time, but it is more systematic than the method described here. Herbrand theory will be useful in proving that the resolution we define works.

3.4 Resolution for first-order logic

We now define resolution for first-order logic. Let φ be any sentence in SNF. Then φ has the form $\forall x_1 \forall x_2 \cdots \forall x_m \varphi_0$ where φ_0 is a conjunction of disjunctions of literals. In particular, φ_0 is quantifier-free, and so it can be viewed as a formula of propositional logic that is in CNF. Let $\mathcal{C}(\varphi_0)$ denote the set of all clauses in the CNF formula φ_0. We define $\mathcal{C}(\varphi)$ to be $\mathcal{C}(\varphi_0)$. That is, $\mathcal{C}(\varphi) = \{C_1, \ldots, C_m\}$ where C_i is the set of all literals occurring in the ith disjunction.

For example, if φ is the sentence

$$= \forall x \forall y \forall z((P(x,y) \vee \neg Q(x,z)) \wedge ((R(x,y,z) \vee \neg P(f(x,y),z)),$$

then $\mathcal{C}(\varphi)$ is the set

$$\{\{P(x,y), \neg Q(x,z)\}, \{R(x,y,z), \neg P(f(x,y),z)\}\}.$$

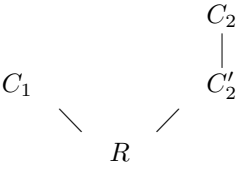
Note that a sentence φ in SNF uniquely determines $\mathcal{C}(\varphi)$. Conversely, by Proposition 1.67, $\mathcal{C}(\varphi_0)$ determines φ_0 up to equivalence. It follows that $\mathcal{C}(\varphi)$ determines φ up to equivalence. That is, if $\mathcal{C}(\varphi) = \mathcal{C}(\psi)$ for sentences φ and ψ in SNF, then $\varphi \equiv \psi$. For this reason, we need not distinguish between formulas in SNF and sets of clauses.

We want to say what it means for a clause R to be a *resolvent* of two clauses C_1 and C_2. As in propositional logic, a resolvent of C_1 and C_2 is a consequence of the conjunction of C_1 and C_2. Before giving a formal definition for resolvents, we consider a couple of examples.

Example 3.29 Let $C_1 = \{\neg Q(x,y), P(f(x),y)\}$ and $C_2 = \{\neg P(f(x),y), R(x,y,z)\}$. The clause $R = \{\neg Q(x,y), R(x,y,z)\}$ is a resolvent of C_1 and C_2. This works the same way as in propositional logic. Since the literal $P(f(x),y)$ occurs in one clause and the negation of this same literal occurs in the other, the resolvent can be formed by taking the union of C_1 and C_2 less $P(f(x),y)$ and $\neg P(f(x),y)$.

Example 3.30 Let $C_1 = \{\neg Q(x,y), P(f(x),y)\}$ and $C_2 = \{\neg P(z,y), R(x,y,z)\}$. Then we cannot directly find a resolvent of C_1 and C_2 as in the previous example. Let C_2' be the clause obtained by substituting $f(x)$ for z in the clause C_2. That is, $C_2' = \{\neg P(f(x),y), R(x,y,f(x))\}$. We make two observations. First, we can easily find a resolvent of C_1 and C_2', namely $R = \{\neg Q(x,y), R(x,y,f(x))\}$. Second, note that C_2' is a consequence of C_2. This is because the SNF sentence represented by C_2 asserts that the formula $\neg P(z,y) \vee R(x,y,z)$ holds for every

x, y, and z. In particular, this formula holds in the specific case where $z = f(x)$. That is, C_2 implies C_2'. Hence, R, which is a consequence of $\{C_1, C_2'\}$, is also a consequence of $\{C_1, C_2\}$. We define resolvents so that R is a resolvent of C_1 and C_2 (and of C_1 and C_2' as well). We diagram this situation as follows:



So prior to finding a resolvent, we must first make substitutions for variables to make certain literals look the same. In the previous example, we did a substitution that made $P(f(x), y)$ and $P(z, y)$ identical. This process is called *unification* and we postpone the formal definition of "resolvent" until after we have discussed unification in detail.

