Resolution for Classical Propositional Logic

Lecture Interactive Proof Tools
Summer Term 2003

Hans de Nivelle, Patrick Maier

Classical Propositional Logic

Syntax: Formulas $\varphi, \psi, \dots \in \Phi_0$ are built up from

- propositional variables $A, B, \dots \in R_0$
- the unary operator $\neg$, and
- the binary operators $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$.

Interpretations, Truth:

- An *interpretation* I maps R_0 to the *truth values* $\{0, 1\}$.
- Given I and φ, the relation $I \models \varphi$ (I *satisfies* φ, φ is *true* in I) is defined inductively over φ as usual.
- φ is *satisfiable* iff there is I such that $I \models \varphi$.
- φ is *valid* iff $I \models \varphi$ for all I.

Classical Propositional Logic – Clauses

Clauses are special formulas:

- A *literal* is variable $A \in R_0$ or its negation $\neg A$.
- *Clauses* $C, D, \dots$ are disjunctions of literals.
- The empty disjunction (called *empty clause*) is denoted by $\bot$.
- A set of clauses Γ is *satisfiable* if there is I such that $I \models C$ for all clauses $C \in \Gamma$.

Note that a set of clauses containing $\bot$ is unsatisfiable.

Classical Propositional Logic – Normal Forms

Transformations (in this order):

1. Expansion of implication and equivalence:

$$\varphi \leftrightarrow \psi = (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi) \qquad \varphi \rightarrow \psi = \neg\varphi \vee \psi$$

2. Conversion to negation normal form (pushing $\neg$ in):

$$\neg\neg\varphi = \varphi \qquad \neg(\varphi \wedge \psi) = \neg\varphi \vee \neg\psi \qquad \neg(\varphi \vee \psi) = \neg\varphi \wedge \neg\psi$$

3. Conversion to conjunctive normal form (pulling $\wedge$ out):

$$(\varphi \wedge \psi) \vee \rho = (\varphi \vee \rho) \wedge (\psi \vee \rho)$$

Identify formulas in conjunctive normal form (CNF) with finite sets of clauses.

Resolution Calculus Res

Inference rules inferring clauses from clauses:

- **Resolution:** For all clauses C, D and variables A

$$\frac{C \vee A \qquad \neg A \vee D}{C \vee D}$$

  Terminology: $C \vee D$ *resolvent*, A *resolved atom*.

- **Factoring:** For all clauses C and variables A

$$\frac{C \vee A \vee A}{C \vee A}$$

  Terminology: $C \vee A$ *factor*.

Note that $\vee$ is associative and commutative, i. e., the 'order' of literals in clauses does not matter.

Resolution Calculus Res – Sample Derivation

Given clause set: $\{\neg A \vee \neg A \vee B,\ A \vee B,\ \neg A' \vee \neg B,\ A'\}$.

Refutation:

1.	$\neg A \vee \neg A \vee B$	(given)
2.	$A \vee B$	(given)
3.	$\neg A' \vee \neg B$	(given)
4.	A'	(given)
5.	$\neg A \vee B \vee B$	(Res. 2 and 1)
6.	$\neg A \vee B$	(Fact. 5)
7.	$B \vee B$	(Res. 2 and 6)
8.	B	(Fact. 7)
9.	$\neg A'$	(Res. 8 and 3)
10.	$\perp$	(Res. 4 and 9)

Resolution Calculus Res – Properties

Reflexive-transitive closure of inference system Res:

$$\begin{aligned}
\mathrm{Res}(\Gamma) &:= \{C \mid C \text{ is resolvent/factor of inference with premises in } \Gamma\} \\
\mathrm{Res}^0(\Gamma) &:= \Gamma \\
\mathrm{Res}^{n+1}(\Gamma) &:= \mathrm{Res}(\mathrm{Res}^n(\Gamma)) \cup \mathrm{Res}^n(\Gamma) \\
\mathrm{Res}^*(\Gamma) &:= \bigcup_{n \geq 0} \mathrm{Res}^n(\Gamma)
\end{aligned}$$

Logical properties:

- **Soundness:** If $I \models \Gamma$ then $I \models \mathrm{Res}^*(\Gamma)$.
  Corollary: If $\perp \in \mathrm{Res}^*(\Gamma)$ then Γ unsatisfiable.
- **Refutational completeness:** If Γ unsatisfiable then $\perp \in \mathrm{Res}^*(\Gamma)$.

