tagger-like fashion to classify `nouns` as `subj` or `obj` according to their position relative to the verb.

```
noun(A) /- verb(_)       ::> subj(A).
verb(_) -\ noun(A)       ::> obj(A).
noun(A), [and], subj(B)  ::> subj(A+B).
obj(A), [and], noun(B)   ::> obj(A+B).
```

Context parts perhaps combined with simplification rules can be used for disambiguation of straightforward and otherwise ambiguous grammars. The following shows a grammar for arithmetic expressions with traditional operator precedence; the semicolon denotes alternatives in syntactic context and the `where` notation stands for syntactic replacement (enhances readability only); notice how standard Prolog tests are applied in guards of the two last rules.

```
e(E1),[+],e(E2) /- Rc <:> e(plus(E1,E2))
   where Rc = (['+'];[')'];[eof]).
e(E1),[*],e(E2) /- Rc <:> e(times(E1,E2))
   where Rc = ([*];[+];[')'];[eof]).
e(E1),[^],e(E2) /- [X]
   <:> X \= ^ | e(exp(E1,E2)).
['('], e(E), [')'] <:> e(E).
[N] <:> integer(N) | e(N).
```

This grammar uses standard follow items but hopefully, it illustrates the flexibility that CHRG provides for a competent grammar writer to engineer compact, yet precise and widely covering languages specifications without too many artificial grammar symbols. We do not give examples using the full flexibility of extra constraints in head and body of rules, but the principle is indicated in the following sections that provide more "structured" ways of using such constraints.

5 Abduction in CHRG

Abduction for language interpretation, as conceptual model and implementation, has been recognized by several authors, e.g., (Charniak & McDermott 85; Hobbs *et al.* 93; Christiansen 93; Gabbay *et al.* 97) just to mention a small fraction. Usually abduction requires a heavy meta-level overhead which may be one of the reasons why abduction seldom is used in practice.

A simple technical trick, initially described by (Christiansen 02a), can be applied to a grammar so it can run as a CHRG without any extra abduction machinery. A brief introduction is given here; full details in (Christiansen 04).

Consider the following grammar rule in which F refers to a fact about the semantical context for a given discourse; it can be read as a CHRG rule or an equivalent DCG rule.

$$\texttt{a, b, } \{F\} \texttt{ ::> ab} \qquad (1)$$

If two subphrases referred to by `a` and `b` have been recognized and the context condition F holds, it is concluded that an `ab` phrase is feasible, grammatically as well as with respect to the context. Analysis with such rules works well when context is known in advance for checking that a given discourse is syntactically and semantically sound with respect to that context.

In case the context is unknown, we have a more difficult abductive problem of finding proper context theory so that an analysis of an observed discourse is possible. Rules of the form (1) are not of much use unless an interpreter that includes abduction is involved.

Our solution is to move the reference to the contextual predicates into the other side of the implication, thus replacing the rule above with the following:

$$\texttt{a, b ::> } \{F\}\texttt{, ab} \qquad (2)$$

Intuitively it reads: If suitable `a` and `b` phrases are found, it is feasible to assert F and, thus, under this assumption conclude `ab`.

Obviously, the two formulations (1) and (2) are not logically equivalent but it is straightforward to formulate and prove correctness. Formulation (2) can be executed as a CHRG with the abductive explanation of a discourse being read out of the final constraint store.

For simplicity of explanation in the following, assume the underlying grammar to be unambiguous; the full CHRG system includes techniques not described here to avoid different hypothesis sets for different parses to be mixed up.

A specification based on abduction needs *integrity constraints* (ICs) to suppress senseless explanations and CHR rules are effective for this purpose as indicated by the following example.

