**Problem 1:** Let $(\mathbb{Q}, <)$ be the structure with domain the set of rational numbers and the binary predicate $<$ interpreted as the natural order on rationals.

(a) Write $(\forall x)(\exists y)(y < x)$ as a simple English sentence.

(b) An ordered set is called dense if between any two distinct members of the set there is a third. Write the proposition “The set of rational numbers is dense” in predicate logic.

**Solution:**

(a) “There is no least rational number.”

(b) $(\forall x)(\forall y)(x < y \rightarrow (\exists z)(x < z \land z < y))$.

**Problem 2:** Give a resolution proof of the following propositional formulas (first convert their negation into clausal form).

(a) $((P \rightarrow Q) \land (P \rightarrow (Q \rightarrow R))) \rightarrow (P \rightarrow R)$

(b) $((P \land Q) \lor (R \land \neg Q)) \rightarrow ((Q \rightarrow P) \land (\neg Q \rightarrow R))$

**Solution:**

(a) We convert the negation to clausal form:

$$\neg ((P \rightarrow Q) \land (P \rightarrow (Q \rightarrow R))) \rightarrow (P \rightarrow R)$$

$$\iff (P \rightarrow Q) \land (P \rightarrow (Q \rightarrow R)) \land \neg (P \rightarrow R)$$

$$\iff (P \lor Q) \land (P \lor \neg R \lor \neg Q) \land P \land \neg R$$

And in clausal form:

$$\{\neg P, Q\}, \{\neg P, \neg Q, R\}, \{P\}, \{\neg R\}$$

A possible resolution:

```
\{\neg P, \neg Q, R\} \rightarrow \{P\} \rightarrow \{\neg R\} \rightarrow \{\neg P, R\} \rightarrow \{R\} \rightarrow \Box
```
(b) We convert the negation to clausal form:

\[- ((P \land Q) \lor (R \land \neg Q)) \rightarrow ((Q \rightarrow P) \land (\neg Q \rightarrow R))\]

\[\iff ((P \land Q) \lor (R \land \neg Q)) \land \neg((Q \rightarrow P) \land (\neg Q \rightarrow R))\]

\[\iff ((P \land Q) \lor (R \land \neg Q)) \land ((Q \land \neg P) \lor (\neg Q \land \neg R))\]

\[\iff (P \lor R) \land (P \lor \neg Q) \land (Q \lor R) \land (Q \lor \neg Q) \land (Q \lor \neg R) \land (\neg P \lor \neg Q) \land (\neg P \lor \neg R)\]

And in clausal form:

\[
\{\{P, R\}, \{P, \neg Q\}, \{Q, R\}, \{Q, \neg Q\}, \{Q, \neg R\}, \{\neg P, \neg Q\}, \{\neg P, \neg R\}\}
\]

A possible resolution:

```
{Q, R}                     \{Q, \neg R\}                     \{P, \neg Q\}                     \{\neg P, \neg Q\}
                   \{Q\}                        \{\neg Q\}                        \[\square\]
```

**Problem 3:** For each of the following formulas, first construct a tableaux proof, then give a resolution proof.

(a) \((\exists x)(\forall y)P(x, y) \rightarrow (\forall x)(\exists y)P(y, x)\)

(b) \((\forall x)(P(x, f(x)) \lor Q(g(x), x)) \rightarrow (\forall x)(\exists y)(P(x, y) \lor Q(y, x))\).

**Solution:**

(a) \(F ((\exists x)(\forall y)P(x, y) \rightarrow (\forall x)(\exists y)P(y, x))\)

```
F ((\exists x)(\forall y)P(x, y) \rightarrow (\forall x)(\exists y)P(y, x))
T (\exists x)(\forall y)P(x, y)
F (\forall x)(\exists y)P(y, x)
T (\forall y)P(c, y)
F (\exists y)P(y, d)
T P(c, d)
F P(c, d)
\[\otimes\]
```