Resolution Calculus Res – Theorem Proving

To prove validity of a propositional formula φ:

1. Convert $\neg\varphi$ to CNF, i. e., to a finite set of clauses Γ.
2. *Saturate* Γ under Res (i. e., add conclusions of inferences) to obtain $\Gamma^* = \mathrm{Res}^*(\Gamma)$.
3. If $\perp \in \Gamma^*$ then $\neg\varphi$ is unsatisfiable, i. e., φ is valid.
   If $\perp \notin \Gamma^*$ then $\neg\varphi$ is satisfiable, i. e., φ is invalid.

The above steps form an effective algorithm iff each step terminates:

1. Termination proven in the exercises.
2. Need not terminate,[a] see exercises for an example.
3. Terminates if $\mathrm{Res}^*(\Gamma)$ is finite, i. e., if saturation terminates.

[a] There exist variants of the resolution calculus where saturation terminates.

Resolution as Decision Procedure for Prop. Logic

To make resolution terminate, saturate up to redundancy:

- C is **redundant** in Γ iff $\exists D \in \Gamma : D$ is a strict submultiset of C.
- Saturation up to reduncancy:

$$\mathrm{Res}_{\mathrm{red}}(\Gamma) := \{C \mid C \text{ is irredundant in } \Gamma \wedge \\ C \text{ is resolvent/factor of clauses in } \Gamma\}$$

- Completeness preserved: $\bot \in \mathrm{Res}^*(\Gamma) \Rightarrow \bot \in \mathrm{Res}^*_{\mathrm{red}}(\Gamma)$.
  Proof: Exercise.
  Hint: If C is irredundant in Γ then $\forall n : C \in \mathrm{Res}^n(\Gamma) \Rightarrow C \in \mathrm{Res}^n_{\mathrm{red}}(\Gamma)$.
- Termination: If Γ is finite then $\mathrm{Res}^*_{\mathrm{red}}(\Gamma)$ is finite.
  Proof: Exercise.
  Hint: By contradiction using Dickson's Lemma (k = number of literals in Γ).

Dickson's Lemma

This useful (and somewhat surprising) lemma is a frequent argument in termination proofs.[a]

Dickson's Lemma: Every infinite sequence of vectors $v_1, v_2, v_3, \ldots$ in $\mathbb{N}^k$ has indices $m < n$ such that $v_m \leq v_n$.
(Here, $\langle v_{m1}, \ldots, v_{mk}\rangle = v_m \leq v_n = \langle v_{n1}, \ldots, v_{nk}\rangle$ iff $v_{ml} \leq v_{nl}$ for all $l \leq k$.)

Proof: By induction on k.

- $k = 1$: Follows from totality and well-foundedness of $\leq$ on $\mathbb{N}$.
- $k > 1$: There is an infinite sequence of indices $i_1 < i_2 < i_3 < \ldots$ such that $v_{i_1 k} \leq v_{i_2 k} \leq v_{i_3 k} \leq \ldots$. For every n, let $v'_n \in \mathbb{N}^{k-1}$ be the projection of v_n which forgets the k-th component. By induction hypothesis there are $m < n$ such that $v'_{i_m} \leq v'_{i_n}$. Hence $v_{i_m} \leq v_{i_n}$.

[a] Generalizations of Dickson's Lemma to sequences of words resp. trees are known as Higman's Lemma and Kruskal's Tree Embedding Theorem, respectively.