15-414: Bug Catching: Automated Program Verification

Lecture Notes on Resolution

Frank Pfenning

Carnegie Mellon University

Lecture 13
Tuesday, March 1, 2022

1 Introduction

In the last lecture we saw the sequent calculus, which has many positive attributes. Sequents are an excellent basis for communicating proof goals. With some refinement, it can serve as a foundation for goal-directed proof search procedures. It is also quite robust, which means that many logics (we have seen dynamic logic, parts of modal logic, classical logic, and a glimpse of intuitionistic logic) have formulations in the sequent calculus. On the other hand, sequent proofs are quite verbose and also lose some of their intuitive appeal if they have multiple succedents.

Many modern provers, however, are not directly based on the sequent calculus. Broadly, we can classify those as either being based on *resolution* in some form or on *co-operating decision procedures*, usually in the form of SMT (Satisfiability Modulo Theories) based on SAT (Satisfiability). In today's lecture we discuss the rudiments of resolution, which also represents a bridge between human-oriented and machine-oriented proof systems. It has the virtue that a resolution proof is quite easy to check, which is not the case for proofs is carried out by a SAT or SMT solver. Therefore, SAT solvers these days produce an independently checkable proof certificate, and SMT solvers should (although implementations are not quite as far advanced). One popular format for such a certificate is in fact a resolution proof.

In this lecture we present only the propositional case (no quantification) and only *binary resolution*. There are many refinements and optimizations of resolution which was one of the dominant methods of automated theorem proving for several decades. Why3 supports many provers, including provers such as Eprover, Spass, and Vampire that have their origins in variants of resolution but continue to evolve.

Learning goals. After this lecture, you should be able to:

- Carry out resolution proofs for propositional logic
- Construct a satisfying assignment from a consistent, saturated theory
- Convert formulas to conjunctive normal form by creating internal names

2 Satisfiability

For today's lecture we restrict ourselves to propositional formulas. We use lowercase p, q for *atoms* which you can also think of as propositional variables. Then our grammar for propositions is:

$$Q, P \quad ::= \quad p \mid P \wedge Q \mid P \vee Q \mid P \rightarrow Q \mid \neg P \mid \top \mid \bot$$

A truth assignment M assigns either true or false to every propositional variable (or atom, as we say). This is analogous to the *state* ω in dynamic logic that assigns an integer to every variable. So, if you like, you can think of propositional theorem proving as deciding the (quantifier-free) theory of Booleans. We write $M \models P$ if the formula P is true given the assignment M. This is defined exactly as we did in dynamic logic on these connectives with one additional clause:

$$M \models p \quad \text{iff} \quad M(p) = \mathsf{true}$$

A convenient way to present a truth assignment is by giving a list of p if $M(p) = \mathsf{true}$ and $\neg p$ if $M(p) = \mathsf{false}$. In principle, this list would have to be infinite, but since every formula contains only finitely many atoms we can use such a finite representation.

The fundamental question is once again that of validity, that is:

> ***Validity**: decide whether $M \models P$ for* every *truth assignment M.*

For reasons that are not entirely clear to me, this is usually turned into a problem of *satisfiability*:

> ***Satisfiability**: decide whether $M \models P$ for* some *truth assignment M.*

These problems are equivalent in the sense that

> *P is valid if and only if $\neg P$ is unsatisfiable*

which you can easily verify from the definitions. So instead of proving P we try to refute $\neg P$ by searching for an assignment M such that $M \models \neg P$. This is also called a *model* for $\neg P$. If no such model exists, then $\neg P$ is *unsatisfiable* which means that P is valid. If such a model exists that $\neg P$ is *satisfiable*, which means that P is *not valid*. In this case the model M for $\neg P$ provides a counterexample to the validity of P in the sense that $M \not\models P$. Read the previous paragraph at least once more. It's crucial in understanding what follows.

3 Clausal Form

There is some redundancy among the connectives of (classical) propositional logic. Theorem proving methods are easier to think about and implement if we can put the formulas into a normal form such that P is satisfiable if and only if its normal form Q is satisfiable. Moreover, we'd like to be able to take any satisfying assignment M for Q and translate it to a satisfying assignment for P.