Consider the discourse *"Garfield eats Mickey, Tom eats Jerry, Jerry is mouse, Tom is cat, Mickey is mouse."* We intend to learn from it a categorization of the individuals and which categories that are food items for others. An interesting question is to which category Garfield belongs as this is not mentioned explicitly. The following vocabulary is defined; the `abducibles` declaration is synonymous with CHR's `constraints` declaration except that it also introduces predicates for negated abducibles with ICs that implement explicit negation.

```
abducibles food_for/2, categ_of/2.
grammar_symbols name/1, verb/1,
  sentence/1, category/1.
```

The following two CHR rules serve as ICs.

```
categ_of(N,C1), categ_of(N,C2) ==> C1=C2.
food_for(C1,C), food_for(C2,C) ==> C1=C2.
```

I.e., the category for a name is unique, and for the sake of this example it is assumed that a given category is the food item for at most one other category. The following part of the grammar classifies the different tokens.

```
[tom] ::> name(tom).    ...
[is]  ::> verb(is).     ...
verb(is) -\ [X] <:> category(X).
```

The last rule applies a syntactic left context part in order to classify any symbol to the right of an occurrence of "is" as a `category`.

A sentence *"Tom is cat"* is only faithful to a context if `categ_of(tom,cat)` holds in it. Thus, if sentence *"Tom is cat"* is taken as true, it is feasible to assume `categ_of(tom,cat)`; in general:

```
name(N), verb(is), category(C) ::>
  {categ_of(N,C)}, sentence(is(N,C)).
```

A sentence *"Tom eats Jerry"* is only faithful to a context in which proper `categ_of` and `food_for` facts hold:

```
name(N1), verb(eats), name(N2) ::>
  {categ_of(N1,C1), categ_of(N2,C2),
                     food_for(C1,C2)},
  sentence(eats(N1,N2)).
```

Let us trace the analysis of the sample discourse; only the context facts are recorded. First sentence *"Garfield eats Mickey"* gives rise to

```
categ_of(garfield,X1), categ_of(mickey,X2),
food_for(X1,X2).
```

The "`X`"s are uninstantiated variables. The next *"Tom eats Jerry"* gives

```
categ_of(tom,X3), categ_of(jerry,X4),
food_for(X3,X4).
```

"Jerry is mouse" gives `categ_of(jerry,mouse)`, and the first IC immediately unifies `X4` with `mouse`. In a similar way "Tom is cat" gives rise to a unification of `X3` with `cat` and `food_for(X3,X4)` has become

```
food_for(cat,mouse).
```

Finally *"Mickey is mouse"* produces `categ_of(mickey,mouse)` that triggers the first IC unifying `X2` with `mouse` and thus the second IC sets `X1=cat` and there is no other possibility. So as part of the solution to this language interpretation problem, we have found that Garfield is a cat.

6 Assumption grammars in CHRG

Assumption Grammars (Dahl *et al.* 97) (AGs) include facilities to communicate hypotheses between different subtrees which differ from abduction in the sense that hypotheses are explicitly produced and explicitly used, possible being consumed. A collection of operators is provided to control the scope of hypotheses which is not possible with an abductive approach. We explain here how the AG operators are included in CHRG. For simplicity, we describe the technique for unambiguous grammars; the full CHRG system is able to handle ambiguity in assumption grammars.

In an AG, the expression `+h(a)` means to assert a linear hypothesis which can be used once in the subsequent text by means of the expression `-h(a)` (or `-h(X)`, binding `X` to `a`) called an *expectation.* Asserting the hypothesis by `*h(a)` means that it can be used over and over again. We deviate slightly from the syntax of (Dahl *et al.* 97) as to achieve a more symmetric notation and introduce three operators for so-called time-less hypotheses, `=+`, `=-`, and `=*`, whose meanings are similar except that assumptions can be used and consumed in any order. The operators are defined as constraints in CHR and can be called from the body of grammar rules. The interaction between the operators is implemented by CHR rules, roughly of the following shape (shown for the time-less versions only).

```
=+A, =-B <=>  A=B.
=*A \ =-B <=> A=B.
```

By the first rule, a pair of assumption `=+h(a)` and expectation `=-h(X)` are removed from the constraint store producing the effect of binding `X` to `a`. If assumption `=*h(a)` were used instead, the second rule can apply to several instances of `=-h(···)`. The actual rules in the system are a bit more complicated so that an expectation `=-h(X)` can try out different possible assumptions.

