2 Resolution Calculus

We start with a general discussion about calculi and their relation to the NP vs. co-NP problem. Afterwards we specialize to the resolution calculus and analyze its particular properties. Recall that *SAT* is NP-complete, and therefore, $\overline{SAT}$ as well as *TAUT* (the set of propositional tautologies) are co-NP-complete (see appendix).

2.1 Calculi and NP versus co-NP

A *calculus* is understood as a collection of transformations which manipulate formulas or clause sets. A well-known traditional calculus consists of the usual collection of rules for producing equivalent formulas, like the associative laws, the commutative laws, the distributive laws, the deMorgan laws, etc. If it is possible to transform formula F, using the calculus $\mathcal{C}$, into the formula G, we denote this by $F \vdash_{\mathcal{C}} G$.

In general, given some formula F, there are several possibilities to apply a rule of the underlying calculus to F. Therefore, a calculus can be considered as a *nondeterministic* process (or algorithm). Given an input formula F, the nondeterministic algorithm branches, and each computation branch might yield a different output formula G (such that $F \vdash_{\mathcal{C}} G$).

We say that a calculus $\mathcal{C}$ is *correct* if $F \vdash_{\mathcal{C}} G$ implies that $F \rightarrow G$ is a tautology.

A calculus $\mathcal{C}$ is *complete* if for all F, G such that $F \rightarrow G$ is a tautology, it follows that $F \vdash_{\mathcal{C}} G$. For a complete calculus $\mathcal{C}$, this implies that the property that F is a tautology can be proved by showing that $1 \vdash_{\mathcal{C}} F$. The property of F being unsatisfiable is equivalent to the property $F \vdash_{\mathcal{C}} 0$.

A sequence of formulas $F = F_0, F_1, F_2, \ldots, F_t = G$ such that $F_i \vdash_{\mathcal{C}} F_{i+1}$ (for $i = 0, 1, \ldots, t-1$) is a single application of a calculus rule, is called a *proof* (of the fact that $F \rightarrow G$ is a tautology) within the calculus $\mathcal{C}$. Furthermore, t is the *length* of this proof.

In the following we will only consider the *resolution calculus* $\mathcal{R}$. This calculus $\mathcal{R}$ is correct, but it is not complete in the sense defined above. However, the resolution calculus is *refutation complete* as we will show below. This means that for every unsatisfiable formula F we have $F \vdash_{\mathcal{R}} 0$. Actually, compared with completeness, this is not a very strong restriction. Suppose, the goal is to show that $F \rightarrow G$ is a tautology. Then instead, one can show that $F \wedge \neg G$ is unsatisfiable which is equivalent to $(F \wedge \neg G) \vdash_{\mathcal{R}} 0$.

Additionally, notice that the resolution calculus $\mathcal{R}$ can only be applied to formulas in *CNF*, that is, to clause sets. Instead of 0 (for the generic unsatisfiable formula), one obtains the empty clause $\square$, therefore we write $F \vdash_{\mathcal{R}} \square$.

It is an interesting observation that there is a certain way of certification for both, the satisfiability as well as the unsatisfiability of a formula F. Satisfiability can be certified by a satisfying assignment. This can be done by a nondeterministic, polynomial-time algorithm which "guesses" such an assignment. This is the bottom line when showing that $SAT \in$ NP (see appendix about P and NP).

On the other hand, a proof $F \vdash_{\mathcal{C}} 0$ is a certificate for the fact that F is unsatisfiable. Also, such a proof can be found by a nondeterministic algorithm. Put in succinct mathematical terms:

$$F \in SAT \quad \textit{iff} \quad \exists \alpha \,:\, F\alpha = 1$$
$$F \in \overline{SAT} \quad \textit{iff} \quad \exists F_1, F_2, \ldots F_t \,:\, F \vdash_{\mathcal{C}} F_1 \vdash_{\mathcal{C}} \cdots \vdash_{\mathcal{C}} F_t = 0$$

The question remains how many proof steps are necessary to prove that a formula F (which is unsatisfiable, or which is a tautology) is indeed unsatisfiable (resp. a tautology). It may be expected that in the worst case (i.e. for some formulas of a given size n) this requires exponentially in n many proof steps. Indeed, for the resolution calculus we will show this in a moment. The expectation is justified by the following theorem.

Theorem

If there is a correct and refutation-complete calculus $\mathcal{C}$, as well as a polynomial p, such that for every unsatisfiable formula (or clause set) F it is possible to show $F \vdash_{\mathcal{C}} 0$ by a proof of length at most $p(n)$ (where n is the length, or the number of variables, of F) then it follows NP = co-NP.

Proof: The set of unsatisfiable formulas, *UNSAT*, as well as the set of tautologies. *TAUT*, is co-NP-complete. Therefore, it follows NP = co-NP, if we could show *UNSAT* $\in$ NP (resp. *TAUT* $\in$ NP). But this is what the assumption of the theorem claims: there is a nondeterministic process (given by the calculus $\mathcal{C}$) which, given an unsatisfiable formula of size n permits a proof of unsatisfiability which has length at most $p(n)$. A nondeterministic Turing machine can "guess" such a proof and verify that all steps are correct, and if so, accept. $\square$

This connection between the question of whether NP = co-NP and the lengths of proofs for unsatisfiability (or the property of being a tautology) was first observed in (Cook, Reckhow, 1979).

2.2 Refutation Completeness

In this section we introduce the resolution calculus in detail, and we prove its refutation completeness.

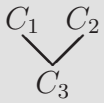
Definition

Two clauses C_1 and C_2 are said to be *resolvable* if there exists a literal u such that $u \in C_1$ and $\overline{u} \in C_2$. In this case a third clause C_3 can be defined, called the *resolvent* of C_1 and C_2, by

$$C_3 \;=\; (C_1 \setminus \{u\}) \cup (C_2 \setminus \{\overline{u}\})$$

If u or $\overline{u}$ is the variable x we say that C_3 was derived from C_1 and C_2 by *resolving on* x. Also, C_1 and C_2 are called the *parent clauses* of C_3.

The following symbolic notation is used to express the situation as described in this definition.



When resolving C_3 from C_1 and C_2, the following semantic property can be easily verified. If α is an assignment which satisfies both C_1 and C_2, that is $(C_1 \wedge C_2)\,\alpha \;=\; 1$, then α satisfies C_3 too, i.e. $C_3\,\alpha \;=\; 1$. This simple observation is the key for the correctness of the resolution calculus (see below).

Producing a resolvent is just a single step in a complete resolution proof, which we define next.

Definition

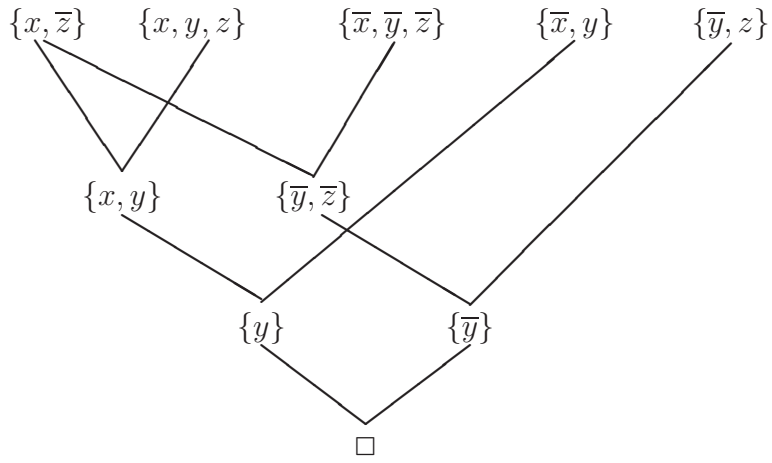
Let F be a clause set (a formula in *CNF*). A *resolution proof* (or *resolution refutation*) is a sequence of clauses $\mathcal{R} \;=\; (C_1, C_2, \ldots, C_t)$ such that the last clause is the empty clause: $C_t = \Box$. Furthermore, for $i = 1, 2, \ldots, t$ either C_i is one of the clauses from F, or C_i is a resolvent of two clauses C_j and C_l which appeared earlier in the sequence, i.e. $j, l < i$.

Suppose formula F consists of m clauses. Without loss of generality we may assume in this definition that the first m clauses of the sequence are the clauses of F. As the *length* of the proof $\mathcal{R}$ we count the number of clauses which are resolvents, i.e. $t - m$.

