CSC2512
Types Resolution Proofs and their Complexity

Fahiem Bacchus
Department of Computer Science
University of Toronto

1 Types of Resolution Proofs

General resolution is the proof system we discussed before.

A general resolution proof from a CNF formula $F$ is a sequence of clauses $c_1, c_2, \ldots, c_m$ where for each $c_i$ in this sequence we either have

1. $c_i \in F$ ($c_i$ in an input clause) or
2. $c_i = R(c_j, c_k)$ with $j, k < i$ (the result of resolving two prior clauses in the sequence).

From this sequence we can draw a DAG. The input clauses are the source nodes in this DAG (no incoming arcs, and each clause that arises from a resolution step (each resolvant) has two incoming arcs, one from each clause used in the resolution step. The empty clause is a sink node.

Often in the latter case the incoming arcs are labeled by the literals $x$ and $\neg x$ that were resolved away.

**Negative Resolution** A resolution step $R[(A, x), (B, \neg x)]$ is **negative** whenever $B$ contains only negative literals. Negative resolution requires all resolution steps to be negative.

**Semantic Resolution** Given a truth assignment $\pi$ for the input CNF $F$, a $\pi$-resolution of $F$ is a resolution in which for every resolution step $R[(A, x), (B, \neg x)]$ one of $(A, x)$ or $(B, \neg x)$ must be **falsified** by $\pi$ (note it is impossible for both clauses to be falsified by $\pi$). A refutation is called **semantic** if it is a $\pi$-resolution for some truth assignment $\pi$. 

1
**Linear Resolution**  The refutation has a linear underlying DAG. That is, the proof \( c_1, c_2, \ldots, c_m = () \) has the property that when \( c_i \) is a resolvant it is the result of resolving \( c_{i-1} \) and some prior \( c_j \) (\( j < i \)). (We always use the prior clause in ever resolution step.

**Regular Resolution**  In the DAG each path from the empty clause to an input clause has the property that no arc variable label is repeated.

**Ordered Resolution**  In the DAG the sequence of variable labels along any path from the empty clause to an input clause respects some total ordering of the variables.

**Tree Like Resolution**  The DAG is a tree except that input clauses can participate in multiple resolution steps. Clauses produced by resolution can only be used once.

## 2 Complexity of These Refinements

If we restrict resolution to only consider proofs with these properties we end up with a resolution refinement proof. Each of these refinements is by itself a sound and (refutation) complete proof system.

In the paper “The Complexity of Resolution Refinements” by Joshua Buresh-Oppenheim and Toniann Pitassi, Journal of Symbolic Logic, volume 72, number 4, pages 1336–1352, 2007. The complexity of these refinements was investigated in terms of which system can p-simulate which other system.
<table>
<thead>
<tr>
<th></th>
<th>Neg</th>
<th>Sem</th>
<th>Lin</th>
<th>Order</th>
<th>Reg</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neg</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sem</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Lin</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Order</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Reg</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Tree</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 1: P-Simulation results from Oppenheim and Pitassi