3.4.1 Unification. Let $\mathbb{L} = \{L_1, \ldots, L_n\}$ be a set of literals. We say $\mathbb{L}$ is *unifiable* if there exist variables $x_1, \ldots, x_m$ and terms $t_1, \ldots, t_m$ such that substituting t_i for x_i (for each i) makes each literal in $\mathbb{L}$ look the same. We denote such a substitution by $sub = (x_1/t_1, x_2/t_2, \ldots, x_m/t_m)$. For any sentence φ in SNF, we denote the result of applying this substitution to φ by φsub.

For example, if $sub = (x/w, y/f(a), z/f(w))$ and $\varphi = \{\neg Q(x, y), R(a, w, z)\}$, then $\varphi sub = \{\neg Q(w, f(a)), R(a, w, f(w))\}$.

If $\mathbb{L}$ is a set of literals, then $\mathbb{L}sub$ denotes the set of all $L_i sub$ such that $L_i \in \mathbb{L}$. So $\mathbb{L}$ is unifiable if and only if there exists a substitution sub such that $\mathbb{L}sub$ contains only one literal. If this is the case, we call sub a *unifier* for $\mathbb{L}$ and say that sub *unifies* $\mathbb{L}$.

Example 3.31 Let $\mathbb{L} = \{P(f(x), y), P(f(a), w)\}$. Let $sub_1 = (x/a, y/w)$ and $sub_2 = (x/a, y/a, w/a)$. Then both sub_1 and sub_2 unify $\mathbb{L}$. We have $\mathbb{L}sub_1 = \{P(f(a), w)\}$ and $\mathbb{L}sub_2 = \{P(f(a), a)\}$. Note that, by making another substitution, we can get $\mathbb{L}sub_2$ from $\mathbb{L}sub_1$. Namely, if $sub_3 = (w/a)$, then $sub_1 sub_3$ (sub_1 followed by sub_3) has the same effect as sub_2. However, we cannot generate $\mathbb{L}sub_1$ from $\mathbb{L}sub_2$ since $\mathbb{L}sub_2$ has no variables. So, in some sense, the unifier sub_1 is better for our purposes. It is more versatile. "Our purposes" will be resolution, and if we choose sub_2 as our unifier instead of sub_1, we might needlessly limit our options.

Definition 3.32 Let $\mathbb{L}$ be a set of literals. The substitution sub is a *most general unifier* for $\mathbb{L}$ if it unifies $\mathbb{L}$ and for any other unifier sub' for $\mathbb{L}$, we have $subsub' = sub'$.

In Example 3.31, sub_1 is the most general unifier. As we pointed out, this is the best unifier for our purposes.

Proposition 3.33 A finite set of literals is unifiable if and only if it has a most general unifer.

There are two possiblities for a finite set $\mathbb{L}$ of literals, either it is unifiable or it is not. Proposition 3.33 asserts that if $\mathbb{L}$ is unifiable, then it automatically has a most general unifier. We prove this by exhibiting an algorithm that, given $\mathbb{L}$ as input, outputs "not unifiable" if no unifier exists and otherwise ouputs a most gerneal unifier for $\mathbb{L}$. The algorithm runs as follows.

The unification algorithm

Given: a finite set of literals $\mathbb{L}$.

Let $\mathbb{L}_0 = \mathbb{L}$ and $sub_0 = \emptyset$.

Suppose we know $\mathbb{L}_k$ and sub_k. If $\mathbb{L}_k$ contains just one literal, output "$sub_0 sub_1 \cdots sub_k$ is a most general unifier for $\mathbb{L}$."