One such normal form is called *conjunctive normal form* (CNF). A formula is in conjunctive normal form if it is a conjunction of disjunction of atoms and negated atoms. That is:

$$\begin{array}{llll} \text{Literal} & L & ::= & p \mid \neg p \\ \text{Clause} & D, C & ::= & \bot \mid L \mid C \vee D \\ \text{Theory} & S, T & ::= & \top \mid C \mid S \wedge T \end{array}$$

For the algorithms, it is convenient to think of a clause as a *set of literals* $\{L_1, \ldots, L_n\}$ where we write $\bot$ for the empty set. We will still write this as $L_1 \vee \ldots \vee L_n$. Similarly, we think of a theory as a *sequence of clauses*, $C_0, C_1, \ldots, C_k$.

There are a number of methods to put a propositional formula into an equisatisfiable clausal form. We'll see a good one in Section 7. A not particularly efficient one would first push in negations and then use the laws of distributivity, but this could blow up the size of the formula exponentially.

In order to satisfy a formula in conjunctive normal form we have to satisfy each conjunct. On the other hand, it is sufficient to satisfy a single literal in each clause. If a clause is empty (representing $\bot$) then it can not be satisfied by any assignment. Consequently, a theory containing the empty clause is unsatisfiable.

As a simple running example, consider

$$P = (p \rightarrow (q \rightarrow r)) \rightarrow ((p \wedge q) \rightarrow r)$$

with atoms p, q, and r, where we have written some redundant parentheses for clarity. As a first step, we'll negate this, repeatedly using the laws $(P \rightarrow Q) \leftrightarrow (\neg P \vee Q)$ and $\neg(P \rightarrow Q) \leftrightarrow (P \wedge \neg Q)$.

$$\begin{array}{rcl} \neg P & = & \neg((p \rightarrow (q \rightarrow r)) \rightarrow ((p \wedge q) \rightarrow r)) \\ & \leftrightarrow & (p \rightarrow (q \rightarrow r)) \wedge \neg((p \wedge q) \rightarrow r) \\ & \leftrightarrow & (\neg p \vee (\neg q \vee r)) \wedge (p \wedge q) \wedge \neg r \\ & \leftrightarrow & (\neg p \vee \neg q \vee r) \wedge p \wedge q \wedge \neg r \end{array}$$

At this point we have reached a conjunctive normal form. Written as a theory, labeling each clause:

$$\begin{array}{ll} \neg p \vee \neg q \vee r & C_0 \\ p & C_1 \\ q & C_2 \\ \neg r & C_3 \end{array}$$

It is easy to see that this theory is *unsatisfiable*. Since it should be read as a conjunction, all of C_1, C_2, C_3 must be true, forcing that for any model, p and q would have to be true, and r false. But if p and q are true and r false, then C_0 will be false, preventing us from simultaneously satisfying all clauses.

4 Binary Resolution

Resolution is both the name of a rule of inference and a (nondeterministic) algorithm searching for a *refutation* of a theory T. Such a *refutation* is evidence that the theory is unsatisfiable. In brief, we add more and more consequences to a theory in the hope of reaching the empty clause. If we do, we conclude that the original theory was in fact *unsatisfiable* because our notion of consequence preserves satisfiability. The other outcome is that we reach *saturation*, that is, any further inference would only add clauses already in the theory. In that case, it will turn out, the theory is *satisfiable*.

The single rule of inference we use can be written as follows:

$$\frac{p \vee C \quad \neg p \vee D}{C \vee D}\ \mathsf{resolution}$$

The two premises of the rule are clauses (really: sets of literals), even if we write them using disjunction, so C has no copy of p and D has no copy of $\neg p$.