We sketch an example adapted from (Dahl *et al.* 97). We consider sentences with pronouns and coordination such as "Martha likes and Mary likes Paul; she hates her". We add gender to names and pronouns, and whenever a name appears as subject or object (in this grammar grouped as `np`s), an assumption is made that the given name is `acting`. A pronoun as subject or object gives rise to an expectation for someone `acting` of appropriate gender. The principle is shown by the following excerpt.

```
[mary] <:> name(mary, fem).
[she] <:> pronoun(fem).
name(X,Gender)  <:>
  *acting(X,Gender), np(X,Gender).
pronoun(Gender) <:>
  -acting(X,Gender), np(X,Gender).
```

To handle the coordination problem, an incomplete sentence raises a time-less expectation for a subject which is met by the assumption produced by the full sentence at the end.

```
np(A,_), verb(V) /- [and] <:>
  =-ref_object(B), sentence(s(A,V,B)).
np(A,_), verb(V), np(B,_) <:>
  =*ref_object(B), sentence(s(A,V,B)).
```

One of the possible final states produced for the sample text above contains `sentence` symbols with the following attributes: `s(martha,like,paul)`, `s(mary,like,paul)`, and `s(mary,hate,martha)`. The CHRG version of AG goes beyond the original proposal by adding integrity constraints.

7 Conclusion

CHR Grammars founded of current constraint logic technology have been introduced, and their application to aspects of natural language syntax illustrated by small examples. CHRG is a technologically updated ancestor of Definite Clause Grammars: A relative transparent layer of syntactic sugar over a declarative programming language, providing both conceivable semantics and fairly efficient implementation. In CHRG we have just replaced Prolog by Constraint Handling Rules. The result of this shift is a very powerful formalism in which several linguistic aspects usually considered to be complicated or difficult are included more or less for free:

- Ambiguity and grammatical errors are handled in a straightforward way, all different (partial) parses are evaluated in parallel.

- Context-sensitive rules, which are inherent part of the paradigm, handle coordination in an immediate way.

- Abduction, which is useful for identifying indirectly implied information, is expressed directly with no additional computational devices needed.

Context-sensitive rules combined with ability to handle left-recursion (as opposed to DCG) are a great help for producing grammars with relatively few, concise rules without artificial nonterminals; a drawback is the lack of empty production.

There is a large unexplored potentiality in CHRG and language processing by means of CHR. We can mention the possibility of integrating arbitrary constraint solvers, and adding weights to prioritize between different parses (and abductive explanations!) seems quite straightforward. As already mentioned, (Bistarelli *et al.* 02) have shown how to handle soft constraints in CHR and this opens up for integrating recent results in statistically based parsing.

In another paper (Christiansen & Dahl 02) we have extended with facilities for error detection and correction. Robustness combined with flexibility (e.g., error correction) makes application in speech systems interesting: If, e.g., the phonetic component cannot distinguish a token from being `hats` or `cats`, we simply add both to the input state with identical boundaries. Parsing from a state {`token(0,1,hats)`, `token(0,1,cats)`, `token(1,2,eat)`, `token(2,3,mice)`} will explore the different options in parallel, with only those satisfying syntactic and semantic requirements of the actual grammar leading to a full parse tree.

No real-world applications have been developed in CHRG yet, but we have good expectation for scalability as selected grammars can run in linear time. Furthermore, the full flexibility of the underlying CHR and Prolog machinery is available for optimizations. Independently, CHRG is available as powerful modeling and prototyping tool.

In a way, the idea is naïve, almost too naïve, just applying grammar rules bottom-up over and over until the process stops. However, we can rely now on the underlying, well-established computational paradigm of CHR for such rules-based computations. Recent extensions to the CHRG system concern notation for optional elements and improvement of indexing techniques for evaluation of different abducibles/assumptions sets in parallel, and new grammars are under development in the system.

It is our hope that the availability of the CHRG system can stimulate research in constraint-based language analysis, ideally leading to a full integration of lexical, grammatical, semantical, and pragmatic processing.