We use this not very suggestive representation of resolution proofs, as a sequence of clauses according to the definition, mainly for proof-technical reasons. A graphic description of a resolution proof can be given as follows. As an example, let

$$F \;=\; (x \vee \overline{z}) \wedge (x \vee y \vee z) \wedge (\overline{x} \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee y) \wedge (\overline{y} \vee z)$$

A resolution refutation of F can be represented as in the figure.



The next two lemmas will be useful when dealing with resolution proofs. The first lemma shows how an existing resolution proof can be restricted by a partial assignment such as $\{u = 1\}$. The second lemma shows how a resolution proof can be extended by adding a literal u into certain clauses.

Resolution Restriction Lemma
Let $\mathcal{R} = (C_1, C_2, \ldots, C_t)$, $C_t = \square$, be a resolution proof for a set of clauses F. Let u be a literal occurring in F. Then there is a resolution proof $\mathcal{R}'$ for $F\{u = 1\}$. This new resolution proof does not contain any clause with literal u or $\overline{u}$. Further, the length of $\mathcal{R}'$ is at most the length of $\mathcal{R}$ minus the number of clauses in F which contain literal u.

Proof: We construct the desired resolution proof $\mathcal{R}'$ iteratively, for $i = 1, \ldots, t$, from the proof $\mathcal{R}$. In every step there are several possibilities: either the clause C_i from $\mathcal{R}$ will be cancelled and does no longer occur in $\mathcal{R}'$. This happens exactly when $u \in C_i$. The second possibility is that a clause $C'_i \subseteq C_i$ is adopted in $\mathcal{R}'$. In logical terms this means that C'_i implies C_i. This clause C_i is either a clause from $F\{u = 1\}$, or it is a resolvent of two clauses $C'_j \subseteq C_j$ and $C'_l \subseteq C_l$, which have been listed in $\mathcal{R}'$ before, i.e. $j < l < i$. Another possibility is that C'_i is a duplicate of another clause C'_j from $\mathcal{R}'$ that was listed before (i.e. $j < i$, $C'_j = C'_i$).

1. If C_i is a clause from F and does not contain u or $\overline{u}$, then $\mathcal{R}'$ also lists C_i at this point, i.e. $C'_i = C_i$.

2. If C_i is a clause from F and contains $\overline{u}$ then we let $C'_i = C_i \setminus \{\overline{u}\}$.

3. If C_i contains the literal u (whether C_i comes from F or is a resolvent), then we don't have a clause in $\mathcal{R}'$ at this point (i.e. the clause is cancelled).

4. If C_i is the resolvent of two clauses C_j, C_l, and there exist respective clauses C'_j and C'_l in $\mathcal{R}'$, and these clauses contain the variable on which they were resolved, then C'_i is the resolvent of C'_k and C'_l.

5. If C_i is the resolvent of two clauses C_j and C_l, and there exist respective clauses C'_j and C'_l in $\mathcal{R}'$, and one of these clauses, say C'_j, no longer contains the variable on which they were resolved, then we let $C'_i = C'_j$ (i.e. a duplication). Observe that in this case we also have $C'_i \subseteq C_i$, as desired.

6. If C_i is the resolvent of two clauses C_j and C_l, but one of the parent clauses, say C_j, was cancelled (according to rule 3) because it contained u, then C_i does not contain u (otherwise rule 3 would have applied). It was the literal u resp. $\overline{u}$ which was resolved on in this resolution step. Consequently, C'_l does no longer contain $\overline{u}$, and we let $C'_i = C'_l$ (i.e. a duplication). Again, we have $C'_i \subseteq C_i$, as desired.

Since C_t does not contain u, a corresponding clause C'_t exists in $\mathcal{R}'$. From $C'_t \subseteq C_t = \Box$ it follows that $C'_t = \Box$. By renumbering the non-cancelled clauses in $\mathcal{R}'$, and by listing the duplicate clauses just once, one obtains a resolution refutation for $F\{u = 1\}$. $\Box$

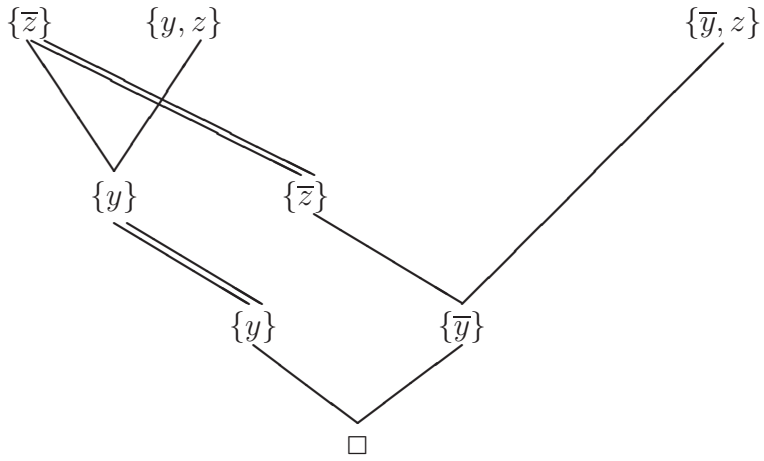
As an example, we take the resolution proof of the formula

$$F \;=\; (x \vee \overline{z}) \wedge (x \vee y \vee z) \wedge (\overline{x} \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee y) \wedge (\overline{y} \vee z)$$

as presented above. By setting $x = 0$ we obtain

$$F\{x = 0\} \;=\; (\overline{z}) \wedge (y \vee z) \wedge (\overline{y} \vee z)$$

The restricted resolution proof, obtained by applying the rules of the lemma, has the following form, where we indicate duplications by a double line.



Resolution Expansion Lemma
Let F be a clause set and u be a literal occurring in F. Let $\mathcal{R}'$ be a resolution refutation of $F' = F\{u = 1\}$. Then there exists a sequence $\mathcal{R}$ of resolution steps, based on the clause set F which either ends with $\Box$, like $\mathcal{R}'$, or ends with the unit clause $\{\overline{u}\}$.

Proof: For each clause $C \in F$ which contains the literal $\overline{u}$ there exists in $\mathcal{R}'$ a corresponding clause $C' = C \setminus \{\overline{u}\}$. By reinstalling $\overline{u}$ in such clauses – as well as in all subsequent resolvents – this modification can have the effect that the final clause $\Box$ becomes the unit clause $\{\overline{u}\}$. $\Box$

Resolution Theorem
The resolution calculus for clause sets is *correct* and *refutation complete*, that is, a clause set F is unsatisfiable *if and only if* there exists a resolution refutation of F.

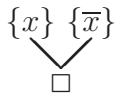
Proof: (Correctness) If there exists a resolution refutation of F, and if we assume that F is satisfiable with some assignment α, then, by the remark given after the definition of resolution, this assignment also satisfies each resolvent, especially the last clause which is the empty clause. But this is impossible. Therefore, F must be unsatisfiable.

(Refutation completeness) Conversely, let F be unsatisfiable. We show by induction on $n = |\mathit{Var}(F)|$, the number of variables, that F has a resolution refutation.

If $n = 0$, we have $F = \{\Box\}$, and we are done.

If $n > 0$, let x be an arbitrary variable in F. Then both clause sets $F_0 := F\{x = 0\}$ and $F_1 := F\{x = 1\}$ are unsatisfiable, too. These clause sets contain at most $n - 1$ variables. Therefore, F_0 and F_1, by the induction hypothesis, have resolution refutations, say $\mathcal{R}_0$ and $\mathcal{R}_1$.

By reestablishing the original clauses of F in $\mathcal{R}_0$ and $\mathcal{R}_1$, as in the resolution expansion lemma, one obtains two resolution sequences ending in the two unit clauses $\{x\}$ and $\{\overline{x}\}$ (or with $\Box$). In a last resolution step



the empty clause can be obtained. $\Box$

It can be observed that the length $l(n)$ of the resolution refutation constructed in this proof satisfies

$$l(n) \leq \begin{cases} 0, & n = 0 \\ 2 \cdot l(n-1) + 1, & n > 0 \end{cases}$$

where n is the number of variables in F. This recursion results in $l(n) \leq 2^n - 1$. Furthermore, the resolution proof constructed has a tree structure, a so called *tree resolution*, in the sense that every resolvent is used in at most one further resolution step. In many cases resolution proofs can be much shorter, especially when they do not have a tree structure.