Otherwise, there exist L_i and L_j in $\mathbb{L}_k$ such that the nth symbol of L_i differs from the nth symbol of L_j (for some n). Suppose n is least in this regard. If the nth symbol of L_i is a variable v and the nth symbol of L_j is the first symbol of a term t that does not contain v or vice versa (with L_i and L_j reversed) then:

Let $sub_{k+1} = (v/t)$ and $\mathbb{L}_{k+1} = \mathbb{L}_k sub_{k+1}$.

If any of the hypotheses of the previous sentence do not hold, output "$\mathbb{L}$ is not unifiable."

We must verify that this algorithm works. First we give a demonstration.

Example 3.34 Let $\mathbb{L} = \{R(f(g(x)), a, x), R(f(g(a)), a, b), R(f(y), a, z)\}$. First set $\mathbb{L}_0 = \mathbb{L}$ and $sub_0 = \emptyset$.

As we read each of the three literals in $\mathbb{L}_0$ from left to right, we see that each begins with "$R(f(\ldots$", but then there is a discrepency. Whereas the second literal continues with "$g(a)$", the third literal has "y". We check that one of these two terms is a variable and the other is a term that does not contain that variable. This is the case and so we let

$$sub_1 = (y/g(a)), \text{ and}$$
$$\mathbb{L}_1 = \mathbb{L}_0 sub_1 = \{R(f(g(x)), a, x), R(f(g(a)), a, b), R(f(g(a)), a, z)\}.$$

We note that $\mathbb{L}_1$ contains more than one literal and proceed. Now all literals begin with $R(f(g(\ldots$, but then the first literal has "x" and the second has "a".

One of these is a variable and the other is a term that does not contain that variable, and so we let

$$sub_2 = (x/a), \text{ and}$$
$$\mathbb{L}_2 = \mathbb{L}_1 sub_2 = \{R(f(g(a)), a, a), R(f(g(a)), a, b), R(f(g(a)), a, z)\}.$$

The set $\mathbb{L}_2$ still contains more than one literal, and so we continue. Each literal in $\mathbb{L}_2$ looks the same up to $R(f(g(a)), a, \ldots,$ but then the first literal has "a" and the second has "b." Neither of these is a variable, and so the algorithm concludes with output "$\mathbb{L}$ is not unifiable."

If the algorithm outputs "not unifiable," it is for one of two reasons. One is illustrated by the previous example. Here we had a discrepency between two literals that did not involve a variable. Where one literal had the constant a, the other had b. Clearly, this cannot be reconciled by a substitution and the set is, in fact, not unifiable. The other possibility is that the dicrepency involves a variable and a term, but the variable occurs in the term. For example, the set $\{P(x, y), P(x, f(y))\}$ is not unifiable. No matter what we substitute for the variables x and y, the second literal will have one more occurrence of f than the first literal. The algorithm, noting a discrepency occurs with y and $f(y)$, will terminate with "not unifiable" because the variable y occurs in the term $f(y)$. Both reasons for concluding "not unifiable" are good reasons. If the algorithm yields this output, then the set must not be unifiable.

Note that, when applied to the set $\mathbb{L}$ from Example 3.31, this algorithm outputs sub_1 as the most general unifier. So, in these examples, the algorithm works. We want to show that it always works.

If the set $\mathbb{L}$ is a finite set, then only finitely many variables occur in $\mathbb{L}$. It follows that the algorithm when applied to $\mathbb{L}$ must terminate in a finite number of steps. If it terminates with "$\mathbb{L}$ is not unifiable," then, as we have already mentioned, $\mathbb{L}$ must not be unifiable. Otherwise, the algorithm outputs "$sub_0 sub_1 \cdots sub_k$ is a most genral unifier." We must show that, when this statement is the output, it is true.

The algorithm outputs "$sub_0 sub_1 \cdots sub_k$ is the most genral unifier" only if $\mathbb{L}_k = \mathbb{L} sub_0 sub_1 \cdots sub_k$ contains just one literal. If this is the output, then $sub_0 sub_1 \cdots sub_k$ is a unifier. We must show that it is a most general unifier.

