On the Satisfiability of Complex Constraints

Luis Damas
Universidade do Porto, Rua do Campo Alegre 823, 4000 Porto, Portugal
Giovanni B. Varile
CEC , 2920 Luxembourg, Luxembourg

Abstract

The main problem arising from the use of complex constraints in Computational Linguists is due to the NP-hardness of checking whether a given set of constraints is satisfiable and, in the affirmative case, of generating the set of minimal models which satisfy the constraints. In this paper we show how the CLG approach to rewriting constraints while constructing a partial model can greatly reduce the size of the original constraints and thus contribute to reduce the computational problem.

1 Introduction

Most of the more recent formalisms in Computational Linguistics involve complex constraints, i.e. involving either disjunction or and negation of equality constraints [Kay85, KB82, PS87].

From a practical point of view, the main problem arising from the introduction of complex constraints is due to the NP-hardness of checking whether a given set of constraints is satisfiable and, in the affirmative case, of generating the set of minimal models which satisfy the constraints.

Since the NP-hardness of the problem tends to manifest in practical applications in a dramatic way there were several proposals of algorithms to minimize this problem [Kas87, ED88, MK91].

However, in our opinion, any practical approach to the problem should rely on applying an exponential satisfaction algorithm as seldom as possible and also, when that cannot be avoided, to reduce as much as possible the size of the input formulae to which the algorithm is applied.

A classical way in which one can reduce the size of input formulae is by factoring, i.e. expressing it as a conjunction of smaller formulae which do not have variables in common. As a matter of fact if instead of checking the satisfiability of a formula of length nk we check the satisfiability of n formulae of size k we are gaining an exponential reduction in time. This strategy is also explored in [MK91].

Another way of reducing the size of input formulae which has been used in CLG implementations [DV89, BDMV90, DMV91b], and which can be combined with the previous approaches, is to build a partial model of the input formula and to apply the algorithm to the remaining constraints. Since in many cases this is able to do away with all the constraints this also achieves the goal of using the exponential satisfaction algorithm as seldom as possible. This method can also be seen as a generalization of steps 1 and 2 of the algorithm described in [Kas87], while avoiding some of its pitfalls.

In this paper we start by showing, by translating to the feature logics context the methods for first order terms introduced in [DMV91a, DMV91b], how the CLG approach to rewriting constraints while constructing a partial model can greatly reduce the size of the original constraints and thus contribute to reduce the computational problem.

After that will review the notion of factorization for feature logic formulae showing under which conditions it can be done.

2 An example

From an intuitive point of view, our approach attempts to explore the fact that complex constraints arising from combining descriptions of linguistic objects, tend to take the form of a large conjunction of smaller constraints.

To illustrate this approach we will use the following example from [MK91] originated by conjoining the agreement features of the lexical information for the German words *die* and *Koffer*.

$$\begin{aligned}&[(f\,case) = nom \lor (f\,case) = acc] \land\\ &[[(f\,gend) = fem \land (f\,num) = sg] \lor\\ &\quad (f\,num) = pl] \land\\ &(f\,gend) = masc \land (f\,pers) = 3 \land\\ &[[(f\,num) = sg \land (f\,case) \neq gen] \lor\\ &\qquad [(f\,num) = pl \land (f\,case) \neq dat]] \qquad (1)\end{aligned}$$

By looking at the top equality conjuncts in (1) we can conclude that any model of (1) is subsumed by

$$f = \begin{bmatrix} gend = masc \\ pers = 3 \end{bmatrix} \qquad (2)$$

Using this information and the standard axioms of Feature Logics we can rewrite (1) as follows

$$\begin{aligned}&[(f\,case) = nom \lor (f\,case) = acc] \land\\ &[[false \land (f\,num) = sg] \lor\\ &\quad (f\,num) = pl] \land\\ &true \land true \land\\ &[[(f\,num) = sg \land (f\,case) \neq gen] \lor\\ &\qquad [(f\,num) = pl \land (f\,case) \neq dat]] \qquad (3)\end{aligned}$$

which, by using the standard rules of Propositional Calculus, can be simplified to

$$\begin{aligned}&[(f\,case) = nom \lor (f\,case) = acc] \land\\ &\quad (f\,num) = pl \land\\ &\quad [[(f\,num) = sg \land (f\,case) \neq gen] \lor\\ &\qquad [(f\,num) = pl \land (f\,case) \neq dat]] \qquad (4)\end{aligned}$$