Next, we present an alternative proof for the refutation completeness of resolution (from a personal communication with Volker Diekert). We assume that all potential resolvents derivable from F have been constructed (which form a finite set). Let us denote this set of clauses (i.e. F and all conceivable resolvents) as G. This is, G is the closure of F under resolution. We will show that under the precondition $\square \notin G$, a satisfying assignment for F can be constructed. This assignment naturally also satisfies G since all clauses in G follow from the clauses in F. We start with the empty assignment $\alpha = \emptyset$, and successively add an assignment to each variable. Let $x \in \mathit{Var}(F)$ be an as yet unassigned variable. If the unit clause $\{x\}$ occurs in G (which prevents $\{\overline{x}\}$ from being in G, otherwise also $\square \in G$), then we set $\alpha := \alpha \cup \{x = 1\}$, otherwise, if $\{x\} \notin G$, we set $\alpha := \alpha \cup \{x = 0\}$.

In the next step we set $G := G\{x = a\}$ where $a = 1$ in the former case, and $a = 0$ in the latter case.

Before we continue with the discussion, let us verify that this modification of α and of G satisfies the following invariants:

First, $G\{x = a\}$ is also closed under resolution. Whenever two clauses in $G\{x = a\}$ are resolvable, then the original clauses in G neither contain x nor $\overline{x}$ (otherwise at least one of these clauses will be deleted in $G\{x = a\}$). Therefore, the resolvent is also present in $G\{x = a\}$.

Second, it holds that $\square \notin G\{x = a\}$. The empty clause could only appear if, in the case of $a = 1$, the unit-clause $\{\overline{x}\}$ was present in G. But this is impossible, by the discussion above. The other case $a = 0$ is symmetric.

We continue by fixing assignments for variables and plugging these assignments into the clause set G. Finally, $G = \emptyset$ which corresponds to a tautology. This means that F is satisfied by the determined assignment to the variables.

Example: Consider the given clause set

$$F \;=\; \{\, \{x\}, \{\overline{x}, z\}, \{\overline{x}, y, z\}, \{\overline{y}, \overline{z}\} \,\}$$

We construct every possible resolvent, starting out from F, and obtain the following set:

$$G \;=\; F \cup \{\, \{z\}, \{y, z\}, \{\overline{x}, \overline{y}\}, \{\overline{x}, z, \overline{z}\}, \{\overline{x}, y, \overline{y}\}, \{\overline{y}\}, \{z, \overline{z}\}, \{y, \overline{y}\}, \{\overline{x}, \overline{y}, \overline{z}\} \,\}$$

Since $\{x\} \in G$ we let the assignment to x be 1, and in the next step we get

$$G\{x = 1\} \;=\; \{\, \{z\}, \{y, z\}, \{\overline{y}, \overline{z}\}, \{\overline{y}\}, \{z, \overline{z}\}, \{y, \overline{y}\} \,\}$$

Since $\{y\} \notin G\{x = 1\}$ we set y to 0, and we get

$$G\{x = 1, y = 0\} \;=\; \{\, \{z\}, \{z, \overline{z}\} \,\}$$

Since $\{z\} \in G\{x = 1, y = 0\}$ we set z to 1, and we finally get

$$G\{x = 1, y = 0, z = 1\} \;=\; \emptyset \quad \text{(tautology)}$$

This means that $\{x = 1,\ y = 0,\ z = 1\}$ is a satisfying assignment for F.

We have presented both proofs for refutation completeness since each proof provides a different constructive argument. Under the assumption that F is unsatisfiable, we have constructed a resolution refutation for F in the first proof. Under the assumption that the empty clause cannot be resolved from F, we have constructed a satisfying assignment for F in the second proof.

Finally, observe that there is an interesting connection between resolution refutations and BDD's (see the appendix). If one turns a resolution refutation "upside-down" so that it starts with the empty clause, the resulting graph can be interpreted as a BDD. Each node in this graph is assigned the variable which was used for the respective resolution step. Each path in this graph corresponds to an assignment α, and it ends in a clause from F which is falsified by this particular assignment. This connection between resolution proofs and BDD's can be used to transform lower bound proofs for the size of BDD's to lower bound proofs for the length of resolution refutations, and vice versa.

2.3 Unit Clauses, Subsumption and Pure Literals

Inspecting the proof of the resolution completeness theorem, it should be clear that in general it is not possible to remove the parent clauses once they have been used for a resolution step. It is still possible that such a parent clause needs to be used for another resolution step. In this section several forms of possible simplifications will be discussed, in the sense of an equivalent or a sat-equivalent transformation.

Example: The clause set $F = \{\{x\}, \{\overline{x}, y\}, \{\overline{x}, \overline{y}\}\}$ is unsatisfiable. By resolving the first two clauses one obtains $\{y\}$. After this, it is not possible to remove $\{x\}$ since this clause has to be used for another resolution together with the third clause of F.

In contrast, it is possible to remove the second clause of F since this parent clause $\{\overline{x}, y\}$ is a logical consequence of the resolvent $\{y\}$.

Definition

If we have two clauses C_1, C_2 such that $C_1 \subset C_2$, it is said that the clause C_1 *subsumes* the clause C_2 (in logical terms, this simply means that $C_1 \rightarrow C_2$ is true).

Theorem

If C_1 subsumes C_2, i.e. $C_1 \subset C_2$, and both clauses occur in a clause set F, then F and $F \setminus \{C_2\}$ are equivalent. That is, C_2 can be eliminated from F.

Proof: If $F \setminus \{C_2\}$ is satisfied by some assignment α, then α also satisfies C_1 which occurs in $F \setminus \{C_2\}$, and since $C_1 \subset C_2$, the clause C_2 is also satisfied by α, and therefore also $F\alpha = 1$. □

So it is possible to remove a clause from a clause set without harm if it is subsumed by another clause. On the other hand, searching for a clause set for pairwise subsumption is relatively expensive. Therefore, often such a test is not implemented. Often it turns out that the fact of one clause subsuming another clause comes about "automatically", as a side effect, for example during a resolution step with a unit clause as resolvent. In this case (as in the example above) the resolvent subsumes one of its parent clauses, and this parent clause can be removed.

Usually we are content with transformations which are sat-equivalent. There are more possibilities to achieve sat-equivalence than to achieve equivalence. The above example contains the unit clause $\{x\}$. Using unit propagation (cf. page 18) it is possible to apply the assignment $x = 1$ and to obtain the sat-equivalent formula $F\{x = 1\} = \{\{y\}, \{\overline{y}\}\}$. Now, $\{y\}$ is also a unit clause (as well as $\{\overline{y}\}$). After another unit propagation one obtains $F\{x = 1, y = 1\} = F\{x = 1, y = 0\} = \{\square\}$, hence F is unsatisfiable. So we could show unsatisfiability by applying unit propagation only, without resolution. Later we will see that unit propagation is always successful in the case of (renamable) Horn formulas.

Another possibility for formula simplification was already mentioned in the section on autark assignments. Suppose we have a *pure literal* u which occurs in F (possibly at several places), but the complementary literal $\overline{u}$ does not occur anywhere in F. In this case the formula F can be simplified to $F\{u = 1\}$. The formulas F and $F\{u = 1\}$ are sat-equivalent.

Finally, to discuss the case in which a resolution step involves a *tautology*, that is, a clause which contains a variable together with its complement. This concerns both a tautology as parent clause, as well as a tautology as resolvent.

Consider the following example: $\{x_1, x_2, \overline{x_3}\}$ and $\{\overline{x_2}, x_3, x_4\}$. It is possible to resolve these two clauses using the variable x_2 producing the resolvent $\{x_1, \overline{x_3}, x_3, x_4\}$. Another possibility is to use the variable x_3 for resolution and obtain the resolvent $\{x_1, x_2, \overline{x_2}, x_4\}$. Note that the resolution rule, as defined, does not allow us to resolve to $\{x_1, x_4\}$. Actually, this would be logically incorrect. Indeed, both resolvents are tautologies. In order to achieve the final goal to derive the empty clause, there is no progress in deriving or using a tautology in the resolution proof. Actually, the number of resolution steps increases unnecessarily. There is no harm, no loss of refutation completeness, if the use of tautologies in resolution proofs is forbidden. This can be considered as a simple form of complete resolution restriction, a notion that is considered in the next section.

Formally, it is not hard to see that "resolution without tautologies" is still a refutation complete calculus. The statement and the proof of the resolution theorem (page 42) can be easily adapted, essentially by changing everywhere "resolution proof" to "resolution

proof without tautologies". Another way to see this is by way of the last theorem: a tautology is subsumed by any other clause, and thus, can be eliminated.