Let sub' be any other unifier for $\mathbb{L}$. We know that $sub_0 sub' = sub'$ because sub_0 is empty. Now suppose that we know $sub_0 \cdots sub_m sub' = sub'$ for some m, $0 \leq m < k$. Then $\mathbb{L}_m sub' = \mathbb{L} sub_0 \cdots sub_m sub' = \mathbb{L} sub' = \{L\}$. That is, since sub' unifies $\mathbb{L}$, it also unifies $\mathbb{L}_m$.

Suppose sub_{m+1} is (x/t). By the definition of the algorithm, t must be a term in which the variable x does not occur. Moreover, for some literals L_i and L_j in $\mathbb{L}_m$, x occurs in the nth place of L_i and t begins in the nth place of L_j

(for some n). Since sub' unifies $\mathbb{L}_m$, sub' must do the same thing to both x and t. That is, $xsub' = tsub'$. It follows that $sub_{m+1}sub' = (x/t)sub' = sub'$. By induction, we have $sub_0 \cdots sub_{m+1}sub' = sub'$ for all $m < k$. In particular, $sub_0 \cdots sub_k sub' = sub'$ and $sub_0 \cdots sub_k$ is the most general unifier for $\mathbb{L}$.

3.4.2 Resolution. We now define resolution for first-order logic. Recall that for any literal L, $\overline{L}$ is the literal defined by $\overline{L} = \neg L$ or $\neg\overline{L} = L$.

Definition 3.35 Let C_1 and C_2 be two clauses. Let s_1 and s_2 be any substitutions such that C_1s_1 and C_2s_2 have no variables in common. Let $L_1, \ldots, L_m \in C_1s_1$ and $L'_1, \ldots, L'_n \in C_2s_2$ be such that $\mathbb{L} = \{\overline{L}_1, \ldots, \overline{L}_m, L'_1, \ldots, L'_n\}$ is unifiable. Let sub be a most general unifier for $\mathbb{L}$.

Then $R = [(C_1s_1 - \{L_1, \ldots, L_m\}) \cup (C_2s_2 - \{L'_1, \ldots, L'_n\})]sub$ is a *resolvent* of C_1 and C_2.

Let φ be a sentence in SNF. Then $\varphi = \{C_1, \ldots, C_n\}$ for some clauses $C_1, \ldots, C_n$.
Let $Res(\varphi) = \{R|$ R is a resolvent of some C_i and C_j in $\varphi\}$.
Let $Res^0(\varphi) = \varphi$, and $Res^{n+1} = Res(Res^n(\varphi))$.
Let $Res^*(\varphi) = \bigcup_n Res^n(\varphi)$.

The same notation was used in propositional logic. However, unlike the propositional case, $Res^*(\varphi)$ may be an infinite set. To justify this notation and the definition of "resolvent" we need to show that $\emptyset \in Res^*(\varphi)$ if and only if φ is unsatisfiable. First we look at an example.

Example 3.36 Let $C_1 = \{Q(x, y), P(f(x), y)\}$, and $C_2 = \{R(x, c), \neg P(f(c), x), \neg P(f(y), h(z))\}$.

Suppose we want to find a resolvent of C_1 and C_2. First, we need to rename some variables since x and y occur in both C_1 and C_2. Let $s_1 = (x/u, y/v)$. Then $C_1s_1 = \{Q(u, v), P(f(u), v)\}$ which has no variables in common with C_2.

