1. (a) Program

\[ \forall X. n(e, X) \]
\[ \forall X, Y. n(X, Y) \rightarrow n(f(X), g(Y)) \]
\[ \forall X, Y. n(X, Y) \rightarrow n(g(X), f(Y)) \]

Query:

\[ \exists X. n(X, f(g(e))) \land n(X, f(f(e))) \]

(b) The full search tree is:

Thus Prolog returns \( X = e \) as the first solution, and \( X = g(e) \) as the second solution.

2. For convenience, we name the three substitutions:

\[ \theta_1 = \{ X = f(Y, Y) \} \]
\[ \theta_2 = \{ X = f(g(Z), W) \} \]
\[ \theta_3 = \{ X = f(g(Z), g(Z)) \} \]

\( \theta_1 \) is more general than \( \theta_3 \) because \( \theta_3 = \theta_1 \{ Y = g(Z) \} \).

\( \theta_2 \) is more general than \( \theta_3 \) because \( \theta_3 = \theta_2 \{ W = g(Z) \} \).

Neither of \( \theta_1 \) and \( \theta_2 \) is more general than the other. For example, \( X = f(a, a) \) is an instance of \( \theta_1 \) but not of \( \theta_2 \), and \( X = f(g(a), a) \) is an instance of \( \theta_2 \) but not of \( \theta_1 \).

(In this question and solution, I have not been careful to respect the technical restriction that terms being substituted contain only “fresh” variables. This annoying technical restriction is discussed in Theory Lecture 4. It is better not worried about too much in practice.)
3. (a) A most general unifier is \( \{ X = g(W), U = h(Z) \} \). Another (not most
general) unifier is \( \{ X = g(a), Z = a, W = a, U = h(a) \} \).

(b) A most general unifier is \( \{ X = g(W), Z = a \} \). Another (not most
general) unifier is \( \{ X = g(a), Z = a, W = a \} \).

Most general unifiers are *unique* in the sense that if \( \theta \) and \( \theta' \) are both
most general unifiers then the substitutions \( \theta \) and \( \theta' \) are equivalent in the
sense that each of \( \theta \) and \( \theta' \) is more general than the other (i.e., \( \theta \not\leq \theta' \) and
\( \theta' \not\leq \theta \), using the notation of Theory Lecture 4).

4. Definite clauses and Prolog program clauses are in one-to-one correspon-
dence by associating the definite clause \( D \) below

\[ \neg A_1 \lor \cdots \lor \neg A_k \lor B \]

with the Prolog clause \( F_D \)

\[ A_1 \land \cdots \land A_k \rightarrow B \]

Note that \( D \) and \( F_D \) are (logically) equivalent formulas. Note also that
facts are included in this correspondence by taking \( k = 0 \).

Goal clauses are in one-to-one correspondence with Prolog goals by associ-
ating the goal clause \( G \) below

\[ \neg E_1 \lor \cdots \lor \neg E_m \]

with the Prolog goal \( H_G \):

\[ E_1, \ldots, E_m \]

Note that the formula \( G \) is equivalent to \( \neg \bigwedge H_G \), where by \( \bigwedge H_G \) we mean
the formula \( E_1 \land \cdots \land E_m \).

We thus consider a set of clauses

\[ G, D_1, \ldots, D_N \]

as corresponding the the Prolog program

\[ F_{D_1}, \ldots, F_{D_N} \]

and goal \( H_G \).

The unsatisfiability of the conjunction of the universal quantifications of
the clauses

\[ G, D_1, \ldots, D_N \]

amounts to, as in (ii) in the question, the logical consequence

\[ \forall Vars(D_1). D_1, \ldots, \forall Vars(D_N). D_N, \forall Vars(G). G \models \text{false} \]

Exploiting the equivalences between \( D_i \) and \( F_{D_i} \), and between \( G \) and
\( \neg \bigwedge H_G \), both noted earlier, we obtain that the above logical consequence
is equivalent to

\[ \forall Vars(F_{D_1}). F_{D_1}, \ldots, \forall Vars(F_{D_N}). F_{D_N}, \forall Vars(H_G). \neg \bigwedge H_G \models \text{false} \]
But $\forall \neg$ is equivalent to $\neg \exists$, so the logical consequence is equivalent to

$$\forall \text{Vars}(F_{D_1}), \ldots, \forall \text{Vars}(F_{D_N}), \neg \exists \text{Vars}(H_G). \bigwedge H_G \models \text{false}$$

which is, in turn, equivalent to

$$\forall \text{Vars}(F_{D_1}), \ldots, \forall \text{Vars}(F_{D_N}). \bigwedge F_{D_1}, \ldots, \bigwedge \exists \text{Vars}(H_G). \bigwedge H_G$$

which indeed captures Prolog’s objective in answering a query.

We now establish a one-to-one correspondence between:

(i) SLD resolution refutations from the set of clauses

$$G, D_1, \ldots, D_N$$

and

(ii) proofs of the goal $H_G$ from the program $F_{D_1}, \ldots, F_{D_N}$ in the inference system of Lecture 5.