2.4 Strategies and Restrictions

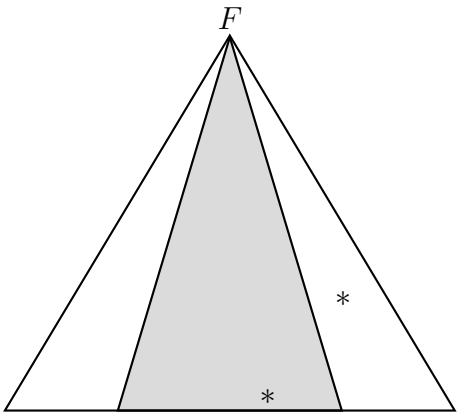
Resolution is a *nondeterministic* calculus. By fixing a particular sequential arrangement of the resolution steps one obtains a *deterministic* algorithm. The ordering of resolution steps in a particular way, thereby obtaining a deterministic process, is called a (resolution) *strategy*. Since every possible resolution step is performed sooner or later, a strategy still has the property of being refutation complete. The hope is that doing the resolution steps in a particular order will lead to the empty clause somewhat faster.

An example for a strategy is the *unit preference strategy* whereby resolution steps involving unit clauses are always done whenever possible. Applying this strategy does not lead to a deterministic algorithm since there are still nondeterministic choices if no unit resolution step is available, but still, the amount of nondeterminism is certainly restricted.

The DPLL-type algorithms discussed in chapter 4 can be considered as certain types of resolution strategies since the algorithmic transformations done by these algorithms can be understood as resolution steps.

In contrast, a *resolution restriction* means that certain resolution steps, if they do not satisfy certain criteria, are simply forbidden. The advantage is, again, that the scope of nondeterministic alternatives is restricted. On the negative side, it can be observed that the proof length might increase as compared to the general case.

The following diagram outlines this effect.



The grey area indicates the restricted nondeterministic search space; the stars indicate solutions (i.e. the possibility of deriving the empty clause).

Even worse than lengthening the proof is possible: a strong restriction of a calculus might result in the loss of the (refutation) completeness. In the above picture this means there is no star at all in the grey area. Later we will see an example of a restriction that makes resolution incomplete.

Definition
We define several resolution restrictions: *P-resolution* (or, the *positive* resolution restriction) means that in each resolution step it is required that one of the parent clauses has to be positive (i.e. consists of positive literals only).
Similarly, *N-resolution* (or, the *negative* resolution restriction) means that one of the parent clauses must consist of negative literals only.
The *linear resolution restriction* (or, linear resolution, for short) requires that the resolution proof forms a linear chain. First, a so-called *base clause* has to be selected from F. It forms the first element of the chain. From then on each resolution step uses the last element of the chain as one of the parent clauses. After the resolution step the resolvent forms the new last element of the chain.

We show that these resolution restrictions are still refutation complete.

Theorem
P-resolution is refutation complete.

Proof: Let F be an unsatisfiable clause set. We show by induction on $n = |\mathit{Var}(F)|$ that it is possible to obtain the empty clause by P-resolution steps only.

(*Case* $n = 0$): In this case we have $F = \{\Box\}$, and there is nothing to prove.

(*Case* $n > 0$): Let $x \in \mathit{Var}(F)$. Since $F\{x = 1\}$ and $F\{x = 0\}$ are unsatisfiable and contain at most $n - 1$ variables, the induction hypothesis can be applied and yields P-resolution proofs starting from $F\{x = 0\}$ as well as from $F\{x = 1\}$. By re-entering the variable x into the P-refutation of $F\{x = 0\}$ we obtain a sequence of resolution steps starting from clauses in F which lead to the unit clause $\{x\}$ (cf. the resolution expansion lemma). Notice that this is still a P-resolution since x is a positive literal. Next we add resolution steps which resolve the resulting unit clause $\{x\}$ with each clause in F that contains $\overline{x}$. Again, these are P-resolution steps. We obtain the clauses in $F\{x = 1\}$. Finally, the P-resolution proof of $F\{x = 1\}$ can be attached which exists by the induction hypothesis. □

Theorem
N-resolution is refutation complete.

Proof: Interchange the roles of positive and negative, x and $\overline{x}$, as well as $F\{x = 0\}$ and $F\{x = 1\}$ in the previous proof. □

Theorem
Linear resolution is refutation complete.
More precisely: for every unsatisfiable clause set F there exists a clause $C \in F$ such that the empty clause can be derived by a linear chain of resolution steps, starting with the base clause C.

Proof: Let F be unsatisfiable and let $F' \subseteq F$ be a minimally unsatisfiable subset (i.e. F' is also unsatisfiable, but for every clause $C \in F'$ the clause set $F' \setminus \{C\}$ is satisfiable). We claim that every clause C in F' can be used as the base clause in a linear resolution proof. The proof of this claim proceeds again by induction on the number n of variables, $n = |\mathit{Var}(F')|$.

Induction base $(n = 0)$:
In this case we have $F' = \{\Box\}$ and $C = \Box$. There is nothing to prove.

Induction step $(n > 0)$:
Case 1: $|C| = 1$, hence $C = \{u\}$ for some literal u. The formula F' must have a clause C' with $\overline{u} \in C'$. The clause set $F'\{u = 1\}$ is unsatisfiable and contains at most $n - 1$ variables. Let F'' be a minimally unsatisfiable subset of $F'\{u = 1\}$. In F'' there exists the clause $C' \setminus \{\overline{u}\}$. Consider an assignment α which satisfies all clauses in F' except C'. This assignment exists since F is minimally unsatisfiable. Since $u\alpha = 1$, the assignment α also satisfies all clauses in $F'\{u = 1\}$ except $C' \setminus \{\overline{u}\}$. Therefore, this clause must be contained in F''.

By the induction hypothesis there is a linear resolution refutation of $F'\{u = 1\}$, based on the clause $C' \setminus \{\overline{u}\}$. The desired linear refutation, based on $C = \{u\}$, is now constructed as follows. The first step resolves C with C'. The obtained resolvent is $C' \setminus \{\overline{u}\}$. Afterwards we take the linear resolution based on $C' \setminus \{\overline{u}\}$ given by the induction hypothesis. Each time, in this resolution proof, where in the initial clauses the literal $\overline{u}$ was eliminated, we re-insert this literal in the proof and obtain as the last clause the unit clause $\{\overline{u}\}$ (cf. resolution expansion lemma). In a last step, we resolve $\{\overline{u}\}$ against the base clause $\{u\}$ and obtain the empty clause.

Case 2: $|C| > 1$.
In this case we choose an arbitrary literal $u \in C$ and let $C' = C \setminus \{u\}$. The clause set $F'\{u = 0\}$ is unsatisfiable and it contains C' as a clause. Let α be an assignment which satisfies all clauses in F' except C. This assignment exists since F is minimally unsatisfiable. Since $u\alpha = 0$, the assignment α also satisfies all clauses in $F'\{u = 0\}$ except C'. Therefore, this clause must be contained in a minimally unsatisfiable subset F'' of $F'\{u = 0\}$. Applying the induction hypothesis on F'' gives a linear resolution of $F'\{u = 0\}$ based on the clause C'. By re-inserting the literal u (cf. resolution expansion lemma) we obtain a linear resolution of F based on C which ends with the unit clause $\{u\}$.

Now $(F' \setminus \{C\}) \cup \{\{u\}\}$ is unsatisfiable (observe that $\{u\}$ subsumes C) and $F' \setminus \{C\}$ is satisfiable. Therefore, Case 1 can be used which implies that a linear resolution refutation of $(F' \setminus \{C\}) \cup \{\{u\}\}$ exists based on $\{u\}$. By attaching this refutation to the one obtained earlier which leads to $\{u\}$ we obtain the desired linear resolution refutation for F based on the clause C. □

In the following we define and analyze *Davis-Putnam resolution* (cf. (Davis, Putnam, 1960), *DP-resolution* for short), which can be used and extended to define an algorithm, called *Davis-Putnam algorithm* or *DP algorithm* for short. The particular use of resolution here has both the character of a strategy and a restriction. It is partly a strategy because one needs to fix a particular order of the variables and apply the basic steps of DP-resolution in this order. The order can be dynamically fixed during the process.

In a basic step in DP-resolution we first produce all resolvents from a given clause set F which can be derived by a particular fixed variable x (as resolution variable). Afterwards all parent clauses of these resolvents are eliminated resulting in a new clause set which does no longer contain the variable x.

Let us describe this process more formally. Let $F_x \subseteq F$ be those clauses in F which contain x or its complement $\overline{x}$. Let $R_x(F)$ be the set of resolvents which can be obtained from F_x by using variable x for resolution. Observe that no clause in $R_x(F)$ contains x anymore. The set of clauses F from which we started will now be replaced by $(F \cup R_x(F)) \setminus F_x$.