Second, note that C_1s_1 contains a literal of the form $P(\_, \_)$ and C_2 contains literals of the form $\neg P(\_, \_)$. Namely, $P(f(u), v)$ is in C_1s_1 and $\neg P(f(c), x)$ and $\neg P(f(y), h(z))$ are in C_2. Let

$$\mathbb{L} = \{P(f(u), v), P(f(c), x), P(f(y), h(z))\}.$$

By applying the unification algorithm, we see that $\mathbb{L}$ is unifiable and $sub = (u/c, y/c, v/h(z), x/h(z))$ is a most general unifier. We conclude that C_1 and C_2 have resolvent

$$\begin{aligned} R &= [(C_1s_1 - \{P(f(u), v)\}) \cup (C_2 - \{\neg P(f(c), x), \neg P(f(y), h(z))\})]sub \\ &= \{Q(u, v), R(x, c)\}sub = \{Q(c, h(z)), R(h(z), c)\}. \end{aligned}$$

We verify that the resolvent R from the previous example is in fact a consequence of C_1 and C_2. Recall that C_1 and C_2 represent sentences in SNF.

$$C_1 \text{ represents } \forall x \forall y (Q(x, y)) \vee P(f(x), y)), \text{ and}$$
$$C_2 \text{ represents } \forall x \forall y \forall z (R(x, c)) \vee \neg P(f(c), x) \vee \neg P(f(y), h(z)).$$

Suppose C_1 and C_2 hold (in some structure). Then, since these sentences are universal, they hold no matter what we plug in for the variables. In particular,

$$C_1 s_1 sub \equiv \forall z (Q(c, h(z)) \vee P(f(c), h(z))), \text{ and}$$
$$C_2 sub \equiv \forall z (R(h(z), c)) \vee \neg P(f(c), h(z))$$

both hold. That is, $C_1 s_1 sub$ is a consequence of C_1 and $C_2 sub$ is a consequence of C_2. Put another way,

$$C_1 s_1 sub \equiv \forall z (\neg Q(c, h(z)) \rightarrow P(f(c), h(z))), \text{ and}$$
$$C_2 sub \equiv \forall z (P(f(c), h(z)) \rightarrow R(h(z), c)).$$

From these two sentences, we can deduce

$$\forall z (\neg Q(c, h(z)) \rightarrow R(h(z), c))$$

which is equivalent to

$$\forall z (Q(c, h(z)) \vee R(h(z), c))$$

which is the sentence represented by R. Hence, R is a consequence of the conjunction of C_1 and C_2.

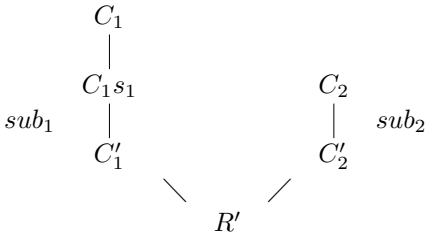
In a similar manner, we can show that any resolvent of any two clauses is necessarily a consequence of the conjunction of the two clauses. It follows that if $\emptyset \in Res^*(\varphi)$, then φ must be unsatisfiable. Conversely, suppose φ is unsatisfiable. We need to show that $\emptyset \in Res^*(\varphi)$.

At the end of the previous section we showed that φ is unsatisfiable if and only if the set $E(\varphi)$ is unsatisfiable. Recall that $E(\varphi)$ is the set of all sentences obtained by replacing each variable of φ with a term from the Herbrand universe. These sentences can be viewed as sentences of propositional logic. Suppose that C_1' and C_2' are in $E(\varphi)$ and R' is a resolvent of C_1' and C_2' in the sense of propostional logic. Then there are some clauses C_1 and C_2 of φ such that $C_1' = C_1 sub_1$ and $C_2' = C_2 sub_2$. In the following lemma we show that there exists a resolvent R of C_1 and C_2 (in the sense od first-order logic) and a substitution sub such that $Rsub = R'$. So, essentially, this lemma says that any R' that can be derived from $E(\varphi)$ using propositional resolution can also be derived from φ using first-order resolution.

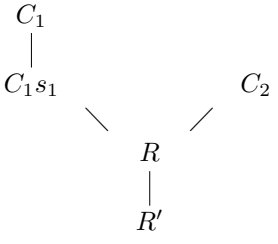
Lemma 3.37 (Lifting lemma) Let φ be a sentence in SNF. If $R' \in Res(E(\varphi))$, then there exists $R \in Res(\varphi)$ such that such that $Rsub' = R'$ for some substitution sub'.

