19.4. Fill in the missing values for the clauses $C_1$ or $C_2$ (or both) in the following sets of clauses, given that $C$ is the resolvent of $C_1$ and $C_2$. If there is more than one possible solution, provide one example of each different kind.

a. $C = \text{True} \Rightarrow P(A, B), C_1 = P(x, y) \Rightarrow Q(x, y), C_2 =$??.
   There is no possible value for $C_2$ here. The resolution step would have to resolve away both the $P(x, y)$ on the LHS of $C_1$ and the $Q(x, y)$ on the right, which is not possible.\(^1\)

b. $C = \text{True} \Rightarrow P(A, B), C_1 =$??, $C_2 =$??.
   Without loss of generality, let $C_1$ contain the negative (LHS) literal to be resolved away. The LHS of $C_1$ therefore contains one literal $l$, while the LHS of $C_2$ must be empty. The RHS of $C_2$ must contain $l'$ such that $l$ and $l'$ unify with some unifier $\theta$. Now we have a choice: $P(A, B)$ on the RHS of $C$ could come from the RHS of $C_1$ or of $C_2$. Thus the two basic solution templates are,

$$
C_1 = l \Rightarrow \text{False} \quad ; \quad C_2 = \text{True} \Rightarrow l' \lor P(A, B)^{-1} \\
C_1 = l \Rightarrow P(A, B)^{-1} \quad ; \quad C_2 = \text{True} \Rightarrow l'
$$

where $\theta^{-1}$ is the **inverse substitution** of $\theta$. Within these templates, the choice of $l$ is entirely unconstrained. Suppose $l$ is $Q(x, y)$ and $l'$ is $Q(A, B)$, then $P(A, B)^{-1}$ could be $P(x, y)$ (or $P(A, y)$ or $P(x, B)$) and the solutions are

$$
C_1 = Q(x, y) \Rightarrow \text{False} \quad ; \quad C_2 = \text{True} \Rightarrow Q(A, B) \lor P(x, y) \\
C_1 = Q(x, y) \Rightarrow P(x, y) \quad ; \quad C_2 = \text{True} \Rightarrow Q(A, B)
$$

\(^1\)Technically resolution can remove more than one literal from a clause, but only if those literals are redundant—i.e. one subsumes the other.
Suppose that \( F \) is a binary predicate and \( F \) has five different variables.

a. How many functionally different literals can be generated? Two literals are functionally identical if they differ only in the names of the new variables that they contain.

b. Can you find a general formula for the number of different literals with a predicate of arity \( r \)?

c. \( C = P(x, y) \Rightarrow P(x, f(y)), C_1 = ??, C_2 = ?? \).

As before, let \( C_1 \) contain the negative (LHS) literal to be resolved away, with \( l' \) on the RHS of \( C_2 \). We now have four possible templates because each of the two literals in \( C \) could have come from either \( C_1 \) or \( C_2 \):

\[
C_1 = l \Rightarrow \text{False} \quad ; \quad C_2 = P(x, y) \theta^{-1} \Rightarrow l' \lor P(x, f(y)) \theta^{-1}
\]

\[
C_1 = l \Rightarrow P(x, f(y)) \theta^{-1} \quad ; \quad C_2 = P(x, y) \theta^{-1} \Rightarrow l'
\]

\[
C_1 = l \land P(x, y) \theta^{-1} \Rightarrow \text{False} \quad ; \quad C_2 = \text{True} \Rightarrow l' \lor P(x, f(y)) \theta^{-1}
\]

\[
C_1 = l \land P(x, y) \theta^{-1} \Rightarrow P(x, f(y)) \theta^{-1} \quad ; \quad C_2 = \text{True} \Rightarrow l'
\]

Again, we have a fairly free choice for \( l \). However, since \( C \) contains \( x \) and \( y \), \( \theta \) cannot bind those variables (else they would not appear in \( C \)). Thus, if \( l \) is \( Q(x, y) \), then \( l' \) must be \( Q(x, y) \) also and \( \theta \) will be empty.

19.6. Suppose that FOIL is considering adding a literal to a clause using a binary predicate \( P \) and that previous literals (including the head of the clause) contain five different variables.

a. How many functionally different literals can be generated? Two literals are functionally identical if they differ only in the names of the new variables that they contain.

It is important to note that position is significant. That is, \( P(A, B) \) is very different from \( P(B, A) \). The first argument position can contain one of the five existing variables or a new variable. For each of these six choices, the second position can contain one of the five existing variables or a new variable, except that the literal with two new variables is disallowed. Hence there are \( 5 \cdot 5 = 25 \) literals with no new variables, \( 5 + 5 = 10 \) literals with exactly one new variable, giving a total of 35 choices. With negated literals also included, the total branching factor is 70.

b. Can you find a general formula for the number of different literals with a predicate of arity \( r \) when there are \( n \) variables previously used?