Thinking of the theory as a sequence of clauses $C_0, \ldots, C_{k-1}$, the rule looks like this:

$$\begin{array}{ll} p \vee C & C_i \quad (i < k) \\ \neg p \vee D & C_j \quad (j < k) \\ \hline C \vee D & C_k = C_i \bowtie_p C_j \end{array}$$

Here we have invented a notation for the justification of the new clause C_k as the result of *resolving* C_i with C_j.

It is easy to see that this rule is *sound* in the sense that it preserves the set of models. To see that, consider an assignment M such that $M \models p \vee C$ and $M \models \neg p \vee D$. We consider two cases:

1. $M(p) = \mathsf{true}$. Then $M \models D$ since $M \models \neg p \vee D$. Therefore $M \models C \vee D$.

2. $M(p) = \mathsf{false}$. Then $M \models C$ since $M \models p \vee C$. Therefore $M \models C \vee D$.

Either way, M is a model of $C \vee D$. Since we just extend the theory, any model of the extended theory will automatically be also a model of the original theory.

If we can obtain the empty clause $\perp$ by repeated application of this rule, we know the original theory cannot be satisfiable. That's because any all models are preserved, and the empty clause has no models.

Let's apply resolution in our example. We draw a line here between our original sequence of clauses an the further clauses inferred by resolution. You should check you understand the justification of each new clause.

$$\begin{array}{ll} \neg p \vee \neg q \vee r & C_0 \\ p & C_1 \\ q & C_2 \\ \neg r & C_3 \\ \hline \neg q \vee r & C_4 = C_1 \bowtie_p C_0 \\ r & C_5 = C_2 \bowtie_q C_4 \\ \bot & C_6 = C_5 \bowtie_r C_3 \end{array}$$

Since we have derived the empty clause we know the original theory is unsatisfiable. This is turn means that the original formula we started with $(p \rightarrow (q \rightarrow r)) \rightarrow (p \wedge q \rightarrow r)$ is valid.

By the way, one reason to think of the theory as a sequence instead of a tree is that we can reuse intermediate clauses in multiple future inferences. This is not used in this example, but it is a frequent occurrence in realistic examples and can save an exponential amount of space and time.

5 Saturation

When we are not able to reach a contradiction, then by necessity the sequence of clauses must reach *saturation*, that is, any further application of resolution will only lead to clauses already in the sequence. We must reach such a state because if we start with a finite set of clauses they contain only finitely many literals (say n) from which we can form at mot 2^n distinct clauses.

As an example of saturation, we try to prove

$$(p \rightarrow (q \rightarrow r)) \rightarrow (p \rightarrow r)$$

which is *not valid*. Negating as before, we get the CNF

$$(\neg p \vee \neg q \vee r) \wedge p \wedge \neg r$$

Turning this into a sequence of clauses we get the same as before, except the previous clause C_2 is no longer available.

$$\begin{array}{ll} \neg p \vee \neg q \vee r & C_0 \\ p & C_1 \\ \neg r & C_2 \\ \hline \neg q \vee r & C_3 = C_1 \bowtie_p C_0 \\ \neg q & C_4 = C_3 \bowtie_r C_2 \\ \neg p \vee \neg q & C_5 = C_0 \bowtie_r C_2 \end{array}$$

Besides inferences we have already made, we could also try $C_1 \bowtie_p C_5 = \neg q$ but this is equal to C_4. In other words, we have reach saturation without deducing a contradiction and we conclude the initial theory is satisfiable.

But what is the satisfying assignment? Robinson's original paper [Rob65] proves the *completeness* of resolution by constructing a model from a saturated theory. Together with the soundness proof of the last section this means resolution is a sound and complete algorithm for determining satisfiability.

We refer you to the original paper for its correctness, but we show the algorithm. We assume we have an enumeration

$$p_0, p_1, \ldots, p_{n-1}$$

of all the atoms in the saturated theory S. We build up an assignment M by considering each p_i in turn, deciding whether we should set $M(p_i) = \mathsf{true}$ or $M(p_i) = \mathsf{false}$. We use the representation of such a partial assignment as a set of literals containing p_i in the former case and $\neg p_i$ in the latter case.