This is called the "Lifting lemma" because we are "lifting" the resolvent R' from propositional logic to first-order logic.

Let φ be a sentence in SNF and let C_1 and C_2 be two clauses of φ. Let s_1 be a substitution such that $C_1 s_1$ and C_2 have no variables in common. Let C_1' and C_2' in $E(\varphi)$ be such that $C_1 s_1 sub_1 = C_1'$ and $C_2 sub_2 = C_2'$ for some substitutions sub_1 and sub_2. Let R' be a resolvent (in propositional logic sense) of C_1' and C_2'. This setup can be diagramed as follows:



The lemma says that if this setup holds, then there exists a resolvent R of $C_1 s_1$ and C_2 (in the sense of first-order logic) such that $Rsub' = R'$ for some substitution sub'. This conclusion can be diagramed as follows:



In the first diagram, the resolvent is taken as in propositional logic. In the second diagram, the resolvent R is as in Definition 3.35. The vertical lines in each diagram refers to a substitution. The lemma can be summarized as saying "if the first diagram holds, then so does the second diagram."

Proof of Lemma Supose the first diagram holds. Then there must exist some literal $L \in C_1'$ such that $\overline{L} \in C_2'$ and $R' = (C_1' - \{L\}) \cup (C_2' - \{\overline{L}\})$. This is the definition of resolvent for propositional logic.

Let $sub' = sub_1 sub_2$. Since $C_1 s_1$ and C_2 have no variables in common, $C_1 s_1 sub' = C_1 s_1 sub_1 = C'_1$ and $C_2 sub' = C_2 sub_2 = C'_2$.

Let $\mathbb{L}_1 = \{L_1, \ldots, L_n\}$ be the set of all L_i in $C_1 s_1$ such that $L_i sub' = L$. Likewise, let $\mathbb{L}_2 = \{L'_1, \ldots, L'_m\}$ be the set of all L'_i in C_2 such that $L'_i sub' = \overline{L}$. We have the following diagram:

$$\begin{array}{ccccccccc} & & \mathbb{L}_1 & \subset & C_1 s_1 & & C_2 & \supset & \mathbb{L}_2 & \\ sub' & & \vert & & \vert & & \vert & & \vert & sub' \\ & & L & \in & C'_1 & & C'_2 & \ni & \overline{L} & \\ & & & \diagdown & & \diagup & & & & \\ & & & & & R' & & & & \end{array}$$

Let $\mathbb{L} = \{\overline{L}_1, \ldots, \overline{L}_n, L'_1, \ldots, L'_m\}$ (that is $\mathbb{L} = \bar{\mathbb{L}}_1 \cup \mathbb{L}_2$). This set is unifiable since $\mathbb{L} sub' = \{\overline{L}\}$. Let sub be a most general unifier for $\mathbb{L}$. Then we can apply Definition 3.35 to find the following resolvent of C_1 and C_2:

$$R = [(C_1 s_1 - \mathbb{L}_1) \cup (C_2 - \mathbb{L}_2)] sub.$$

Referring to the second diagram of the lemma, we see that it remains to be shown that R' can be obtained from R by a substitution. We complete the proof of the lemma by showing that $R sub' = R'$. By applying sub' we get

$$R sub' = [(C_1 s_1 - \mathbb{L}_1) \cup (C_2 - \mathbb{L}_2)] sub sub'.$$

Since sub' is a unifier for $\mathbb{L}$ and sub is a most general unifier for $\mathbb{L}$, we know $sub sub' = sub'$. So we have

$$\begin{aligned} R sub' &= [(C_1 s_1 - \mathbb{L}_1) \cup (C_2 - \mathbb{L}_2)] sub' \\ &= (C_1 s_1 sub' - \mathbb{L}_1 sub') \cup (C_2 sub' - \mathbb{L}_2 sub') \\ &= (C'_1 - \{L\}) \cup (C'_2 - \{\overline{L}\}) = R'. \end{aligned}$$ □

Corollary 3.38 Let φ be a sentence in SNF. If $C' \in Res^*(E(\varphi))$, then there exists $C \in Res^*(\varphi)$ and a substitution sub' such that $C sub' = C'$.