Theorem (completeness of DP-resolution)
The clause set F is satisfiable if and only if $(F \cup R_x(F)) \setminus F_x$ is satisfiable.

Proof: If F is satisfiable, then so is $F \cup R_x(F)$, by the correctness of the resolution calculus. Therefore, also $(F \cup R_x(F)) \setminus F_x$ is satisfiable.

Conversely, suppose $(F \cup R_x(F)) \setminus F_x$ is satisfiable by some assignment α (which does not involve x). Suppose there is a clause $C \cup \{x\}$ in F_x not being satisfied by α. Let $F' \subseteq F_x$ be the set of clauses in F_x which contain the literal $\overline{x}$. The clause $C \cup \{x\}$ can be resolved with every clause in F' (using variable x) and the resolvent then belongs to $R_x(F)$. By assumption, all these resolvents are satisfied by α. Therefore, *all* clauses in F' are satisfied by α (without an assignment to x – otherwise α would satisfy C). By setting $\beta = \alpha \cup \{x = 1\}$, we obtain an assignment that satisfies F.

An analogous argument holds if there is a clause $C \cup \{\overline{x}\}$ in F_x that is not satisfied by α. □

Let us consider an example. A formula in 3-*CNF* has 100 variables and consists of 400 clauses, each with 3 literals. Therefore, there are 1200 occurrences of literals in the formula. On average, a variable occurs in $1200/100 = 12$ clauses. Suppose that x has 6 positive instances and 6 negative instances in the clauses. Then we have $|F_x| = 12$ and

$|R_x(F)| = 6 \cdot 6 = 36$. After a basic DP-step, F is replaced by $(F \cup R_x(F)) \setminus F_x$. This new clause set has 99 variables and $400 + 36 - 12 = 424$ clauses of which at most 36 clauses might be in 4-*CNF*.

It might happen that some of the clauses in $R_x(F)$ coincide with each other or with clauses being already in F, so we do not actually need to add these clauses. Furthermore, it is possible that a clause in $R_x(F)$ contains complementary literals stemming from its two parent clauses. This is a tautology clause and can be eliminated without harm (cf. page 45). Therefore, it might happen (not so seldom) that we obtain a clause set with fewer variables than before *and* with no more clauses than before. Because of this, this procedure has been adapted for *preprocessing* clause sets before they become input to some *SAT* solver.

It remains to present the DP algorithm in pseudo code. Here, it seems advantageous to insert unit propagation (cf. page 18) between each basic DP-step.

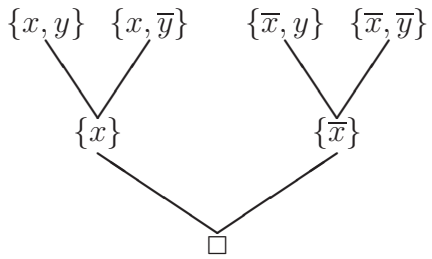
proc DavisPutnam (F : clause set) : **bool**
// yields 1, if F is satisfiable, and 0 otherwise
if $\square \in F$ **then return** 0
if $F = \emptyset$ **then return** 1
while (F contains a unit-clause $\{u\}$) **do** $F := F\{u = 1\}$
Select a variable $x \in \mathit{Var}(F)$
return DavisPutnam ($(F \cup R_x(F)) \setminus F_x$)

Although the number of clauses might decrease during a DP-step, the above example shows that it might also happen that the number of clauses increases dramatically. Therefore, heuristics for deciding which variable to select next play an important role here. On the other hand, once the last variables have to be selected, it is clear that the number of clauses (which can just contain the remaining variables) will decrease again, just as rapidly. It seems very difficult to prove theoretical bounds on the number of clauses which have to be handled during the course of a DP-resolution. Computer experiments show that other approaches like DPLL and its successors are more efficient than Davis-Putnam.

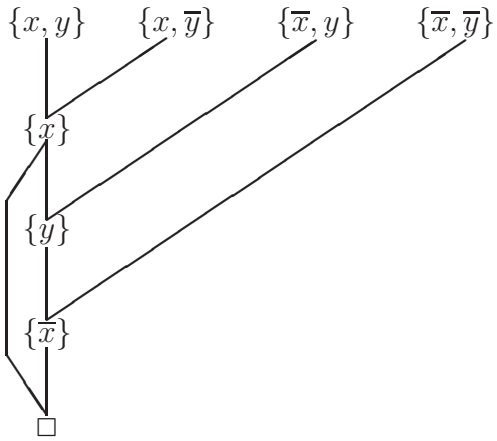
Next we define two resolution restrictions which lose their refutation completeness. Nevertheless they can be useful, especially in the context of Horn formulas as can be seen in the next chapter.

Definition
Input resolution requires that at least one of the parent clauses in a resolution step is an input clause, that is, it comes from the original clause set F.
The *unit resolution* restriction requires that at least one of the parent clauses in a resolution step is a unit clause.

These both resolution restrictions are incomplete as can be seen by the following example.



This resolution proof of length 3 is the shortest possible, but it does not satisfy any of the defined restrictions. The following proof



is a P-resolution, as well as a linear resolution, but it requires 4 resolution steps.

It is easy to check that neither a unit resolution nor an input resolution is possible. But we observe the following.

Theorem

A clause set F has a refutation with the unit resolution restriction *if and only if* it has a resolution refutation obeying the input resolution restriction.

Proof: We show by induction on the number of variables in F that every unit resolution proof can be transformed into an input resolution proof, and vice versa. This is obvious for $n = 0$ and $n = 1$.

Let $n > 1$. First we argue from unit to input. We assume that a unit resolution proof exists. Therefore, in the clause set F there must exist a unit clause $\{u\}$. We restrict the clause set and its resolution proof to $F\{u = 1\}$ according to the resolution restriction lemma. The restricted proof is still a unit resolution proof. By the induction hypothesis there is an input resolution proof for $F\{u = 1\}$. By re-inserting the literal $\overline{u}$ which was removed before, into the clauses and their respective resolvents (cf. resolution expanding

lemma) we obtain a sequence of input resolution steps (starting from F) which ends with the clause $\{\overline{u}\}$. Finally, a last input resolution step with $\{u\}$ yields the empty clause.

Now, conversely, suppose we have an input resolution proof for F. In the very last resolution step, two unit clauses are resolved producing the empty clause as resolvent. At least one of these, say $\{u\}$, has to stem from the input F. Again, consider $F\{u = 1\}$ and the restricted resolution proof which is still an input resolution. By the induction hypothesis this proof can be transformed to a unit resolution proof for $F\{u = 1\}$. We obtain the desired unit resolution proof for the original formula F as follows. First, we start by resolving each clause in F which contains $\overline{u}$ with the unit-clause $\{u\}$. Now we have all clauses in $F\{u = 1\}$ available and can add the unit resolution proof of $F\{u = 1\}$ which leads to the empty clause. □

2.5 Exponential Lower Bounds for the Length of Resolution Proofs

In this chapter we show that the length of the refutations of certain formulas must be exponential with respect to the formula size. The clause set PH_n is especially difficult to refute for the resolution calculus. This set of clauses encodes the *pigeonhole principle* from Dirichlet as a Boolean formula: if $n + 1$ pigeons are placed in n pigeonholes, then at least one pigeonhole must contain at least two pigeons. We define an unsatisfiable formula when we postulate that the $n + 1$ pigeons can be placed in such a way that each pigeonhole contains at most one pigeon. We use for this the variables $x_{i,j}$ in the formula, interpreting that the variable takes the value 1 when pigeon i is placed in pigeonhole j.

