1 Introduction

The resolution method for (propositional) logic due to J.A. Robinson [4] (1965) is well-known to be a sound and complete procedure for checking the unsatisfiability of a set of clauses. However, it appears that the completeness proofs that can be found in the literature (for instance, Chang and Lee [1], Lewis and Papadimitriou [3], Robinson [5]) are existence proofs that proceed by contradiction to show that if a set of clauses is unsatisfiable, then it must have a resolution refutation because otherwise a satisfying assignment can be obtained. In particular, none of these proofs yields (directly) an algorithm producing a resolution refutation from an unsatisfiable set of clauses. In that sense, these proofs are nonconstructive. In Gallier [2] (1986), we gave a completeness proof based on an algorithm for converting a Gentzen-like proof (using sequents) into a resolution DAG (see Chapter 4). Such a method is more constructive than the others but, we found later on that it is possible to give a simple and constructive proof of the completeness of propositional resolution which consists of an algorithm together with a proof of its correctness. This algorithm and its correctness are the object of this note.

It should be noted that Judith Underwood gave other constructive proof procedures in her Ph.D. thesis, notably for the intuitionistic propositional calculus [6].

2 Review of Propositional Resolution

Recall that a *literal*, L, is either a propositional letter, P, or the negation, $\neg P$, of a propositional letter. A *clause* is a finite set of literals, $\{L_1, \ldots, L_k\}$, interpreted as the disjunction $L_1 \vee \cdots \vee L_k$ (when $k = 0$, this is the empty clause denoted $\square$). A set of clauses, $\Gamma = \{C_1, \ldots, C_n\}$, is interpreted as the conjunction $C_1 \wedge \cdots \wedge C_n$. For short, we write $\Gamma = C_1, \ldots, C_n$.

The *resolution method* (J.A. Robinson [4]) is a procedure for checking whether a set of clauses, Γ, is unsatisfiable. The resolution method consists in building a certain kind of labeled DAG whose leaves are labeled with clauses in Γ and whose interior nodes are labeled according to the *resolution rule*. Given two clauses $C = A \cup \{P\}$ and $C' = B \cup \{\neg P\}$ (where P is a propositional letter, $P \notin A$ and $\neg P \notin B$), the *resolvent of C and C'* is the clause

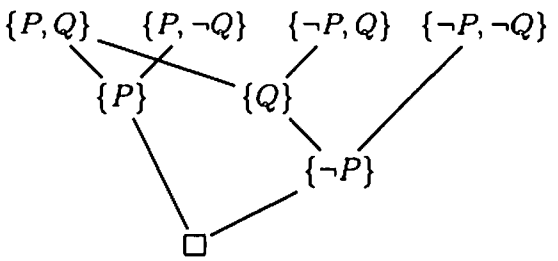
$$R = A \cup B$$

obtained by cancelling out P and $\neg P$. A *resolution DAG for* Γ is a DAG whose leaves are labeled with clauses from Γ and such that every interior node n has exactly two predecessors, n_1 and n_2 so that n is labeled with the resolvent of the clauses labeling n_1 and n_2. In a resolution step involving the nodes, n_1, n_2 and n, as above, we say that the two clauses C and C' labeling the nodes n_1 and n_2 are the *parent clauses* of the resolvent clause, R, labeling the node n. In a resolution DAG, D, a clause, C' is said to be a *descendant* of a clause, C, iff there is a (directed) path from some node labeled with C to a node labeled with C'. A *resolution refutation for* Γ is a resolution DAG with a single root whose label is the empty

clause. (For more details on the resolution method, resolution DAGs, etc., one may consult Gallier [2], Chapter 4, or any of the books cited in Section 1.)

Here is an example of a resolution refutation for the set of clauses

$$\Gamma = \{\{P, Q\}, \{P, \neg Q\}, \{\neg P, Q\}, \{\neg P, \neg Q\}\}:$$



3 Completeness of Propositional Resolution: An Algorithm and its Correctness

Let Γ be a set of clauses. Thus, Γ is either the empty clause, $\square$, or it is a conjunction of clauses, $\Gamma = C_1, \ldots, C_n$. We define the *complexity*, $c(C)$, of a clause, C, as the number of disjunction symbols in C; i.e., if C consists of a single literal (i.e., $C = \{L\}$, for some literal, L), then $c(C) = 0$, else if $C = \{L_1, \ldots, L_m\}$ (with $m \geq 2$) where the L_i's are literals, then $c(C) = m - 1$ (we also set $c(\square) = 0$). If Γ is a conjunction of clauses, $\Gamma = C_1, \ldots, C_n$, then we set

$$c(\Gamma) = c(C_1) + \cdots + c(C_n).$$

We now give a recursive algorithm, `buildresol`, for constructing a resolution DAG from any set of clauses and then prove its correctness, namely, that if the input set of clauses is unsatisfiable, then the output resolution DAG is a resolution refutation. This establishes the completeness of propositional resolution constructively.

Our algorithm makes use of two functions, `percolate`, and `graft`.

1. The function `percolate`(D, A, L)

The inputs are: a resolution DAG, D, some selected leaf of D labeled with a clause, A, and some literal, L. This function adds the literal L to the clause A to form the clause $A \cup \{L\}$ and then "percolates" L down to the root of D. More precisely, we construct the resolution DAG, D', whose underlying unlabeled DAG is identical to D, as follows: Since D and D' have the same unlabeled DAG we refer to two nodes of D of D' as *corresponding nodes* if they are identical in the underlying unlabeled DAG. Consider any resolution step of D. If both parent clauses are not descendants of the premise A, then the corresponding resolution step of D' is the same. If the parent clauses in D are C and C' where C' is a descendant of the premise A (resp. C is a descendant of the premise A) and if R is the