This seems to be quite a tricky combinatorial problem. One way to solve it seems to be to start by including the multiple possibilities that are equivalent under renaming of the new variables as well as those that contain only new variables. Then these redundant or illegal choices can be removed later. Now, we can use up to \( r - 1 \) new variables. If we use \( i \) or fewer variables, we can write \( (n + i)^r \) literals, so using exactly \( i > 0 \) variables we can write \( (n + i)^r - (n + i - 1)^r \) literals. Each of these is functionally isomorphic under any renaming of the new variables. With \( i \) variables, there are \( i \) renamings. Hence the total number of distinct literals (including those illegal ones with no old variables) is

\[
n^r + \sum_{i=1}^{r-1} \frac{(n + 1)^r - (n + i - 1)^r}{i!}
\]

Now we just subtract off the number of distinct all-new literals. With \( i \) or fewer new variables, the number of (not necessarily distinct) all-new literals is \( i^r \), so the number with exactly \( i > 0 \) is \( i^r - (i - 1)^r \). Each of these has \( i! \) equivalent literals in the set. This gives us the final total for distinct, legal literals:

\[
n^r + \sum_{i=1}^{r-1} \frac{(n + 1)^r - (n + i - 1)^r}{i!} - \sum_{i=1}^{r-1} \frac{i^r - (i - 1)^r}{i!}
\]

which can doubtless be simplified. One can check that for \( r = 2 \) and \( n = 5 \) this gives 35.
c. Why does Foil not allow literals that contain no previously used variables?

If a literal contains only new variables, then either a subsequent literal in the clause body connects one or more of those variables to one or more of the “old” variables, or it doesn’t. If it does, then the same clause will be generated with those two literals reversed, such that the restriction is not violated. If it doesn’t, then the literal is either always true (if the predicate is satisfiable) or always false (if it is unsatisfiable), independent of the “input” variables in the head. Thus, the literal would either be redundant or would render the clause body equivalent to False.

A. Suppose that your hypothesis space is the set of all first-order logic formulas. That is, your learning algorithm may output any first-order formula as a hypothesis \( H \). Given a set of training examples,

\[
L = \{ Q(A,B), Q(B,A), \neg Q(A,A), \neg Q(B,B) \}
\]

with no background knowledge \( B \) or descriptions \( D \), what is the most specific hypothesis \( H \) for this learning problem? That is, what is the most specific hypothesis such that,

\[
H \models L
\]

The most distinct \( H \) is the following:

\[
\{ [(x = A) \land (y = B)] \lor [(x = B) \land (y = A)] \} \iff Q(x,y)
\]

which will only assert that \( Q \) is true for \( Q(A,B) \) and \( Q(B,A) \). For all other arguments \( Q \) will be false.

I would also accept the answer \( Q(A,B) \lor Q(B,A) \) though this does not correspond to the syntactic form used for hypotheses in the book or notes.

B.1 Denote by \( \text{VS}(HSE) \) the version space with respect to hypothesis space \( HS \) for the training set \( E \).

Consider two hypothesis spaces \( HS_1 \) and \( HS_2 \) and a set of training data \( E \).

Is it true in general that \( \text{VS}(HS_1 \cup HS_2, E) = \text{VS}(HS_1, E) \cup \text{VS}(HS_2, E) \)?

If so, give a proof; otherwise, provide a counter example.

Yes this is true. The proof is straightforward.

\[
h \text{ is an element of } \text{VS}(HS_1 \cup HS_2, E) \iff h \text{ is consistent with } E \text{ and is either in } HS_1 \text{ or in } HS_2
\]

\[
h \text{ is either in } \text{VS}(HS_1, E) \text{ or } \text{VS}(HS_2, E) \iff h \text{ is in } \text{VS}(HS_1, E) \cup \text{VS}(HS_2, E)
\]

This shows that the two sets above are equivalent.
B.2 Given two hypothesis spaces $HS_1$ and $HS_2$ we denote the conjunction of the spaces by,

$$[HS_1 \land HS_2] = \{ h_1 \land h_2 \mid h_1 \in HS_1 \text{ and } h_2 \in HS_2 \}$$

that is, $[HS_1 \land HS_2]$ is the set of all hypotheses that can be formed by conjoining a hypothesis from $HS_1$ with a hypothesis from $HS_2$. (The hypothesis $h_1 \land h_2$ says "positive" iff $h_1$ and $h_2$ both say "positive".)

Is it true that $VS([HS_1 \land HS_2], E) = [VS(HS_1, E) \land VS(HS_2, E)]$?

If so, give a proof; otherwise, provide a counter example.

This is not true in general.

Consider a learning problem where there are only three training examples:

$$E = \{ (x_1, -), (x_2, +), (x_3, -) \}$$

We will consider two hypotheses $h_1$ and $h_2$ that make the following predictions on the examples:

$$
\begin{align*}
  h_1(x_1) &= + & h_2(x_1) &= - \\
  h_1(x_2) &= + & h_2(x_2) &= + \\
  h_1(x_3) &= - & h_2(x_3) &= +
\end{align*}
$$

Let $HS_1 = \{h1\}$ and $HS_2 = \{h2\}$. Clearly we have that both

$$VS(HS_1, E) = VS(HS_2, E) = \{\}$$

That is the version spaces are empty since neither $h_1$ or $h_2$ are consistent with the examples. This means that the RHS of the above expression

$$[VS(HS_1, E) \land VS(HS_2, E)] = \{\}$$

But we have that $[HS_1 \land HS_2] = \{h_1 \land h_2\}$, and the hypothesis $h_1 \land h_2$ is consistent with the training data, i.e.

$$
\begin{align*}
  h_1 \land h_2(x_1) &= - \\
  h_1 \land h_2(x_2) &= + \\
  h_1 \land h_2(x_3) &= -
\end{align*}
$$

Thus, $VS([HS_1 \land HS_2], E) = \{h_1 \land h_2\}$ which is the LHS of the above expressions. Since the LHS does not equal the RHS we have successfully created a counter example.