We use the notation $\overline{M}$ to denote the negation of all the literals in M. One property we need is that if $C \subset \overline{M}$ then $M \not\models C$ because M falsifies all literals in C. Furthermore, no extension of M could possibly satisfy C because the truth value for all literals in C has already been decided.

The basic strategy is to assign truth to an atom unless is is absolutely necessary to assign it false. This would be the case at stage i if there is a clause C in the saturated theory such that $C \subseteq \overline{M \cup \{p\}}$. So in that case we assign p the value false, represented by adding $\neg p$ to the partial model.

Writing out this process more formally:

$$\begin{array}{lcll}
M_0 & = & \{\,\} & \\
M_{i+1} & = & M_i \cup \{p_i\} & \text{provided there is no } C \in S \text{ s.t. } C \subseteq \overline{M_i \cup \{p_i\}} \\
M_{i+1} & = & M_i \cup \{\neg p_i\} & \text{provided there is a } C \in S \text{ s.t. } C \subseteq \overline{M_i \cup \{p\}} \\
 & & & \\
M & = & M_n &
\end{array}$$

Then $M \models S$, as proved by Robinson.

Lets apply this algorithm to our saturated theory and the order

$$p, q, r$$

so $n = 3$, $p_0 = p$, $p_1 = q$, and $p_2 = r$. For our saturated theory

$$\begin{array}{ll}
\neg p \vee \neg q \vee r & C_0 \\
p & C_1 \\
\neg r & C_2 \\
\hline
\neg q \vee r & C_4 = C_1 \bowtie_p C_0 \\
\neg q & C_5 = C_4 \bowtie_r C_2 \\
\neg p \vee \neg q & C_6 = C_0 \bowtie_r C_2
\end{array}$$

we obtain the following sequence:

$$\begin{array}{lcll}
M_0 & = & \{\,\} & \\
M_1 & = & \{p\} & \text{since there is no } C \subseteq \overline{\{p\}} = \{\neg p\} \\
M_2 & = & \{p, \neg q\} & \text{since } C_5 \subseteq \overline{\{p, q\}} = \{\neg p, \neg q\} \\
M_3 & = & \{p, \neg q, \neg r\} & \text{since } C_2 \subseteq \overline{\{p, \neg q, r\}} = \{\neg p, q, \neg r\}
\end{array}$$

In this example we can now easily verify that $M_3 \models S$. Since it is a model every clause in the saturated set, it is of course also a model for the original set of clauses and therefore a counterexample to satisfiability.

6 Checking Certificates

If we are given a resolution refutation (that is, one deducing $\bot$), it is easy to check that all applications of the resolution rule are correct as claimed. We wouldn't even need the intermediate clauses because we can always reconstruct them by calculating $C_i \bowtie_p C_j$. One can also prune some intermediate clauses by working backwards from the contradiction and keeping only those clauses involved in its generation.

If we are given a saturated theory *without* an empty clause, we could easily check that all rule applications are correct and that any further rule applications will not lead to new consequences.

An alternative would be for the prover to apply Robinson's algorithm to construct a satisfying assignment and use that as a small certificate of satisfiability. A checker can just take the original theory T and quickly verify that there is a true literal in each clause. With such a certificate we no longer need the saturated theory at all.

7 The Inverse Method

Even though what we done so far seems closely tied to classical logic in many respects, such as transforming formulas into conjunctive normal form, resolution actually is quite robust and can be used for other logics with suitable extensions and modifications. Briefly, we can derive a resolution calculus from a sequent calculus for a logic.

In fact, the inverse method was developed by Maslov [Mas64] around the same as resolution by Robinson [Rob65] and it provides and alternative path to understanding resolution. Maslov's original calculus is related to a resolution strategy called *hyperresolution*, here we show a less efficient form that relates to binary resolution instead.