Definition
The pigeonhole set of clauses PH_n is defined for every $n > 0$ as follows: On the one hand for $i = 0, 1, \ldots, n$ the clauses

$$(x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n})$$

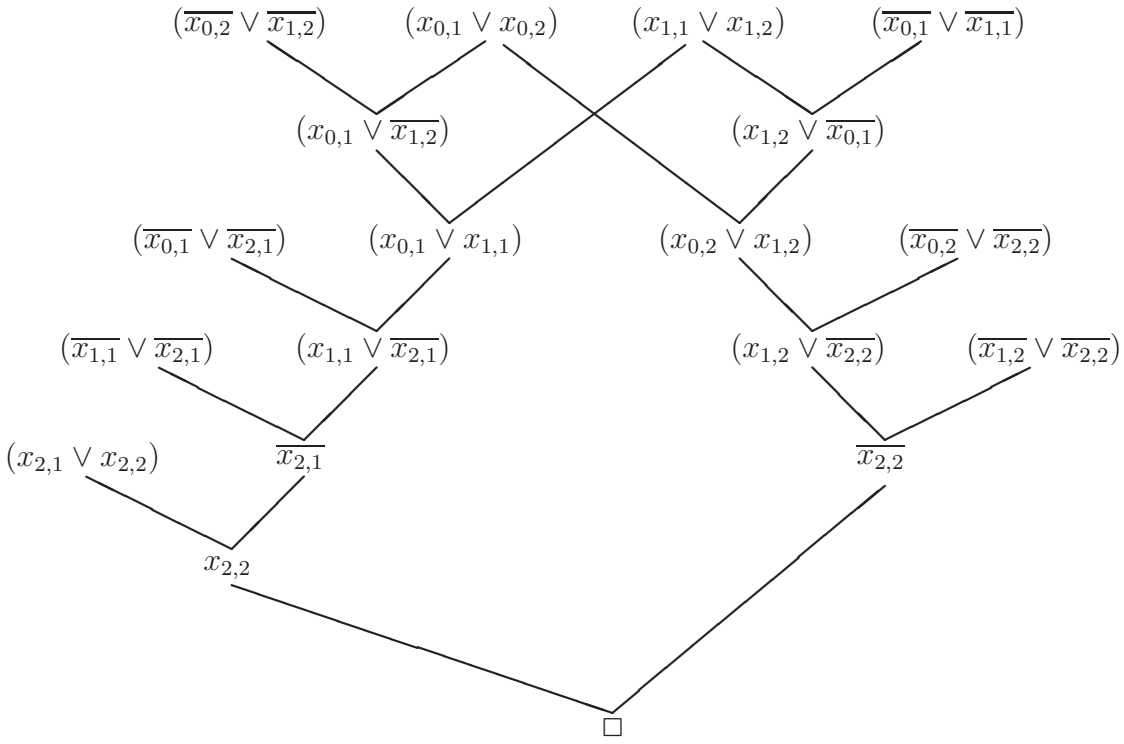
indicate that "pigeon i must be placed in pigeonhole 1 or 2 or ... or n."
On the other hand, we have for every $j = 1, 2, \ldots, n$ the set of $\binom{n+1}{2}$ clauses

$$(\overline{x_{0,j}} \vee \overline{x_{1,j}}),\ (\overline{x_{0,j}} \vee \overline{x_{2,j}}), \ldots, (\overline{x_{n-1,j}} \vee \overline{x_{n,j}})$$

indicating that "in pigeonhole j at most one pigeon can be placed."

It is clear that one cannot place $n + 1$ pigeons in n pigeonholes in this way, and therefore the set of clauses PH_n is unsatisfiable. We observe that PH_n contains $n \cdot (n + 1) = O(n^2)$ Boolean variables and has $(n + 1) + n \cdot \binom{n+1}{2} = O(n^3)$ clauses.

The following is a resolution refutation of PH_2 (that is, 3 pigeons do not fit in 2 pigeonholes):



From this example one already can get the feeling that such a refutation is (and has to be) complex. The following theorem proves this.

Theorem

The length of every resolution refutation of PH_n is at least $2^{n/20}$.

Let k be the number of variables and m the number of clauses in PH_n. This result says that the length of a resolution refutation for PH_n is at least $2^{\Omega(\sqrt{k})}$, respectively $2^{\Omega(\sqrt[3]{m})}$.

Proof: Let us suppose that there is a resolution refutation $\mathcal{R} = (K_1, K_2, \ldots, K_s)$, $K_s = \square$, of PH_n with $s < 2^{n/20}$. We associate with every clause K in $\mathcal{R}$ a clause $\widehat{K}$, by substituting in K every negative literal $\overline{x_{i,j}}$ by the set of positive literals $x_{i,k}$ for $k \neq j$. In the following we will call a clause K in $\mathcal{R}$ *large*, if the associated clause $\widehat{K}$ contains at least $n(n+1)/10$ (positive) literals. Let r be the number of large clauses in the resolution refutation $\mathcal{R}$. Of course, $r \leq s$ holds. There must be at least $rn(n+1)/10$ literals in the r many clauses associated with the large clauses. Therefore, there has to be a literal $x_{i,j}$, that appears in at least $r/10$ such clauses. This is also a consequence of the pigeonhole principle. By applying step by step the following partial assignment to $\mathcal{R}$, according to the Resolution Restriction Lemma,

$$\{x_{i,j} = 1\} \cup \{x_{i,k} = 0 \mid k \neq j\} \cup \{x_{l,j} = 0 \mid l \neq i\}$$

we obtain a resolution refutation for PH_{n-1}, since the assignments mentioned above "freeze" the correspondence from pigeon i to pigeonhole j, and only a resolution refutation of a PH_{n-1}-formula remains. Pigeon i and pigeonhole j do no longer appear in this PH_{n-1}-formula. Besides, because of this action, as one can infer from the Resolution Restriction Lemma, at least $r/10$ large clauses K are eliminated from the new resolution refutation for PH_{n-1}. This is because from this assignment either the positive literal $x_{i,j}$ was set to 1 in K, or a negative literal of the form $\overline{x_{i,k}}$, $k \neq j$, was set of 0. After such a restriction step, the obtained resolution refutation for PH_{n-1} contains at most $9r/10 \leq 9s/10$ large clauses. Continuing in this way, eliminate in each step a fraction of at least one tenth of the remaining large clauses, so that after at most $\log_{10/9} s$ many restriction steps, no large clauses are left. At this point we have to deal with a resolution refutation for $PH_{n'}$ satisfying

$$n' \;\geq\; n - \log_{10/9} s \;> n - \log_{10/9} 2^{n/20} \;>\; 0.671 \cdot n\,.$$

This means that all (associated) clauses $\widehat{K}$ in the resulting resolution refutation for $PH_{n'}$ have fewer than $n(n+1)/10$, and therefore also fewer than $2n'(n'+1)/9$ literals. The following lemma shows that this is a contradiction. This implies that our hypothesis $s < 2^{n/20}$ is wrong, and the theorem is proven. □

Lemma
In every resolution refutation for PH_m there must be a clause C_0, so that the associated clause $\widehat{C_0}$ contains at least $2m(m+1)/9$ many literals.

Proof: Let $\mathcal{R}$ be a resolution refutation for PH_m.

An assignment α is called *k-critical*, when it assigns to every pigeon except for pigeon k exactly one pigeonhole and no two pigeons are assigned the same hole. More formally: α is k-critical, if there is a bijective function f from $\{0,1,\ldots,m\} \setminus \{k\}$ to $\{1,2,\ldots,m\}$, so that we have

$$\alpha(x_{i,j}) \;=\; \begin{cases} 1, & \text{if } f(i) = j \\ 0, & \text{otherwise} \end{cases}$$

We define the *significance* of a clause C in $\mathcal{R}$, abbreviated by $\sigma(C)$, as follows:

$$\sigma(C) \;=\; \sum_{k=0}^{m} [\,\text{there is a } k\text{-critical assignment, that falsifies } C\,]$$

The square brackets express an indicator function: when the proposition between the brackets holds, then its value is 1, otherwise it is 0. It is clear, that the initial clauses in PH_m have significance 1 or 0. The empty clause, at the end of the resolution refutation $\mathcal{R}$, has significance $m+1$. Moreover, when C is the resolvent of two clauses C_1, C_2,

having significance s_1 and s_2, then the significance of C is at most $s_1 + s_2$, since every assignment that falsifies C, falsifies also C_1 or C_2. In other words: there can be no assignment that falsifies C, while satisfying both C_1 and C_2. From this it follows that there must be a clause C_0 in $\mathcal{R}$ with significance $s \in [\frac{m+1}{3}, \frac{2(m+1)}{3}]$. One can choose for C_0 the first clause C in $\mathcal{R}$ with $\sigma(C) \geq \frac{m+1}{3}$. Consider an arbitrary k-critical assignment α, falsifying C_0 (and therefore also the associated clause $\widehat{C_0}$). According to the definition of $\sigma(C)$, one can choose such an assignment in s different ways, or in other words, for s many k's. Let j be such that *all* j-critical assignments satisfy the clause C_0 (and therefore also $\widehat{C_0}$). Such a j can be chosen in $m + 1 - s$ different ways; there are therefore $s \cdot (m + 1 - s)$ many such (k, j) combinations. We modify α in one position, so that it mutates to a j-critical assignment α': instead of not assigning pigeon k any hole and assigning j one pigeonhole, say l, α' assigns the hole l to the pigeon k, while j does not receive any pigeonhole; apart from this modification, there is no other difference between α' and α. As a consequence of the modification, α' satisfies the clause C_0. This implies that the literal $x_{k,l}$ must occur in $\widehat{C_0}$. On can prove in this way the existence of $s \cdot (m + 1 - s)$ many literals in $\widehat{C}$. Observe that for a fixed k-critical assignment α and changing values of j, also the values for l are different. From $s \in [\frac{m+1}{3}, \frac{2(m+1)}{3}]$ it follows, that $s \cdot (m + 1 - s) \geq \frac{2m(m+1)}{9}$, and this proves the lemma. □

This exponential lower bound for the set of clauses PH_n applies, of course, also to all algorithms proceeding in a way that can be interpreted as the implementation of resolution steps (for example the DP algorithm). Such algorithms require exponential running time when the input is PH_n. Because of this, PH_n is frequently used as an input benchmark for testing *SAT* solvers.