An instance of the resolution rule in the SLD refutation must take the form

$$\neg A_1 \lor \ldots \lor \neg A_k \lor B \quad \neg E_1' \lor \ldots \lor \neg E_m'$$

$$\frac{(E_1' \lor \ldots \lor \neg E_{l-1}' \lor \neg A_1 \lor \ldots \lor \neg A_k \lor E_{l+1}' \lor \ldots \lor \neg E_m')\theta}{A_1 \lor \ldots \lor A_k \rightarrow B}$$

for some $l \leq m$, where $A_1 \lor \ldots \lor \neg A_k \lor B$ is (a variable renaming of) one of the definite clauses $D_i$ and $\theta$ is the mgu of $B$ and $E_i'$.

This rule is ‘inverted’ as follows in the Lecture-5 derivation:

$$\frac{(E_1', \ldots, E_{l-1}', A_1, \ldots, A_k, E_{l+1}', \ldots, E_m')\theta}{E_1', \ldots, E_m'} A_1 \land \ldots \land A_k \rightarrow B$$

Note that $A_1 \land \ldots \land A_k \rightarrow B$ is (a variable renaming of) one of the program clauses $F_{D_i}$.

Then the entire SLD refutation tree, which necessarily has the shape of a spine with the unique leaf labelled with the goal clause $G$ at its top, and which ends with the empty clause $\bot$ at its bottom (root), gets turned upside down into a Lecture-5 derivation with (as is required) the empty goal at the top, and the goal $H_G$ at the bottom.

The entire solution is best illustrated by an example. Consider as the initial clauses:

$$\neg \text{num}(s(s(X))), \text{num}(z), \neg \text{num}(X) \lor \text{num}(s(X))$$

This corresponds to the Prolog program (in logical notation)

$$\text{num}(z)$$

$$\text{num}(X) \rightarrow \text{num}(s(X))$$

and query

$$\text{num}(s(s(X)))$$
In this case, we have the following SLD-resolution refutation

\[
\begin{array}{c}
\neg \text{num}(Y) \lor \text{num}(s(Y)) \\
\text{num}(z) \\
\rightarrow \\
\neg \text{num}(X)
\end{array}
\]

which corresponds to the derivation:

\[
\begin{array}{c}
\epsilon \\
\text{num}(z) \\
\rightarrow \\
\text{num}(X) \\
\rightarrow \\
\text{num}(s(Y)) \\
\rightarrow \\
\text{num}(s(s(X)))
\end{array}
\]

5. (a) **Technical lemma 1.** If the goals \(G_1, \ldots, G_m\) and \(G'_1, \ldots, G'_n\) both have derivations, and have no variables in common, then the amalgamated goal \(G_1, \ldots, G_m, G'_1, \ldots, G'_n\) also has a derivation.

**Proof.** We use \(\Pi\) and \(\Pi'\) to represent the derivations of \(G_1, \ldots, G_m\) and \(G'_1, \ldots, G'_n\) respectively. These may be pictured as:

\[
\begin{array}{c}
\epsilon \\
\Pi \\
G_1, \ldots, G_m
\end{array}
\]

\[
\begin{array}{c}
\epsilon \\
\Pi' \\
G'_1, \ldots, G'_n
\end{array}
\]

where \(\Pi\) and \(\Pi'\) represent the sequence of rule applications, including all intermediate goals and all program clauses used. Also, because the variables in the two goals \(G_1, \ldots, G_m\) and \(G'_1, \ldots, G'_n\) are distinct, we can ensure that all variables in \(\Pi\) and \(\Pi'\) are distinct.

Then a derivation for the amalgamated goal \(G_1, \ldots, G_m, G'_1, \ldots, G'_n\) is given by

\[
\begin{array}{c}
\epsilon \\
\Pi' \\
G'_1, \ldots, G'_n
\end{array}
\]

\[
\begin{array}{c}
\Pi[G'_1, \ldots, G'_n] \\
G_1, \ldots, G_m, G'_1, \ldots, G'_n
\end{array}
\]

where \(\Pi[G'_1, \ldots, G'_n]\) is the tree of rule applications obtained by appending \(G'_1, \ldots, G'_n\) to the end of every goal in \(\Pi\); just as \(G'_1, \ldots, G'_n\) is replacing the empty goal \(\epsilon\) at the top of \(\Pi\). It remains to check that the steps in \(\Pi[G'_1, \ldots, G'_n]\) are indeed *bona fide* rule applications. But this is so for the following reason. Each rule application in \(\Pi\)

\[
\begin{array}{c}
(C_1, \ldots, C_{l-1}, A_1, \ldots, A_k, C_{l+1}, \ldots, C_j)\theta \\
\rightarrow \\
A_1 \land \cdots \land A_k \rightarrow B
\end{array}
\]

\[
\begin{array}{c}
C_1, \ldots, C_j
\end{array}
\]
gets transformed to
\[
(C_1, \ldots, C_{l-1}, A_1, \ldots, A_k, C_{l+1}, \ldots, C_j)\theta, G'_1, \ldots, G'_n \quad A_1 \land \cdots \land A_k \rightarrow B
\]
\[
C_1, \ldots, C_j, G'_1, \ldots, G'_n
\]
This is itself indeed a rule application because variables in \(G'_1, \ldots, G'_n\) are distinct from variables in \(C_1, \ldots, C_j\) and \(A_1 \land \cdots \land A_k \rightarrow B\), hence
\[
(C_1, \ldots, C_{l-1}, A_1, \ldots, A_k, C_{l+1}, \ldots, C_j)\theta, G'_1, \ldots, G'_n
\]
is the same as
\[
(C_1, \ldots, C_{l-1}, A_1, \ldots, A_k, C_{l+1}, \ldots, C_j, G'_1, \ldots, G'_n)\theta
\]
This completes the proof.

