Resolution of Constraints in Algebras of Rational Trees

Luis Damas, Nelma Moreira, Sabine Broda
{luis,nam,sbb}@ncc.up.pt
LIACC, Universidade do Porto
Rua do Campo Alegre, 823, 4100 Porto
Portugal

Abstract. This work presents a constraint solver for the domain of rational trees. Since the problem is NP-hard the strategy used by the solver is to reduce as much as possible, in polynomial time, the size of the constraints using a rewriting system before applying a complete algorithm. This rewriting system works essentially by rewriting constraints using the information in a partial model. An efficient C implementation of the rewriting system is described and an algorithm for factoring complex constraints is also presented.

1 Introduction

The topic of constraints over syntactic domains has raised a considerable interest in the Computational Linguistics community. As a matter of fact constraints over the algebra of rational trees can significantly contribute to the reduction of the search space of problems in NLP while increasing the expressive power of formalisms for NLP [DMV91, DV89].

Unification-based grammar formalisms ([SUP+83], [Usz86],[KB82], [PS87], etc.) describe linguistic information by means of constraints over feature structures, which are basically sets of attribute-value pairs, where values can be atomic symbols or embedded feature structures, e.g.

\[ \text{[cat = np, agr = [num = sg, pers = 3rd]]} \]

These structures and their combination can be seen as conjunctions of equality constraints which satisfiability can be tested by efficient unification algorithms. But the extension of these formalisms to express complex constraints involving negation and disjunction of equality constraints, besides arising formal theoretical problems leads to a NP-hard satisfiability problem.

From a formal point of view the problem is well understood after the foundational works of [RK86, Smo89] establishing Feature Logics. It turns out that the standard model for feature logics, namely rational trees, has a close relationship with the standard algebra of rational trees in Logic Programming and with its complete axiomatization presented in [Mah88]. As a matter of fact it can be proved [DMV92] that the satisfiability problem for the complete axiomatization
of feature logics can be reduced to the satisfiability problem for Maher complete axiomatization of the algebra of rational trees.

From a practical point of view the fact that the satisfiability problem is NP-hard tends to manifest itself in a dramatic way in practical applications motivated several specialized algorithms to minimize this problem [Kas87, ED88, MK91].

In [DV92] it was argued that any practical approach to the satisfiability problem should use factorization techniques to reduce the size of the input formulae to which any complete algorithm for satisfiability is applied, since such factorization can reduce by an exponential factor the overall computational cost of the process. In that work a rewrite system, working in polynomial time, was used to factor out deterministic information contained in a complex constraint and simplify the remaining formula using that deterministic information. In [DMV92] that rewrite system was extended to a complete rewrite system for satisfiability which avoided as much as possible the multiplication of disjunctions which is the origin of the NP-hardness of the satisfiability problem.

In this work we present a solver for constraints over the domain of rational trees using the rewrite system mentioned above.

The rest of this paper proceeds as follows. We start by defining our constraint language, which was designed to enable the introduction of equality constraints on terms or rational trees without using any quantifiers, in section 2. In section 3 we define a complete rewrite system for expressions in the constraint language. In section 4 we present in some detail the low-level implementation of part of the rewrite system. In section 5 we will present an algorithm for factoring out of two constraints any deterministic information which is common to both.

2 The Constraint Language

Consider the first order language \( \mathcal{L} \) built from the countable sets of variables \( Vars = \{x, y, z, \ldots\} \), function symbols \( F = \{f, g, h, \ldots\} \) and equality as the only predicate symbol. As usual, the functional symbols of arity 0 will be denoted by \( a, b, \ldots \) and will be referred to as atoms. For this language Maher presented a complete axiomatization of the algebra of rational trees, \( \mathcal{RT} \), [Mah88]. This theory of \( \mathcal{RT} \) is complete in the sense that a sentence is valid in \( \mathcal{RT} \) if and only if it is a logical consequence of the theory. We now introduce a quantifier free constraint language, in which the formulae of \( \mathcal{L} \) will be encoded. Here function symbols appear, whenever necessary, with their arities and are then denoted by \( f^n \), while \( f^n_i \) stands for the \( i \)-th projection of a function symbol \( f \) of arity \( n \). The letters \( s \) and \( t \) will always denote variables or atoms. Expressions of the form \( x.f^n_i \) will be called slots. We define the constraints of the language by:

\[
c ::= t.f^n | t = t | t.f^n_i = t, \ 1 \leq i \leq n |
false | true | \neg c | c \land c | c \lor c
\]