The clause setPH_n contains clauses of size n. One may wonder whether this fact was necessary to derive the exponential proof-length lower bound, or whether the result can still be obtained when the original clause set has only clauses of size at most 3, instead. Notice that clauses of size 2 are certainly not sufficient because the number of potential resolvents is polynomial in this case, cf. page 64.

A clause of size $n > 3$, like $(x_1 \vee x_2 \vee \cdots \vee x_n)$, can be split into several clauses of size 3 by introducing new auxiliary variables y_i:

$$(x_1 \vee x_2 \vee y_1) \wedge (\overline{y_1} \vee x_3 \vee y_2) \wedge (\overline{y_2} \vee x_4 \vee y_3) \wedge \cdots \wedge (\overline{y_{n-4}} \vee x_{n-2} \vee y_{n-3}) \wedge (\overline{y_{n-3}} \vee x_{n-1} \vee x_n)$$

This construction leads to a sat-equivalent formula and it can also be applied in the case of the PH_n clauses. This is nothing else as a reduction from *CNF-SAT* to 3-*SAT*. For the above proof still to go through, one needs to redefine the concept of a k-critical assignment. Apart from the assignment of the x-variables which is still defined as before, the assignment of the auxiliary y-variables is chosen such that a maximum number of the 3-*CNF* clauses above are satisfied. Notice that the 3-*CNF* clauses which can be obtained from the original clause

$$(x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n})$$

are all satisfied by a k-critical assignment, if $i \neq k$. Therefore, the property that $\sigma(C) \leq 1$ for all initial clauses C is still valid. This property is crucial for the proof.

A different method, based on Craig interpolants, is considered next for obtaining lower bounds for the length of resolution refutations. Craig's Interpolation Theorem says that for every tautology of the kind $F \rightarrow G$ there must be a formula I, containing only the variables common to F and G, so that both $F \rightarrow I$ and $I \rightarrow G$ hold. We show that it is possible to construct a circuit for I from a given resolution refutation $\mathcal{R}$ for $\neg(F \rightarrow G)$, that is, the *CNF* version of this formula. This circuit has approximately the same size as $\mathcal{R}$. From a lower bound for the circuit size of I it follows then a lower bound for the length of a resolution refutation for $\neg(F \rightarrow G)$. The formula $\neg(F \rightarrow G)$ is equivalent to $F \wedge \neg G$. We can interpret an interpolant for $F \rightarrow G$ as a circuit C over the variables *Var*$(F) \cap$ *Var*(G), which for every possible input decides whether F or $\neg G$ are falsified (at least one of both formulas must be false). We denote by A and B the *CNF* for F and $\neg G$ respectively. The gates of circuit C are functions from the set $\{0, 1, \vee, \wedge, sel\}$. We recall that the selector function (also called *if-then-else* function or *ite* function) $sel : \{0,1\}^3 \rightarrow \{0,1\}$ is defined as

$$sel(x, y, z) = \begin{cases} y & \text{if } x = 0 \\ z & \text{if } x = 1 \end{cases}$$

Theorem

Let $\mathcal{R}$ be a resolution refutation of the set of clauses $A \cup B$ (defined as described above from a tautology $F \rightarrow G$). There is then a Boolean circuit C over the set of variables *Var*$(A) \cap$ *Var*(B), which for every assignment α with *Var*(α) = *Var*$(A) \cap$ *Var*(B) satisfies the following conditions:

$C\alpha = 0 \Rightarrow A\alpha$ is unsatisfiable, and

$C\alpha = 1 \Rightarrow B\alpha$ is unsatisfiable.

Moreover, the circuit C has at most $O(|\mathcal{R}|)$ many gates.

Proof: In order to construct the circuit C, we use the structure of the resolution refutation $\mathcal{R} = (K_1, \ldots, K_t)$ for the set of clauses $A \cup B$. Let *Var*$(A) \cap$ *Var*$(B) = \{x_1, \ldots, x_p\}$, *Var*$(A) \setminus$ *Var*$(B) = \{y_1, \ldots, y_q\}$, and *Var*$(B) \setminus$ *Var*$(A) = \{z_1, \ldots, z_r\}$. We denote these sets as x-variables, y-variables, and z-variables.

We denote by H the conjunction of the clauses in $A \cup B$. We know as a consequence of the Resolution Restriction Lemma that for each assignment $\alpha = \{ x_1 = a_1, \ldots, x_p = a_p\}$ of the x-variables, there is a resolution refutation $\mathcal{R}_\alpha$ for $H\alpha$. The clauses in $\mathcal{R}_\alpha$ do not contain any x-variables. Moreover, a single clause in $\mathcal{R}_\alpha$ can contain only y-variables or z-variables, but not both. This holds because a clause that contains at the same time both y-variables and z-variables, can only happen after a resolution of an x-variable, and such resolution steps cannot exist in $\mathcal{R}_\alpha$. Considering this, we call a clause $K_i' \in \mathcal{R}_\alpha$

a y-clause, if it only contains y-variables, or if $K'_i = \Box$ and K'_i can be reached from a y-clause. The z-clauses are defined in a similar way. The empty clause K_t in $\mathcal{R}$ is transformed into a clause K'_t in $\mathcal{R}_\alpha$: it is either a y-clause or a z-clause. In the first case we have a refutation of $A\alpha$ as part from $\mathcal{R}_\alpha$ and in the second case a refutation of $B\alpha$. In other words, the decision of whether K'_t is a y-clause or a z-clause can be used as an interpolant for $A \cup B$. The circuit C will be constructed based on this idea. The clause C has the x-variables $\{x_1, \ldots, x_p\}$ as inputs, and for each clause K_i in $\mathcal{R}$ a gate g_i in C is defined. The following properties hold for each assignment α of the x-variables:

1. If K'_i is a y-clause in $\mathcal{R}_\alpha$, then g_i gets value 0 in $C\alpha$ and
2. If K'_i is a z-clause in $\mathcal{R}_\alpha$, then g_i gets value 1 in $C\alpha$.

The circuit C then has exactly t gates over the base $\{\vee, \wedge, sel\}$ (see the appendix on circuits) and when these conditions hold and g_t is the output gate, then C computes an interpolant. The gates of C are constructed as follows:

For the initial clauses in $\mathcal{R}$, if $K_i \in A$, then g_i is the constant 0, and if $K_i \in B$ then g_i is the constant 1.

In case K_i is the resolvent of two clauses K_j, K_l, and the variable which is resolved for is an x-variable x_k, then $g_i = sel(x_k, g_j, g_l)$.

In case K_i is the resolvent of two clauses K_j, K_l, and the variable which is resolved for is a y-variable, then $g_i = g_j \wedge g_l$.

In case K_i is the resolvent of two clauses K_j, K_l, and the variable which is resolved for is a z-variable, then $g_i = g_j \vee g_l$.

Let α be an assignment of the x-variables and let K'_i be a y-clause in $\mathcal{R}_\alpha$. We prove by induction over i that g_i takes the value 0. If K_i is an initial clause, then g_i takes the value 0 for each assignment. In case K_i is obtained by resolution of the variable x_k from the parent clauses K_j and K_l $(j, l < i)$ and $x_k \in K_j$, then, depending on whether the assignment α assigns variable x_k with 1 or with 0, so is $K'_i = K'_l$ or $K'_i = K'_j$ (cf. Rule 6 in the Resolution Restriction Lemma). We are dealing in both cases with a y-clause (otherwise K'_j and K'_l were not y-clauses) and because of the induction hypothesis, the value of g_l (respectively g_j) is 0. The sel function computes the correct value for g_i. In case K_i is obtained by resolution of the variable y from the parent clauses K_j and K_l, since K'_i is a y-clause, at least one of the clauses K'_j, K'_l must be a y-clause. The value of $g_i = g_j \wedge g_l$ under the assignment α is then 0. The case for the z-clauses is similar. $\Box$

If the x-variables only appear positively in the clauses in A, or only negatively in the clauses in B, then the construction of C in the previous theorem can be modified in such a way, that C is a monotone circuit (that is, it contains only gates from the set $\{0, 1, \wedge, \vee\}$). This is very useful since there exist strong lower bounds for monotone circuits. Such lower bounds can then be transformed into lower bounds for the length of resolution refutations for certain formulas.