Acknowledgements

Part of this work has been carried out while the author visited Simon Fraser University, Canada, partly supported by the Danish Natural Science Council; thanks to Verónica Dahl for helpful discussion and providing a stimulating environment. This research is supported in part by the OntoQuery project funded by the Danish Research Councils, and the IT-University of Copenhagen.

References

(Abdennadher & Christiansen 00) Slim Abdennadher and Henning Christiansen. An experimental clp platform for integrity constraints and abduction. In *Proceedings of FQAS2000, Flexible Query Answering Systems: Advances in Soft Computing series*, pages 141–152. Physica-Verlag (Springer), 2000.

(Abdennadher & Schütz 98) Slim Abdennadher and Heribert Schütz. CHR$^{\vee}$: A flexible query language. In *Proc. Int. Conference on Flexible Query Answering Systems FQAS, Roskilde, Denmark*, volume 1495 of *LNCS*, pages 1–15. Springer-Verlag, 1998.

(Abney *et al.* 99) Steven P. Abney, David A. McAllester, and Fernando Pereira. Relating probabilistic grammars and automata. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 542–549, San Francisco, California, 1999.

(Allen 95) James Allen. *Natural Language Understanding — 2nd Edition*. The Benjamin/Cummings Publishing Company, 1995.

(Bistarelli *et al.* 02) Stefano Bistarelli, Thom Frhwirth, and Michael Marte. Soft constraint propagation and solving in chrs. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 1–5. ACM Press, 2002.

(Blache 00) Philippe Blache. Constraints, linguistic theories and natural language processing. In *Lecture Notes in Computer Science 1835*, pages 221–232. Springer, 2000.

(Charniak & McDermott 85) E. Charniak and D. McDermott. *Introduction to Artificial Intelligence*. Addison-Wesley Publishing Company, 1985.

(Charniak 97) Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. In *AAAI/IAAI*, pages 598–603, 1997.

(Christiansen & Dahl 02) Henning Christiansen and Veronica Dahl. Logic grammars for diagnosis and repair. In *ICTAI'02, proc. of 14th IEEE International Conference on Tools with Artificial Intelligence, November 4-6, 2002 Washington D.C*, pages 307–314. IEEE, 2002.

(Christiansen 93) Henning Christiansen. Why should grammars not adapt themselves to context and discourse? In *4th International Pragmatics Conference, Kobe, Japan, July 23–30 1993, (Abstract collection)*, page 23. International Pragmatics Association, 1993. Extended version: http://www.dat.ruc.dk/~henning/IPRA93.ps.

(Christiansen 02a) Henning Christiansen. Abductive language interpretation as bottom-up deduction. In Shuly Wintner, editor, *Natural Language Understanding and Logic Programming*, volume 92 of *Datalogiske Skrifter*, pages 33–47, Roskilde, Denmark, July 28 2002.

(Christiansen 02b) Henning Christiansen. CHR Grammar web site, Released 2002. http://www.ruc.dk/~henning/chrg, 2002.

(Christiansen 04) Henning Christiansen. CHR Grammars. To appear in *Theory and Practice of Logic Programming*, 2004.

(Collins 96) Michael John Collins. A new statistical parser based on bigram lexical dependencies. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 184–191, San Francisco, 1996. Morgan Kaufmann Publishers.

(Colmerauer 75) Alain Colmerauer. Les grammaires de metamorphose. Technical report, Groupe d'Intelligence Artificielle, Université de Marseille-Luminy, November 1975. Translated into English as (Colmerauer 78).

(Colmerauer 78) Alain Colmerauer. Metamorphosis grammars. In Leonard Bolc, editor, *Natural Language Communication with Computers*, volume 63 of *Lecture Notes in Computer Science*, pages 133–189. Springer-Verlag, Berlin, 1978. English translation of (Colmerauer 75).

(Dahl *et al.* 97) Veronica Dahl, Paul Tarau, and Renwei Li. Assumption grammars for processing natural language. In Lee Naish, editor, *Proceedings of the 14th International Conference on Logic Programming*, pages 256–270, Cambridge, 1997. MIT Press.