From this point on the material gets increasingly technical, and is included purely for the benefit of mathematically inquisitive students. Your coverage of the course material will not suffer if you ignore the remainder of this solution.

**Technical lemma 2.** If the substituted goal \((G_1, \ldots, G_m)\theta\) has a derivation then so does \(G_1, \ldots, G_m\).

**Proof.** We prove this by induction on the height of the derivations that: for every \(n \geq 0\), for every derivation of height \(n\) of a substitution instance of a goal, there exists a derivation of height \(n\) of the unsubstituted goal. Here, by *height* we mean the number of non-empty goals appearing in the derivation. (It is perhaps worth remarking that a formalized version of the proof of technical lemma 1 above also proceeds by induction on the height of the derivation.)

If the height of the derivation of \((G_1, \ldots, G_m)\theta\) is 0, then \((G_1, \ldots, G_m)\theta\) is the empty goal \(\epsilon\), hence \(G_1, \ldots, G_m\) is empty and thus trivially derivable.

If the height of the derivation is \(n > 0\), then the last rule in the derivation has the form
\[
(G_1\theta, \ldots, G_{l-1}\theta, A_1, \ldots, A_k, G_{l+1}\theta, \ldots, G_m\theta)\theta' \quad A_1 \land \cdots \land A_k \rightarrow B
\]
where \(\theta'\) is the mgu of \(G_l\theta\) and \(B\), and where the variables in \(A_1 \land \cdots \land A_k \rightarrow B\) can be chosen be distinct from those in the domain of \(\theta\) as well as from those in \(G_1\theta, \ldots, G_m\theta\). By the distinctness of the variables, we have \(A_i\theta = A_i\), for every \(A_i\). Similarly \(B\theta = B\). Thus the rule application above can be rewritten as:
\[
(G_1, \ldots, G_{l-1}, A_1, \ldots, A_k, G_{l+1}, \ldots, G_m)\theta' \quad A_1 \land \cdots \land A_k \rightarrow B
\]
Since \(B\theta = B\), we have \(B\theta\theta' = B\theta' = G_l\theta\theta'\), with the last equality holding because \(\theta'\) unifies \(G_l\theta\) and \(B\). This shows that \(\theta\theta'\) is a unifier.
of $G_l$ and $B$. Let $\psi$ be the most general unifier. Since it is most general, we have that $\theta\theta' = \psi\psi'$ for some substitution $\psi'$. Thus the rule application above can again be rewritten as:

$$
\frac{(G_1, \ldots, G_{l-1}, A_1, \ldots, A_k, G_{l+1}, \ldots, G_m)\psi\psi'}{(G_1, \ldots, G_m)\theta}
$$

where, since the entire derivation has height $n$, the subderivation of the goal

$$(G_1, \ldots, G_{l-1}, A_1, \ldots, A_k, G_{l+1}, \ldots, G_m)\psi\psi'$$

has height $n - 1$. We view this goal as a substitution instance (via $\psi'$) of the goal

$$(G_1, \ldots, G_{l-1}, A_1, \ldots, A_k, G_{l+1}, \ldots, G_m)\psi$$

By the induction hypothesis, this latter goal itself has a derivation of height $n - 1$. We can then follow this derivation with the rule instance:

$$
\frac{(G_1, \ldots, G_{l-1}, A_1, \ldots, A_k, G_{l+1}, \ldots, G_m)\psi}{A_1 \land \cdots \land A_k \rightarrow B}
$$

which is legitimate because $\psi$ is the mgu of $G_l$ and $B$. Observe, finally, that we have now produced the required height-$n$ derivation of the goal $G_1, \ldots, G_m$.

(b) We content ourselves with formulating the correct results. The proofs are generalizations of the proofs above, adapted to the more flexible structure (general binary trees rather than spines) of resolution refutations.

**Technical lemma 1.** If each of the two sets of clauses below has a resolution refutation

$$
C_1, \ldots, C_k, L_1 \lor \cdots \lor L_m
$$

and if $L_1 \lor \cdots \lor L_m$ and $L'_1 \lor \cdots \lor L'_n$ have no variable in common, then the set of clauses below has a resolution refutation.

$$
C_1, \ldots, C_k, L_1 \lor \cdots \lor L_m \lor L'_1 \lor \cdots \lor L'_n
$$

**Technical lemma 2.** If the set of clauses below has a resolution refutation

$$
C_1, \ldots, C_k, C\theta
$$

then so does

$$
C_1, \ldots, C_k, C
$$