To start with, we reformulate the classical sequent calculus from Lecture 12 to think of the inferences going in the downward direction, instead of the upward direction. For our example, we just need implication, conjunction, and atoms, so we only show the relevant rules. One change is that we combine the antecedents and succedents from the premises in the conclusion. The other change is that we break down the single left rule for conjunction into two, and similarly for the single right rule for implication. In the presence of contraction, this is equivalent to the rules before. We also restrict the identity to atoms, which is not necessary but makes reasoning simpler because we don't have to look for identical complex formulas on both sides of the turnstile.

$$\frac{}{p \vdash p}\,\mathsf{id} \qquad \frac{\Gamma, P, P \vdash \Delta}{\Gamma, P \vdash \Delta}\,\mathsf{CL} \quad \frac{\Gamma \vdash Q, Q, \Delta}{\Gamma \vdash Q, \Delta}\,\mathsf{CR}$$

$$\frac{\Gamma_1 \vdash P, \Delta_1 \quad \Gamma_2 \vdash Q, \Delta_2}{\Gamma_1, \Gamma_2 \vdash P \wedge Q, \Delta_1, \Delta_2}\,\wedge R \qquad \frac{\Gamma, P \vdash \Delta}{\Gamma, P \wedge Q \vdash \Delta}\,\wedge R_1 \quad \frac{\Gamma, Q \vdash \Delta}{\Gamma, P \wedge Q \vdash \Delta}\,\wedge R_2$$

$$\frac{\Gamma, P \vdash \Delta}{\Gamma \vdash P \to Q, \Delta}\,{\to}R_1 \quad \frac{\Gamma \vdash Q, \Delta}{\Gamma \vdash P \to Q, \Delta}\,{\to}R_2 \qquad \frac{\Gamma_1 \vdash P, \Delta_1 \quad \Gamma_2, Q \vdash \Delta_2}{\Gamma_1, \Gamma_2, P \to Q \vdash \Delta_1, \Delta_2}\,{\to}L$$

In this calculus the rules are no longer invertible, but this is no longer a relevant property because we intend to use these rules starting from the premises.

Let's write the proof of our example formula first with the bottom-up sequent calculus, aggressively eliding antecedents and succedents we no longer need.

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{p \vdash p}\,\mathsf{id} \quad \cfrac{\cfrac{}{q \vdash q}\,\mathsf{id} \quad \cfrac{}{r \vdash r}\,\mathsf{id}}{q \to r, q \vdash r}\,{\to}L}{p \to (q \to r), p, q \vdash r}\,{\to}L}{p \to (q \to r), p \wedge q \vdash r}\,\wedge L}{p \to (q \to r) \vdash (p \wedge q) \to r}\,{\to}R}{\cdot \vdash (p \to (q \to r)) \to ((p \wedge q) \to r)}\,{\to}R$$

To obtain the proof in the forward sequent calculus we only have to expand the $\wedge L$ and ${\to}R$ rules, which are *derived rules* as follows:

$$\cfrac{\cfrac{\cfrac{\Gamma, P, Q \vdash \Delta}{\Gamma, P, P \wedge Q \vdash \Delta}\,\wedge L_2}{\Gamma, P \wedge Q, P \wedge Q \vdash \Delta}\,\wedge L_1}{\Gamma, P \wedge Q \vdash \Delta}\,\mathsf{CL} \qquad \cfrac{\cfrac{\cfrac{\Gamma, P \vdash Q, \Delta}{\Gamma, P \vdash P \to Q, \Delta}\,{\to}R_2}{\Gamma \vdash P \to Q, P \to Q, \Delta}\,{\to}R_1}{\Gamma \vdash P \to Q, \Delta}\,\mathsf{CR}$$

Expanding these rules gives us a proof of our formula in the forward sequent calculus. But how could we possibly have found that? If we start from identities $p \vdash p$, $q \vdash q$ and $r \vdash r$, the space of possible sequents we could infer is vast!

