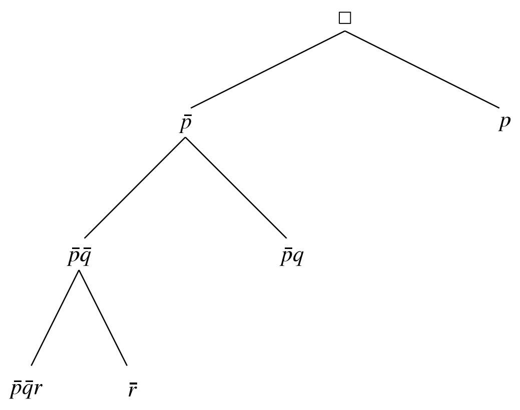


Fig. 4.1 A resolution refutation represented as a tree

Definition 4.20 A derivation of $\Box$ from a set of clauses S is a *refutation by resolution* of S or a *resolution refutation* of S. ■

Since $\Box$ is unsatisfiable, by Theorem 4.17 if there exists a refutation of S by resolution then S is unsatisfiable.

In Example 4.19, we derived the unsatisfiable clause $\Box$, so we conclude that the set of clauses S is unsatisfiable. We leave it to the reader to check that S is the clausal form of $\neg A$ where A is an instance of Axiom 2 of $\mathscr{H}$ $(p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$. Since $\neg A$ is unsatisfiable, A is valid.

4.4 Soundness and Completeness of Resolution *

The soundness of resolution follows easily from Theorem 4.17, but completeness is rather difficult to prove, so you may want to skip the this section on your first reading.

Theorem 4.21 *If the set of clauses labeling the leaves of a resolution tree is satisfiable then the clause at the root is satisfiable.*

The proof is by induction using Theorem 4.17 and is left as an exercise.

The converse to Theorem 4.21 is not true because we have no way of ensuring that the extensions made to $\mathscr{I}$ on all branches are consistent. In the tree in Fig. 4.2, the set of clauses on the leaves $S = \{r, pq\bar{r}, \bar{r}, p\bar{q}r\}$ is not satisfiable even though the clause p at the root is satisfiable. Since S is unsatisfiable, it has a refutation: whenever the pair of clashing clauses r and $\bar{r}$ is chosen, the resolvent will be $\Box$.

Resolution is a refutation procedure, so soundness and completeness are better expressed in terms of unsatisfiability, rather than validity.

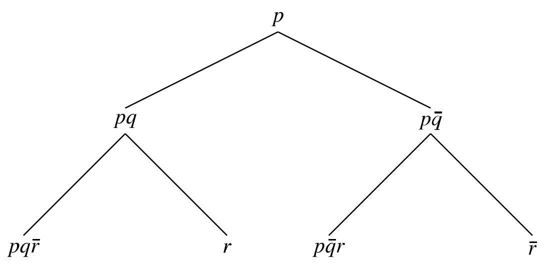


Fig. 4.2 Incomplete resolution tree

Corollary 4.22 (Soundness) *Let S be a set of clauses. If there is a refutation by resolution for S then S is unsatisfiable.*

Proof Immediate from Theorem 4.21 and Lemma 4.10. ■

Theorem 4.23 (Completeness) *If a set of clauses is unsatisfiable then the empty clause $\Box$ will be derived by the resolution procedure.*

We have to prove that given an unsatisfiable set of clauses, the resolution procedure will eventually terminate producing $\Box$, rather than continuing indefinitely or terminating but failing to produce $\Box$. The resolution procedure was defined so that the same pair of clauses is never chosen more than once. Since there are only a finite number of distinct clauses on the finite set of atomic propositions appearing in a set of clauses, the procedure terminates. We need only prove that when the procedure terminates, the empty clause is produced.

Semantic Trees

The proof will use *semantic trees* (which must not be confused with semantic tableaux). A semantic tree is a data structure for recording assignments of T and F to the atomic propositions of a formula in the process of searching for a model (satisfying interpretation). If the formula is unsatisfiable, the search for a model must end in failure. Clauses that are created during a resolution refutation will be associated with nodes of the tree called failure nodes; these nodes represent assignments that falsify the associated clauses. Eventually, the root node (associated with the empty clause $\Box$) will be shown to be a failure node.