Proof If $C' \in Res^*(E(\varphi))$, then $C' \in Res^n(E(\varphi))$ for some n. We prove the corollary by induction on n. If $n = 0$, then $C' \in E(\varphi)$. Then, by the definition of $E(\varphi)$, C' is obtained by substituting variable free terms in for the variables of some $C \in \varphi$.

For the induction step, we utilize the Lifting lemma. Suppose that for some m, each clause of $Res^m(E(\varphi))$ is obtained from some clause of $Res^*(\varphi)$ via substitution. Let $\tilde{\varphi} \subset Res^*(\varphi)$ be such that every clause of $Res^m(E(\varphi))$ comes from some clause in $\tilde{\varphi}$. Then $Res^m(E(\varphi)) \subset E(\tilde{\varphi})$. If $C' \in Res^{m+1}(E(\varphi))$, then $C' \in Res(E(\tilde{\varphi}))$. By the Lifting lemma, there is some $C \in Res(\tilde{\varphi})$ such that $C sub' = C'$ for some substitution sub'. Since $\tilde{\varphi} \subset Res^*(\varphi)$, $C \in Res^*(\varphi)$. □

In particular, if $\emptyset \in Res^*(E(\varphi))$, then there exists some $C \in Res^*(\varphi)$ such that $Csub' = \emptyset$ for some substitution sub'. But this is only possible if $C = \emptyset$. So if $\emptyset \in Res^*(E(\varphi))$, then $\emptyset \in Res^*(\varphi)$. We conclude that if φ is unsatisfiable, then $\emptyset \in Res^*(\varphi)$. We have shown that the notion of resolution defined in this section works. We state this as a theorem.

Theorem 3.39 Let φ be a sentence in SNF. Then φ is unsatisfiable if and only if $\emptyset \in Res^*(\varphi)$.

3.5 SLD-resolution

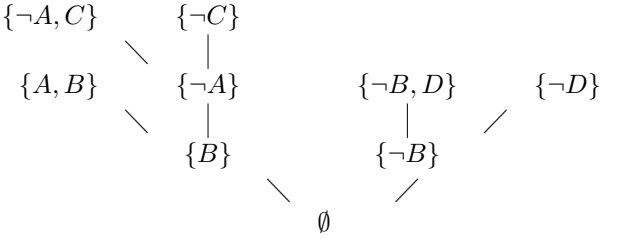
One purpose of resolution is to provide a method of proof that can be done by a computer. Toward this aim, we refine resolution in this section. Our goal is to find a version of resolution that can be completely automated. The advantage of resolution over other formal proof systems is that it rests on a single rule. Resolution proofs may not be the most succinct. They will not lend insight as to why, say, a sentence φ is unsatisfiable. The benefit of resolution is precisely that it does not require any insight. To show that φ is unsatisfiable, we can blindly compute $Res^*(\varphi)$ until we find $\emptyset$. However, this method is not practical. If $\emptyset$ is in $Res^*(\varphi)$, then calculating the clauses in $Res^*(\varphi)$ one-by-one in no particular order is not an efficient way of finding it. The first two theorems of this section show that it is not necesssary to compute all of $Res^*(\varphi)$. We show that we only need to compute resolvents R of clauses C_1 and C_2 that have certain forms. We refer to C_1 and C_2 as the *parents* of R.

Definition 3.40 *N-resolution* requires that one parent contain only negative literals.

We look at an example from propositional logic. Let

$$\varphi = \{\{A, B\}, \{\neg A, C\}, \{\neg B, D\}, \{\neg C\}, \{\neg D\}\}.$$

We show that φ is unsatisfiable using N-resolution.



Note that each resolvent has a parent that contains only negative literals.