Maslov's idea is that in a backwards proof (in our original sequent calculus) only *subformulas of the endsequent* will occur. We can exploit this in two steps:

1. We give new names for each subformula in our goal formula.
2. We specialize the inference rules to these subformulas.

For each of the newly introduced names, we write L^+ if it may occur on the right-hand side of a sequent, and L^- if it may occur on the left-hand side of a sequent. This is actually a bit stricter than simply the subformula property, since we also track its possible place in a sequent. Recall

$$(p \to (q \to r)) \to ((p \wedge q) \to r)$$

Then we introduce a unique new name for each subformula:

$$\begin{array}{rcl} L_0^+ & = & L_1^- \to L_2^+ \\ L_1^- & = & p^+ \to L_3^- \\ L_2^+ & = & L_4^- \to r^+ \\ L_3^- & = & q^+ \to r^- \\ L_4^- & = & p^- \wedge q^- \end{array}$$

Now, for example, the two specialized left rules for $L_4^- = p^- \wedge q^-$ would be

$$\frac{\Gamma, p^- \vdash \Delta}{\Gamma, L_4^- \vdash \Delta}\, 4L_1 \qquad \frac{\Gamma, q^- \vdash \Delta}{\Gamma, L_4^- \vdash \Delta}\, 4L_2$$

Because the antecedents Γ and Δ are always propagated from the premises and combined in the conclusion, we omit them in the rules to write the more compactly. We obtain the following rules:

$$\frac{L_1^- \vdash}{\vdash L_0^+}\, 0R_1 \quad \frac{\vdash L_2^+}{\vdash L_0^+}\, 0R_2 \quad \frac{\vdash p^+ \quad L_3^- \vdash}{L_1^- \vdash}\, 1L \quad \frac{L_4^- \vdash}{\vdash L_2^+}\, 2R_1 \quad \frac{\vdash r^+}{\vdash L_2^+}\, 2R_2$$

$$\frac{\vdash q^+ \quad r^- \vdash}{L_3^- \vdash}\, 3L \quad \frac{p^- \vdash}{L_4^- \vdash}\, 4L_1 \quad \frac{q^- \vdash}{L_4^- \vdash}\, 4L_2$$

What does this have to do with resolution? Since all sequents just consist of the original atoms or those introduced by naming, we can think of a sequent in the following way:

$$p_1, \ldots, p_n \vdash q_1, \ldots, q_m \qquad \sim \qquad \neg p_1 \vee \ldots \vee \neg p_n \vee q_1 \vee \ldots \vee q_m$$

This comes from the interpretation of $\Gamma \vdash \Delta$ as $\bigwedge\Gamma \to \bigvee\Delta$. Let's looks again at

$$\frac{\Gamma, p^- \vdash \Delta}{\Gamma, L_4^- \vdash \Delta}\, 4L_1 \qquad \sim \qquad \frac{\neg p^-, \neg\Gamma, \Delta}{\neg L_4^-, \neg\Gamma, \Delta}\, 4L_1 \qquad \sim \qquad \frac{\neg p^-, C}{\neg L_4^-, C}\, 4L_1$$

where the last transformation just lumps together positive and negative literals as we do with clauses.

The question is how to achieve the inference from premise to conclusion using resolution? We just need a clause

$$p^-, \neg L_4^-$$

because the inference

$$\frac{\neg p^-, C \quad p^-, \neg L_4^-}{\neg L_4^-, C}$$

is an instance of resolution! In this way, we can express each of the single-premise inference rules

$$\frac{L_1^- \vdash}{\vdash L_0^+}\ 0R_1 \quad \frac{\vdash L_2^+}{\vdash L_0^+}\ 0R_2 \quad \frac{L_4^- \vdash}{\vdash L_2^+}\ 2R_1 \quad \frac{\vdash r^+}{\vdash L_2^+}\ 2R_2 \quad \frac{p^- \vdash}{L_4^- \vdash}\ 4L_1 \quad \frac{q^- \vdash}{L_4^- \vdash}\ 4L_2$$

as a clause, which becomes the following theory:

$$\begin{array}{ll} L_1, L_0 & (0R_1) \\ \neg L_2, L_0 & (0R_2) \\ L_4, L_2 & (2R_1) \\ \neg r, L_2 & (2R_2) \\ p, \neg L_4 & (4L_1) \\ q, \neg L_4 & (4L_2) \end{array}$$