Definition 4.24 (Semantic tree) Let S be a set of clauses and let $P_S = \{p_1, \ldots, p_n\}$ be the set of atomic propositions appearing in S. $\mathscr{T}$, the *semantic tree* for S, is a complete binary tree of depth n such that for $1 \leq i \leq n$, every left-branching edge from a node at depth $i - 1$ is labeled p_i and every right-branching edge is labeled by $\bar{p}_i$.

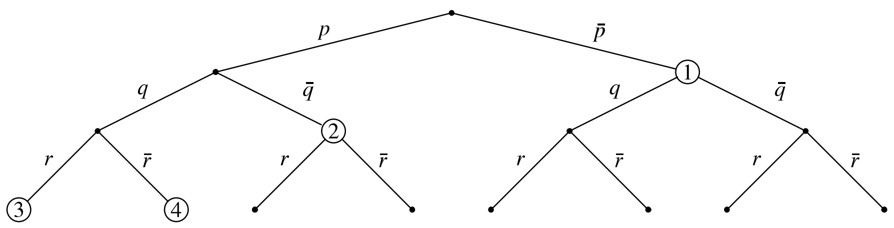


Fig. 4.3 Semantic tree

Every branch b from the root to a leaf in $\mathscr{T}$ is labeled by a sequence of literals $\{l_1, \ldots, l_n\}$, where $l_i = p_i$ or $l_i = \bar{p}_i$. b defines an interpretation by:

$$\begin{aligned} \mathscr{I}_b(p_i) &= T \quad \text{if } l_i = p_i, \\ \mathscr{I}_b(p_i) &= F \quad \text{if } l_i = \bar{p}_i. \end{aligned}$$

A branch b is *closed* if $v_b(S) = F$, otherwise b is *open*. $\mathscr{T}$ is *closed* if all branches are closed, otherwise $\mathscr{T}$ is *open*. ■

Example 4.25 The semantic tree for $S = \{p,\ \bar{p}q,\ \bar{r},\ \bar{p}\bar{q}r\}$ is shown in Fig. 4.3 where the numbers on the nodes will be explained later. The branch b ending in the leaf labeled 4 defines the interpretation:

$$\mathscr{I}_b(p) = T, \qquad \mathscr{I}_b(q) = T, \qquad \mathscr{I}_b(r) = F.$$

Since $v_{\mathscr{I}_b}(\bar{p}\bar{q}r) = F$, $v_{\mathscr{I}_b}(S) = F$ (a set of clauses is the conjunction of its members) and the branch b is closed. We leave it to the reader to check that every branch in this tree is closed. ■

Lemma 4.26 *Let S be a set of clauses and let $\mathscr{T}$ a semantic tree for S. Every interpretation $\mathscr{I}$ for S corresponds to $\mathscr{I}_b$ for some branch b in $\mathscr{T}$, and conversely, every $\mathscr{I}_b$ is an interpretation for S.*

Proof By construction. ■

Theorem 4.27 *The semantic tree $\mathscr{T}$ for a set of clauses S is closed if and only if the set S is unsatisfiable.*

Proof Suppose that $\mathscr{T}$ is closed and let $\mathscr{I}$ be an arbitrary interpretation for S. By Lemma 4.26, $\mathscr{I}$ is $\mathscr{I}_b$ for some branch in $\mathscr{T}$. Since $\mathscr{T}$ is closed, $v_b(S) = F$. But $\mathscr{I} = \mathscr{I}_b$ was arbitrary so S is unsatisfiable.