(Duchier & Thater 99) Denys Duchier and Stefan Thater. Parsing with tree descriptions: A constraint-based approach. In *6th International Workshop on Natural Language Understanding and Logic Programming (NLULP '99), December 3-4*, pages 17–32, Las Cruces, New Mexico, USA, 1999.

(Duchier 00) Denys Duchier. Constraint Programming For Natural Language Processing. Lecture Notes, ESSLLI 2000, 2000.

(Frühwirth 98) Thom Frühwirth. Theory and practice of constraint handling rules, special issue on constraint logic programming. *Journal of Logic Programming*, 37(1–3):95–138, October 1998.

(Gabbay *et al.* 97) Dov Gabbay, Ruth Kempson, and J. Pitt. Labeled abduction and relevance reasoning. In Robert Demolombe and T. Imielinski, editors, *Nonstandard Queries and Nonstandard Answers*, pages 155–185. Oxford Science Publications, , 1997.

(Ganzinger & McAllester 01) Harald Ganzinger and David McAllester. A new meta-complexity theorem for bottom-up logic programs. *Lecture Notes in Computer Science*, 2083:514–528, 2001.

(Ganzinger & McAllester 02) Harald Ganzinger and David McAllester. Logical algorithms. In Peter J. Stuckey, editor, *Logic Programming*, volume 2401 of *Lecture Notes in Computer Science*, pages 209–223. Springer-Verlag, July 29–August 1 2002.

(Hecksher *et al.* 02) Thomas Hecksher, Sune T. B. Nielsen, and Alexis Pigeon. A CHRG model of the ancient Egyptian grammar. Unpublished student project report, Roskilde University, Denmark, 2002.

(Heinecke *et al.* 98) Johannes Heinecke, Jurgen Kunze, Wolfgang Menzel, and Ingo Schroder. Eliminative parsing with graded constraints. In *COLING-ACL*, pages 526–530, 1998.

(Hobbs *et al.* 93) Jerry R. Hobbs, Mark E. Stickel, Douglas E. Appelt, and P. Martin. Interpretation as abduction. *Artificial Intelligence*, 63(1–2):69–142, October 1993.

(Maruyama 94) H. Maruyama. Structural disambiguation with constraint propagation. In *Proceedings of the 28th Annual Meeting of the ACL*, pages 31–38, Pittsburgh, 1994.

(McAllester 00) David A. McAllester. Meta-complexity theorems: Talk abstract. In Rina Dechter, editor, *Principles and Practice of Constraint Programming - CP 2000, 6th International Conference, Singapore, September 18-21, 2000, Proceedings*, volume 1894 of *Lecture Notes in Computer Science*, pages 13–17. Springer, 2000.

(Meyer 00) Bernd Meyer. A constraint-based framework for diagrammatical reasoning. *Journal of Applied Artificial Intelligence*, 14:327–244, 2000.

(Morawietz 00) Frank Morawietz. Chart parsing as contraint propagation. Proceedings of COLING-2000, 2000.

(Penn 00) G. Penn. Applying Constraint Handling Rules to HPSG. Workshop on Rule-Based Constraint Reasoning and Programming. Available at http://www.cs.cmu.edu/~gpenn/trale.ps.gz, 2000.

(Pereira & Warren 80) Fernando C. N. Pereira and David H. D. Warren. Definite clause grammars for language analysis—A survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13(3):231–278, 1980.

(Schröder *et al.* 00) I. Schröder, W. Menzel, K. Foth, and M. Schulz. Dependency modelling with restricted constraints. *International Journal Traitement Automatique des Langues: Les grammaires de dépendance*, 41(1):113–144, 2000.

(Shieber *et al.* 95) Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36, 1995.

(Swedish Institute of Computer Science 03) Swedish Institute of Computer Science. SICStus Prolog user's manual, Version 3.10. Most recent version available at `http://www.sics.se/isl`, 2003.