Again, from the atomic top conjunct in (4) we can refine (2) to obtain

$$f = \begin{bmatrix} gend = masc \\ pers = 3 \\ num = pl \end{bmatrix} \qquad (5)$$

after which (4) reduces to

$$\begin{aligned}&[(f\,case) = nom \lor (f\,case) = acc] \land\\ &\qquad\qquad (f\,case) \neq dat \qquad (6)\end{aligned}$$

Since no atomic top conjunct remains we would now have to resort to the exponential algorithm to check the satisfiability of (6), which, as it will be shown in the next section, is equivalent to the satisfiability of (1). However we should point out that the above process, which takes at most quadratic time since, by using adequate data structures, each rewrite can be done in linear time and the number of rewrites is limited by the size of the formula, has succeed in drastically reducing the size of the original formula.

3 A rewriting system

In this section we introduce a rewriting system, which is essentially an adaptation to feature structures of the one presented in [DMV91a] for constraints on first order terms.

The purpose of this rewriting system is to formalize the rewriting process illustrated in the previous section. The rules of the rewriting system, which are justified by theorems of the logic underlying features structures, have been designed with the aim of bringing out conjunctions to the top while avoiding more than a linear growth of the size of constraints.

To formalize the rewriting system we will use Feature Logics [Smo89] and its notation. Thus we will use a and b to denote atoms, f to denote a feature name, x and y to denote variables and s and t to denote either an atom or a variable.

We start by recalling the notion of ***solved feature clause*** of [Smo89] which is the feature logic version of the standard definition of solved form [Her30, Mah88].

A set of formulae C is said to be a solved feature clause if it satisfies the following conditions:

1. every constraint in C has one of the following forms: $fx \doteq s$, $fx \uparrow$, $x \doteq s$, $x \dot{\neq} s$
2. if $x \doteq s$ is in C then x occurs exactly once in C
3. if $fx \doteq s$ and $fx \doteq t$ are in C then $s = t$
4. if $fx \uparrow$ is in C, then C contains no constraint $fx \doteq s$
5. if $x \dot{\neq} s$ is in C, then $x \neq s$.

Smolka also shows that any satisfiable set C of, possibly negated, atomic formulae can be reduced to solved form by using the following simplification rules, which again are the feature logic version of Herbrand's rules for solving sets of equations of first order terms:

1. $\{x\dot{=}s\} \cup \mathcal{C} \rightarrow \{x\dot{=}s\} \cup [s/x]\mathcal{C}$ if $x \neq s$ and x occurs in $\mathcal{C}$
2. $\{a\dot{=}x\} \cup \mathcal{C} \rightarrow \{x\dot{=}a\} \cup \mathcal{C}$
3. $\{fx\dot{=}s, fx\dot{=}t\} \cup \mathcal{C} \rightarrow \{fx\dot{=}s, s\dot{=}t\} \cup \mathcal{C}$
4. $\{s\dot{=}s\} \cup \mathcal{C} \rightarrow \mathcal{C}$
5. $\{fa \uparrow\} \cup \mathcal{C} \rightarrow \mathcal{C}$
6. $\{a\dot{\neq}x\} \cup \mathcal{C} \rightarrow \{x\dot{\neq}a\} \cup \mathcal{C}$
7. $\{a\dot{\neq}b\} \cup \mathcal{C} \rightarrow \mathcal{C}$ if $a \neq b$.

We call a solved feature clause *positive* iff it includes only constraints of the form $fx\dot{=}s$ and $x\dot{=}s$.

We can now make precise the notion of partial model used in the previous section as a positive solved feature clause $\mathcal{M}$.

Note that the form required for $\mathcal{M}$ is essentially the one produced by an unification algorithm for feature structures.

Given a set of constraints $\mathcal{C}$ we say that a feature clause $\mathcal{C}'$ is a minimal model of $\mathcal{C}$ if every model of $\mathcal{C}'$ is a model of $\mathcal{C}$ and no proper subset of $\mathcal{C}'$ satisfies this condition.

From Theorem 5.6 of [Smo89] we can conclude that for any $\mathcal{C}$ there is a finite number of minimal models of $\mathcal{C}$.

The aim of the CLG rewriting system is to produce from a set of constraints $\mathcal{C}_0$ a partial model $\mathcal{M}$ and a smaller set of constraints $\mathcal{C}$ such that any minimal model of $\mathcal{C}_0$ can be obtained by conjoining (i.e. "unifying") a minimal model of $\mathcal{C}$ with $\mathcal{M}$ and moreover for any minimal model of $\mathcal{C}$ the reunion $\mathcal{M} \cup \mathcal{C}$ is satisfiable.