Conversely, let S be an unsatisfiable set of clauses, $\mathscr{T}$ the semantic tree for S and b an arbitrary branch in $\mathscr{T}$. Then v_b is an interpretation for S by Lemma 4.26, and $v_b(S) = F$ since S is unsatisfiable. Since b was arbitrary, $\mathscr{T}$ is closed. ■

Failure Nodes

When traversing a branch of the semantic tree top-down, a (partial) branch from the root to a node represents a partial interpretation (Definition 2.18) defined by the labels of the edges that were traversed. It is possible that this partial interpretation is sufficiently defined to evaluate the truth value of some clauses; in particular, some clause might evaluate to F. Since a set of clauses is an implicit conjunction, if even one clause evaluates to F, the partial interpretation is sufficient to conclude that the entire set of clauses is false. In a *closed* semantic tree, there must be such a node on every branch. However, if a clause contains the literal labeling the edge to a leaf, a (full) interpretation may be necessary to falsify the clause.

Example 4.28 In the semantic tree for $S = \{p,\ \bar{p}q,\ \bar{r},\ \bar{p}\bar{q}r\}$ (Fig. 4.3), the partial branch $b_{p\bar{q}}$ from the root to the node numbered 2 defines a partial interpretation $\mathscr{I}_{b_{p\bar{q}}}(p) = T$, $\mathscr{I}_{b_{p\bar{q}}}(q) = F$, which falsifies the clause $\bar{p}q$ and thus the entire set of clauses S.

Consider now the partial branches b_p and b_{pq} and the full branch b_{pqr} that are obtained by always taking the child labeled by a positive literal. The partial interpretation $\mathscr{I}_{b_p}(p) = T$ does not falsify any of the clauses, nor does the partial interpretation $\mathscr{I}_{b_{pq}}(p) = T$, $\mathscr{I} b_{pq}(q) = T$. Only the full interpretation $\mathscr{I}_{b_{pqr}}$ that assigns T to r falsifies one of the clauses ($\bar{r}$). ■

Definition 4.29 Let $\mathscr{T}$ be a closed semantic tree for a set of clauses S and let b be a branch in $\mathscr{T}$. The node in b closest to the root which falsifies S is a *failure node*.

Example 4.30 Referring again to Fig. 4.3, the node numbered 2 is a failure node since neither its parent node (which defines the partial interpretation $\mathscr{I}_{b_p}$) nor the root itself falsifies any of the clauses in the set. We leave it to the read to check that all the numbered nodes are failure nodes. ■

Since a failure node falsifies S (an implicit conjunction of clauses), it must falsify at least once clause in S.

Definition 4.31 A clause falsified by a failure node is a *clause associated with the node*. ■

Example 4.32 The failure nodes in Fig. 4.3 are labeled with the number of a clause associated with it; the numbers were given in Examples 4.19. ■

It is possible that more than one clause is associated with a failure node; for example, if q is added to the set of clauses, then q is another clause associated with failure node numbered 2.

We can characterize the clauses associated with failure nodes. For C to be falsified at a failure node n, *all* the literals in C must be assigned F in the partial interpretation.

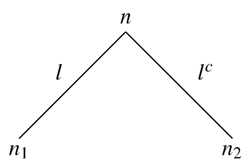


Fig. 4.4 Inference and failure nodes

Example 4.33 In Fig. 4.3, $\bar{r}$ is a clause associated with the failure node numbered 3. $\{\bar{r}\}$ is a proper subset of $\{\bar{p}, \bar{q}, \bar{r}\}$, the set of *complements* of the literals assigned to on the branch. ■

Lemma 4.34 *A clause C associated with a failure node n is a subset of the complements of the literals appearing on the partial branch b from the root to n.*

Proof Let $C = l_1 \cdots l_k$ and let $E = \{e_1, \ldots, e_m\}$ be the set of literals labeling edges in the branch. Since C is the clause associated with the failure node n, $v_b(C) = F$ for the interpretation $\mathscr{I}_b$ defined by $\mathscr{I}_b(e_j) = T$ for all $e_j \in E$. C is a disjunction so for each $l_i \in C$, $\mathscr{I}_b(l_i)$ must be assigned F. Since $\mathscr{I}_b$ only assigns to the literals in E, it follows that $l_i = e_j^c$ for some $e_j \in E$. Therefore, $C = l_1 \cdots l_k \subseteq \{e_1^c, \ldots, e_m^c\}$. ■