The two premise rules have to proceed in two steps:

$$\frac{\vdash p^+ \quad L_3^- \vdash}{L_1^- \vdash}\ 1L \qquad \frac{\vdash q^+ \quad r^- \vdash}{L_3^- \vdash}\ 3L$$

become

$$\begin{array}{ll} \neg p, L_3, \neg L_1 & (1L) \\ \neg q, r, \neg L_3 & (3L) \end{array}$$

For example, the inference on the left is mimicked by the two steps of binary resolution on the right.

$$\frac{\Gamma_1 \vdash p, \Delta_1 \quad \Gamma_2, L_3 \vdash \Delta_2}{\Gamma_1, \Gamma_2, L_1 \vdash \Delta_1, \Delta_2}\ 1L \qquad \sim \qquad \frac{\dfrac{p, C \quad \neg p, L_3, \neg L_1}{L_3, \neg L_1, C} \qquad \neg L_3, D}{\neg L_1, C, D}$$

In order to actually carry out resolution, we need one final clause which expresses that if we derive L_0 (our original goal) then we succeed. But in resolution, success is modeled as deriving a contradiction, so we have the final clause

$$\neg L_0 \quad (S)$$

Now any resolution step in the resulting theory corresponds to a logical inference in the forward sequent calculus, where the two-premise rules are done in a "curried" form requiring two steps. Essentially, as we saw in the example above, the first step generates a (valid) derived rule of inference.

To actually carry out the resolution proof we use something called *unit resolution*, that is, if we have a clause with a single literal L we eagerly resolve it with all clauses that contain its negation $\neg L$.

L_1, L_0	$C_0 = (0R_1)$
$\neg L_2, L_0$	$C_1 = (0R_2)$
L_4, L_2	$C_2 = (2R_1)$
$\neg r, L_2$	$C_3 = (2R_2)$
$p, \neg L_4$	$C_4 = (4L_1)$
$q, \neg L_4$	$C_5 = (4L_2)$
$\neg p, L_3, \neg L_1$	$C_6 = (1L)$
$\neg q, r, \neg L_3$	$C_7 = (3L)$
$\neg L_0$	$C_8 = (S)$
L_1	$C_9 = C_0 \bowtie_{L_0} C_8$
$\neg L_2$	$C_{10} = C_1 \bowtie_{L_0} C_8$
$\neg p, L_3$	$C_{11} = C_0 \bowtie_{L_1} C_6$
L_4	$C_{12} = C_2 \bowtie_{L_2} C_{10}$
$\neg r$	$C_{13} = C_3 \bowtie_{L_2} C_{10}$
p	$C_{14} = C_{12} \bowtie_{L_4} C_4$
q	$C_{15} = C_{12} \bowtie_{L_4} C_5$
L_3	$C_{16} = C_{14} \bowtie_p C_{11}$
$r, \neg L_3$	$C_{17} = C_{15} \bowtie_q C_7$
$\neg L_3$	$C_{18} = C_{17} \bowtie_r C_{13}$
$\perp$	$C_{19} = C_{16} \bowtie_{L_3} C_{18}$

Even though we can read off the a sequent proof from this resolution refutation, the order in which the rules are applied do not necessarily model a pure bottom-up or top-down construction. One can impose strategy to achieve one or the other direction, if that is desired. Numerous other strategic refinements are possible. We can also build an efficient prover for intuitionistic logic in this manner (see [MP09]).

We do not prove the correctness of this approach to resolution here, but we hope we have at least made it plausible by exhibiting the connections to the sequent calculus.

References

[Mas64] S. Maslov. The inverse method of establishing deducibility in the classical predicate calculus. *Soviet Mathematical Doklady*, 5:1420–1424, 1964.

[MP09] Sean McLaughlin and Frank Pfenning. Efficient intuitionistic theorem proving with the polarized inverse method. In R.A.Schmidt, editor, *Proceedings of the 22nd International Conference on Automated Deduction (CADE-22))*, pages 230–244, Montreal, Canada, August 2009. Springer LNCS 5663.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.