We start by defining a set of rewriting rules $\rightarrow_{\mathcal{M}}$ for terms and constraints:

$x \rightarrow_{\mathcal{M}} t$	if $x\dot{=}t \in \mathcal{M}$
$b\dot{=}x \rightarrow_{\mathcal{M}} x\dot{=}b$	
$fa\dot{=}t \rightarrow_{\mathcal{M}} false$	
$fx \uparrow \rightarrow_{\mathcal{M}} false$	if $fx\dot{=}t \in \mathcal{M}$
$a\dot{=}b \rightarrow_{\mathcal{M}} false$	
$t\dot{=}t \rightarrow_{\mathcal{M}} true$	
$f\ x\dot{=}s \rightarrow_{\mathcal{M}} t\dot{=}s$	if $f\ x\dot{=}t \in \mathcal{M}$
$\neg true \rightarrow_{\mathcal{M}} false$	
$\neg false \rightarrow_{\mathcal{M}} true$	
$\neg\neg C \rightarrow_{\mathcal{M}} C$	
$\neg(C \vee C') \rightarrow_{\mathcal{M}} \neg C \wedge \neg C'$	
$C \wedge true \rightarrow_{\mathcal{M}} C$	
$C \wedge false \rightarrow_{\mathcal{M}} false$	
$C \vee true \rightarrow_{\mathcal{M}} true$	
$C \vee false \rightarrow_{\mathcal{M}} C$	

We now define a rewriting system for pairs $\langle \mathcal{M}, \mathcal{C} \rangle$ by first closing $\mathcal{C}$ under $\rightarrow_{\mathcal{M}}$ and then using the following rules

$$\begin{array}{l}
\langle \mathcal{M}, \mathcal{C} \cup \{false\} \rangle \rightarrow \langle \{false\}, \emptyset \rangle \\
\langle \mathcal{M}, \mathcal{C} \cup \{true\} \rangle \rightarrow \langle \mathcal{M}, \mathcal{C} \rangle \\
\langle \mathcal{M}, \mathcal{C} \cup \{x\dot{=}s\} \rangle \rightarrow \langle \mathcal{M} \cup \{x\dot{=}s\}, \mathcal{C} \rangle \\
\langle \mathcal{M}, \mathcal{C} \cup \{fx\dot{=}t\} \rangle \rightarrow \langle \mathcal{M} \cup \{fx\dot{=}t\}, \mathcal{C} \rangle
\end{array}$$

with the convention that after each application of one of the rewrite rules the new partial model is reduced to solved form and the resulting set of constraints is closed under $\rightarrow_{\mathcal{M}}$.

As in [DMV91a] the rewrite process could be extended to use negated atomic constraints in $\mathcal{C}$.

We will now sketch the proof of the claims made above about the rewriting system.

We will first argue that if given an initial set of constraints $\mathcal{C}_0$ we apply the rewriting system to $\langle \emptyset, \mathcal{C}_0 \rangle$ to obtain $\langle \mathcal{M}, \mathcal{C} \rangle$ then $\mathcal{C}_0$ (more precisely the conjunct of all the constraints in $\mathcal{C}_0$) is equivalent to $\mathcal{M} \cup \mathcal{C}$. As a matter of fact this follows from the fact that each rewrite rule is associated with a similar meta-theorem of First Order Logic and/or the axioms of Feature Logics.

As for the other property of the rewriting system, namely that the minimal models of $\mathcal{C}_0$ are obtained by conjoining $\mathcal{M}$ with those of $\mathcal{C}$, we will only sketch the argument of the proof which follows from the fact that if some minimal model $\mathcal{C}'$ of $\mathcal{C}$ was inconsistent with $\mathcal{M}$ then, after reducing $\mathcal{C}$ to disjunctive form, at least one of the disjunctions should subsume $\mathcal{C}'$ and thus should be inconsistent with $\mathcal{M}$ while admiting itself a model. Now, since every atomic formula occurring in the disjunction already occured in $\mathcal{C}$ it is possible to derive a contradiction with the hypothesis that $\mathcal{C}$ was closed under $\rightarrow_{\mathcal{M}}$. To see this we notice that each disjunct in the disjunctive form of $\mathcal{C}$ can be regarded as feature clause $\mathcal{C}''$. Now if $\mathcal{M} \cup \mathcal{C}''$ was unsatisfiable then some sequence of simplification rules should lead to a clash. Now, noticing that any atomic formula in $\mathcal{C}''$ must already be present in $\mathcal{C}$, it is easy to check that any sequence of simplification rules would involve only formulae from $\mathcal{C}''$ and a clash with a formula in $\mathcal{M}$ would thus be impossible.