Inference Nodes

Definition 4.35 n is an *inference node* iff its children are failure nodes. ■

Example 4.36 In Fig. 4.3, the parent of nodes 3 and 4 is an inference node. ■

Lemma 4.37 *Let $\mathscr{T}$ be a closed semantic tree for a set of clauses S. If there are at least two failure nodes in $\mathscr{T}$, then there is at least one inference node.*

Proof Suppose that n_1 is a failure node and that its sibling n_2 is not (Fig. 4.4). Then no ancestor of n_2 can be a failure node, because its ancestors are also ancestors of n_1, which is, by assumption, a failure node and thus the node *closest* to the root on its branch which falsifies S.

$\mathscr{T}$ is closed so every branch in $\mathscr{T}$ is closed, in particular, any branch b that includes n_2 is closed. By definition of a closed branch, $\mathscr{I}_b$, the full interpretation associated with the leaf of b, must falsify S. Since neither n_2 nor any ancestor of n_2 is a failure node, some node below n_2 on b (perhaps the leaf itself) must be the highest node which falsifies a clause in S.

We have shown that given an arbitrary failure node n_1, either its sibling n_2 is a failure node (and hence their parent is an inference node), or there is a failure node at a *greater* depth than n_1 and n_2. Therefore, if there is no inference node, there must be an infinite sequence of failure nodes. But this is impossible, since a semantic tree is finite (its depth is the number of different atomic propositions in S). ■

Lemma 4.38 *Let $\mathscr{T}$ be closed semantic tree and let n be an inference node whose children n_1 and n_2 of n are (by definition) failure nodes with clauses C_1 and C_2 associated with them, respectively. Then C_1, C_2 clash and the partial interpretation defined by the branch from the root to n falsifies their resolvent.*

Proof of the Notation follows Fig. 4.4. Let b_1 and b_2 be the partial branches from the root to the nodes n_1 and n_2, respectively. Since n_1 and n_2 are failure nodes and since C_1 and C_2 are clauses associated with the nodes, they are *not* falsified by any node higher up in the tree. By Lemma 4.34, the clauses C_1 and C_2 are subsets of the complements of the literals labeling the nodes of b_1 and b_2, respectively. Since b_1 and b_2 are identical except for the edges from n to n_1 and n_2, we must have $\bar{l} \in C_1$ and $\bar{l^c} \in C_2$ so that the clauses are falsified by the assignments to the literals.

Since the nodes n_1 and n_2 are failure nodes, $v_{\mathscr{I}_{b_1}}(C_1) = v_{\mathscr{I}_{b_2}}(C_2) = F$. But clauses are disjunctions so $v_{\mathscr{I}_{b_1}}(C_1 - \{\bar{l}\}) = v_{\mathscr{I}_{b_2}}(C_2 - \{\bar{l^c}\}) = F$ and this also holds for the interpretation $\mathscr{I}_b$. Therefore, their resolvent is also falsified:

$$v_{\mathscr{I}_b}(\,(C_1 - \{\bar{l}\}) \cup (C_2 - \{\bar{l^c}\})\,) = F.$$

■

Example 4.39 In Fig. 4.3, $\bar{r}$ and $\bar{p}\bar{q}r$ are clauses associated with failure nodes 3 and 4, respectively. The resolvent $\bar{p}\bar{q}$ is falsified by $\mathscr{I}_{pq}(p) = T$, $\mathscr{I}_{pq}(q) = T$, the partial interpretation associated with the parent node of 3 and 4. The parent node is now a failure node for the set of clauses $S \cup \{\bar{p}\bar{q}\}$. ■

There is a technicality that must be dealt with before we can prove completeness. A semantic tree is defined by choosing an ordering for the set of atoms that appear in *all* the clauses in a set; therefore, an inference node may not be a failure node.