Theorem

Let $\mathcal{R}$ be a resolution refutation of the set of clauses $A \cup B$. If the variables *Var*$(A) \cap$ *Var*(B) appear only positively in the clauses of A, respectively only negatively in the clauses of B, then there is a monotone Boolean circuit C over the variables *Var*$(A) \cap$ *Var*(B) such that for every assignment α of *Var*(C) satisfies the following conditions:

$C\alpha = 0 \Rightarrow A\alpha$ is unsatisfiable, or

$C\alpha = 1 \Rightarrow B\alpha$ is unsatisfiable.

Moreover, circuit C has at most $|\mathcal{R}|$ gates from the set $\{\vee, \wedge\}$.

Proof: Assume that the x-variables appear only positively in the clauses of A. When a negated x-variable appears in a clause K_i, then we get for every assignment α that K_i' cannot be a y-clause. The sel gates from the previous proof are not monotone: they are introduced in one of the cases in the construction of C. We transform this case in the following way: if K_i is obtained by resolution of variable x_k from the two clauses K_j, K_l, and x_k is an x-variable and $x_k \in K_j$, then $g_i = x_k \vee g_j$.

Let α be an assignment of the x-variables. Assume that K_i' is a y-clause or a z-clause in $\mathcal{R}_\alpha$ (that is K_i' was not eliminated in $\mathcal{R}_\alpha$). When x_k is assigned value 1, then $K_i' = K_l'$ (cf. Rule 6 in the Resolution Restriction Lemma). Since $\overline{x_k} \in K_l$, K_l' cannot be a y-clause and K_i' is then a z-clause. The value of g_i is defined as $x_k \vee g_j = 1$. When x_k is assigned value 0, then $K_i' = K_j'$. Accordingly we have that g_i equals g_j.

The case in which the x-variables only appear negated in the B clauses, is symmetric. □

We give as an example two families of formulas F and G, satisfying that $F \rightarrow G$ is a tautology, for which there can be only monotone interpolants of exponential size. These formulas are based on the clique function defined next.

We say that an undirected graph G with n vertices is an (n, k)-*clique*, if G contains a single clique with k vertices and does not have any further edges. That is, G has $n - k$ isolated vertices.

We say that G is an $(n, k-1)$-*coloring*, if the vertices of G can be partitioned in $k-1$ disjoint sets, so that no edges join vertices from the same set, but all the edges between the vertices of different sets are present.

It can be easily observed that for every k, if G is an (n, k)-clique, then G cannot be an $(n, k-1)$-coloring.

Let $Clique_{n,k} : \{0,1\}^{\binom{n}{2}} \rightarrow \{0,1\}$ be the Boolean function with variables $x_{i,j}$, $1 \leq i < j \leq n$, taking value 1 if the values of the variables encode an undirected graph G containing a clique of size k (and taking value 0 otherwise).

The way in which the input variables encode a graph should be interpreted as $x_{i,j} = 1$ if and only if there is an edge between the vertices i and j in G. We observe that $Clique_{n,k}$ is a monotone function.

In (Razborov, 1985) an exponential lower bound for the size of every monotone circuit for the clique function $Clique_{n,n^{1/4}}$ was proven. The proof of this lower bound can be extended to other functions in the following way: let $Q_{n,k}$ be any monotone Boolean function on the variables $x = (x_{i,j})_{1 \leq i < j \leq n}$, encoding a graph G_x with n vertices, and satisfying $Q_{n,k}(G_x) = 1$, if G_x is an (n,k)-clique, and $Q_{n,k}(G_x) = 0$, if G_x is an $(n,k-1)$-coloring (the value of the function is irrelevant in all the other cases). It follows then for $k = n^{1/4}$ and some $\varepsilon > 0$, that every monotone Boolean circuit computing $Q_{n,k}$ must have at least $2^{\Omega(n^\epsilon)}$ gates.

We consider two types of formulas $F_{n,k}(x,y)$ and $G_{n,k}(x,z)$, respectively, expressing that a graph G_x encoded in the x-variables contains a clique of size k and is $(k-1)$-colorable. The formula $F_{n,k}(x,y) \rightarrow \neg G_{n,k}(x,z)$ is a tautology and because of the Interpolation Theorem there must be an interpolant for the formula. The existence of $F_{n,k}$ and $G_{n,k}$ follows from the Cook-Levin Theorem (NP-completeness of *SAT*). We give nevertheless an explicit definition of the formulas in order to be able to estimate its size and that of the interpolant.

The formula $F_{n,k}(x,y)$ contains the variables $x = \{x_{i,j} \mid 1 \leq i < j \leq n\}$ and $y = \{y_{i,l} \mid 1 \leq i \leq n, 1 \leq l \leq k\}$. The x-variables encode the graph and variable $y_{i,l}$ is 1, if vertex i is the l-th vertex in the k-clique. $F_{n,k}$ is the conjunction of the following sub-formulas:

$$\bigvee_{l,m} (y_{i,l} \wedge y_{j,m}) \rightarrow x_{i,j} \quad 1 \leq i < j \leq n$$

expressing that in case the vertices i and j are both in the clique, then there must be an edge between them:

$$\bigvee_{i} y_{i,l} \quad 1 \leq l \leq k$$

$$\neg(y_{i,l} \wedge y_{j,l}) \quad 1 \leq i < j \leq n, 1 \leq l \leq k$$

$$\neg(y_{i,l} \wedge y_{i,m}) \quad 1 \leq i \leq n, 1 \leq l < m \leq k$$

These formulas express that exactly one vertex is the l-th vertex in the clique, and that every vertex appears in the clique at most once.

The formula $G_{n,k}(x,z)$ has the variables $x = \{x_{i,j} \mid 1 \leq i < j \leq n\}$ in the same way as $F_{n,k}$, and besides $z = \{z_{i,l} \mid 1 \leq i \leq n, 1 \leq l \leq k-1\}$. The x-variables encode the graph and the variable $z_{i,l}$ is 1 if and only if vertex i has color l. $G_{n,k}$ is the conjunction of the following sub-formulas:

$$x_{i,j} \rightarrow \bigwedge_{l} (z_{i,l} \rightarrow \neg z_{j,l}) \quad 1 \leq i < j \leq n$$

expressing that in case there is an edge between i and j, then i and j have different colors, and

$$\bigvee_{l} z_{i,l} \quad 1 \leq i \leq n$$

$$\neg(z_{i,l} \wedge z_{i,m}) \quad 1 \leq i \leq n,\, 1 \leq l < m \leq k-1$$

$$\bigvee_i z_{i,l} \quad 1 \leq l \leq k-1$$

These formulas express that every vertex has exactly one color, and that there is at least one vertex for each color.

The formula $F_{n,k}(x,y) \wedge G_{n,k}(x,z)$ is unsatisfiable. This formula can be written in *CNF*, without becoming much larger. Its length can be bounded by $m = O(n^4)$. Moreover, it can be observed that the x-variables only appear positively in the F formula. This formula satisfies the conditions of the theorem for the construction of monotone interpolants out of resolution refutations.

It follows from the lower bound for monotone circuits, that every monotone circuit computing an interpolant for the formulas $F_{n,k}(x,y)$ and $G_{n,k}(x,z)$ for $k = n^{1/4}$, must have size $2^{\Omega(m^\varepsilon)}$ for some $\varepsilon > 0$. From the previous theorem it follows then, that $2^{\Omega(m^\epsilon)}$ is also a lower bound for the size of a resolution refutation of $F_{n,n^{1/4}}(x,y) \wedge G_{n,n^{1/4}}(x,z)$.

The first exponential size lower bound for resolution refutations was proven by (Haken, 1985). The proof given here is an adaptation of the work of (Beame and Pitassi, 1996). The proof using Craig interpolants follows from the works of (Pudlák, 1997) and (Krajiček, 1994), which rely on a lower bound for the size of monotone circuits from (Razborov, 1985). More constructions for formulas having exponentially long resolution refutations have been given, based for example on expander graphs (Urquhart, 1987), (Schöning, 1997), or on random formulas (as in chapter 7), cf. (Chvátal, Szemerédi, 1988).