4 On the factorization of feature constraints

Although the rewriting system introduced in the previous section can be seen as a factorization of the original constraint it is also important, in accordance with the discussion in the introduction, to further factorizing the constraint produced before an exponential satisfaction algorithm is applied.

Now given a conjunction $C \wedge C'$ of feature constraints it is easy to prove, using arguments similar to the ones used in the previous section, that a sufficient condition for the conjunction to be satisfiable iff each of the conjuncts is independently satisfiable, can be expressed as follows:

1. any variable x which occurs in C in an atomic formula $x \doteq s$ or $s \doteq x$ does not occur in C' and vice-versa
2. for every variable x and feature f such that fx occurs in C, fx does not occur in C' and vice-versa.

Note that given a conjunctive set of constraints it is possible to partition it into minimal disjoint sets satisfying the conditions above in quadratic time.

Using the above criteria results in achieving the objective described in [MK91] of separating not only the treatment of constraints dealing with completely independent linguistic descriptions but also of independent phenomena for the same the linguistic descriptions.

5 Conclusions

The strategy for handling complex constraints described in this paper has been successfully put to use in different CLG implementations.

One of the advantages of using a rewrite system is that it is easily extended to other atomic constraints like the ones in CLG(2) dealing with lists. Similarly it can be adapted to cope with the extensions to feature constraints proposed in [Joh91].

References

[BDMV90] Sergio Balari, Luis Damas, Nelma Moreira, and Giovanni B. Varile. CLG: Constraint logic grammars. In H.Karlgren, editor, *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, volume 3, pages 7–12, Helsinki, 1990.

[DMV91a] Luis Damas, Nelma Moreira, and Giovanni B. Varile. The formal and processing models of CLG. In *Fiifth Conference of the European Chapter of the Association for Computational Linguistics*, pages 173–178, Berlin, 1991.

[DMV91b] Luis Damas, Nelma Moreira, and Giovanni B. Varile. CLG - a general logic grammar formalism. In *Proceedings of the Workshop on Constraint Propagation and Linguistic Description, forthcoming*, Lugano, Switzerland, 1991.

[DV89] Luis Damas and Giovanni B. Varile. CLG: A grammar formalism based on constraint resolution. In E.M.Morgado and J.P.Martins, editors, *EPIA 89*, volume 390 of *Lecture Notes in Artificial Intelligence*, pages 175–186. Springer Verlag, 1989.

[ED88] A. Eisele and J. Dörre. Unification of disjunctive feature descriptions. In *26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, New York, 1988.

[GJW86] Barbara Grosz, Karen Sparck Jones, and Bonny Lynn Webber, editors. *Readings in Natural Language Processing*. Morgan Kaufmann, 1986.

[Her30] Jacques Herbrand. *Recherches sur la theorie de la démonstration, Thèse de Doctorat*. Univ. Paris, Paris, 1930.

[Joh91] Mark Johnson. Features and formulae. *Computational Linguistics*, 17(2), 1991.

[Kas87] R. T. Kasper. A unification method for disjunctive feature descriptions. In *25th Annual Meeting of the Association for Computational Linguistics*, Stanford, CA, 1987.

[Kay85] Martin Kay. Parsing in functional unification grammar. In David R. Dowty, Lauri Karttunen, and Arnold M. Zwicky, editors, *Natural Language Parsing*. Ellis Horwood/Wiley, 1985. also appears in [GJW86].

[KB82] R. Kaplan and J. Bresban. Lexical functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations.* MIT Press, 1982.

[Mah88] Michael J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. Technical report, IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, U.S.A., 1988.

[MK91] John T. Maxwell and Ronald M. Kaplan. A method for disjunctive constraint satisfaction. In Masaru Tomita, editor, *Current Issues in Parsing Technology.* Kluwer Academic Publishers, 1991.

[PS87] Carl Pollard and Ivan Sag. *Information Based Syntax and Semantics, Volume 1, Fundamentals,* volume 13. Center for the Study of Language and Information Stanford, 1987.

[Smo89] Gert
Smolka. Feature constraint logics for unification grammars. Technical report, IBM Wissenschafliches Zentrum, Institut für Wissensbasierte Systeme, 1989. IWBS Report 93.