Example 4.40 The semantic tree in Fig. 4.3 is also a semantic tree for the set of clauses $\{p,\ \bar{p}q,\ \bar{r},\ \bar{p}r\}$. Node 3 is a failure node associated with $\bar{r}$ and 4 is a failure node associated with $\bar{p}r$, but their parent is *not* a failure node for their resolvent $\bar{p}$, since it is already falsified by a node higher up in the tree. (Recall that a failure node was defined to be the node *closest* to the root which falsifies the set of clauses.) ■

Lemma 4.41 *Let n be an inference node, $C_1, C_2 \in S$ clauses associated with the failure nodes that are the children of n, and C their resolvent. Then $S \cup \{C\}$ has a failure node that is either n or an ancestor of n and C is a clause associated with the failure node.*

Proof By Lemma 4.38, $v_{\mathscr{I}_b}(C) = F$, where $\mathscr{I}_b$ is the partial interpretation associated with the partial branch b from the root to the inference node. By Lemma 4.34, $C \subseteq \{l_1^c, \ldots, l_n^c\}$, the set of complements of the literals labeling b. Let j be the smallest index such $C \cap \{l_{j+1}^c, \ldots, l_n^c\} = \emptyset$. Then $C \subseteq \{l_1^c, \ldots, l_j^c\} \subseteq \{l_1^c, \ldots, l_n^c\}$ so $v_{\mathscr{I}_b^j}(C) = v_{\mathscr{I}_b}(C) = F$ where $\mathscr{I}_b^j$ is the partial interpretation defined by the partial branch from the root to node j. It follows that j is a failure node and C is a clause associated with it. ■

Example 4.42 Returning to the set of clauses $\{p,\ \bar{p}q,\ \bar{r},\ \bar{p}r\}$ in Example 4.40, the resolvent at the inference node is $C = \{\bar{p}\}$. Now $C = \{\bar{p}\} \subseteq \{\bar{p}, \bar{q}\}$, the complements of the literals on the partial branch from the root to the inference node. Let $j = 1$. Then $\{\bar{p}\} \cap \{\bar{q}\} = \emptyset$, $\{\bar{p}\} \subseteq \{\bar{p}\}$ and $C = \{\bar{p}\}$ is falsified by the partial interpretation $\mathscr{I}_{b_p}(p) = T$. ■

We now have all the machinery needed to proof completeness.

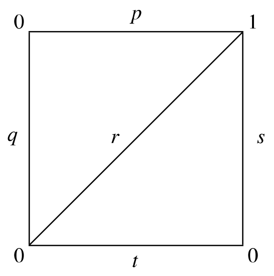
Proof of Completeness of resolution If S is an unsatisfiable set of clauses, there is a closed semantic tree $\mathscr{T}$ for S. If S is unsatisfiable and does not already contain $\Box$, there must be at least two failure nodes in $\mathscr{T}$ (exercise), so by Lemma 4.37, there is at least one inference node in $\mathscr{T}$.

An application of the resolution rule at the inference node adds the resolvent to the set, creating a failure node by Lemma 4.41 and deleting two failure nodes, thus decreasing the number of failure nodes. When the number of failure nodes has decreased to one, it must be the root which is associated with the derivation of the empty clause by the resolution rule. ■

4.5 Hard Examples for Resolution *

If you try the resolution procedure on formulas in clausal form, you will find that is usually quite efficient. However, there are families of formulas on which *any* resolution refutation is necessarily inefficient. We show how an unsatisfiable set of clauses can be associated with an arbitrarily large graph such that a resolution refutation of a set of clauses from this family produces an exponential number of new clauses.

Let G be an undirected graph. Label the nodes with 0 or 1 and the edges with distinct atoms. The following graph will be used as an example throughout this section.



Definition 4.43

- The *parity* of a natural number i is 0 if i is even and 1 if i is odd.
- Let C be a clause. $\Pi(C)$, the *parity of* C, is the parity of the number of complemented literals in C.
- Let $\mathscr{I}$ be an interpretation for a set of atomic propositions $\mathscr{P}$. $\Pi(\mathscr{I})$, the *parity of* $\mathscr{I}$, is the parity of the number of atoms in $\mathscr{P}$ assigned T in $\mathscr{I}$. ■