2
For a resolution, first convert the negation into clausal form (Skolemizing where necessary):

\[\neg ((\exists x)(\forall y)P(x, y) \rightarrow (\forall x)(\exists y)P(y, x))\]

\[\iff ((\exists x)(\forall y)P(x, y)) \land -((\forall x)(\exists y)P(y, x))\]

\[\iff ((\exists x)(\forall y)P(x, y)) \land (\exists x)(\forall y)-P(y, x)\]

\[\iff ((\forall y)P(c, y)) \land (\forall y)-P(y, d)\]

\[\iff (\forall y)(\forall w)(P(c, y) \land -P(w, d))\]

In clausal form:

\[{\{P(c, y)\}, \{-P(w, d)\}}\]

The literals \(P(c, y)\) and \(P(w, d)\) are unified by the substitution \(\{y/d, w/c\}\) and immediately give the empty clause as a resolvent.

(b) \(F (\forall x)(P(x, f(x)) \lor Q(g(x), x)) \rightarrow (\forall x)(\exists y)P(x, y) \lor Q(y, x))\).

\(T (\forall x)(P(x, f(x)) \lor Q(g(x), x))\)

\(F (\forall x)(\exists y)P(x, y) \lor Q(y, x))\)

\(F (\exists y)P(c, y) \lor Q(y, c))\)

\(T (P(c, f(c)) \lor Q(g(c), c))\)

\(F (P(c, f(c)) \lor Q(f(c), c))\)

\(F (P(c, g(c)) \lor Q(g(c), c))\)

\(F P(c, f(c))\)

\(F P(c, g(c))\)

\(F Q(f(c), c)\)

\(F Q(g(c), c)\)

\(\otimes\)

\(\otimes\)

The resolution part:

\[\neg ((\forall x)(P(x, f(x)) \lor Q(g(x), x)) \rightarrow (\forall x)(\exists y)(P(x, y) \lor Q(y, x)))\]

\[\iff (\forall x)(P(x, f(x)) \lor Q(g(x), x)) \land -((\forall x)(\exists y)(P(x, y) \lor Q(y, x)))\]

\[\iff (\forall x)(P(x, f(x)) \lor Q(g(x), x)) \land (\exists x)(\forall y)((\neg P(x, y)) \land (\neg Q(y, x)))\]

\[\iff (\forall x)(\forall y)((P(x, f(x)) \lor Q(g(x), x)) \land (\forall y)((\neg P(c, y)) \land (\neg Q(y, c))))\]

\[\iff (\forall x)(\forall y)((P(x, f(x)) \lor Q(g(x), x)) \land (\neg P(c, y)) \land (\neg Q(y, c)))\]

In clausal form:

\[{\{P(x, f(x)), Q(g(x), x)\}, \{-P(c, y)\}, \{-Q(y, c)\}}\]

Resolution:

\[{P(x, f(x)), Q(g(x), x)}\]

\[\{-Q(y, c)\}\]

\[{x/c, y/f(c)}\]

\[{y/g(c)}\]

\[\{-P(c, y)\}\]

\[\{Q(g(c), c)\}\]

\[\Box\]
Problem 4: Can the following sets of terms be unified? If there is a most general unifier, find it, otherwise explain why the terms cannot be unified. (a is a constant, x, y, z, w are variables, f, g, h are function symbols.)

(a) \{P(f(x), g(f(w)), x), P(y, g(z), g(a))\}
(b) \{f(x, g(a)), f(h(y), z), f(w, w)\}
(c) \{Q(x, f(x)), Q(g(y), y)\}

Solution:

(a) The unification algorithm succeeds here, and produces the mgu \{x/g(a), y/f(g(a)), z/f(w)\}.

(b) The unification algorithm (in the first two stages) defines the substitution \{x/h(y), w/h(y)\}, and then tries to unify \{f(h(y), g(a)), f(h(y), z), f(h(y), h(y))\}. This fails, since the terms \(g(a)\) and \(h(y)\) are not unifiable. Therefore the original set is not unifiable.

(c) The unification algorithm first defines the substitution \{x/g(y)\}, and then tries to unify \{Q(g(y), f(g(y))), Q(g(y), y)\}. This fails, since the disagreement set is \{f(g(y)), y\} and \(y\) appears in \(f(g(y))\). Therefore the original set is not unifiable.

Problem 5: Labeled binary trees are binary trees of which all leaves have labels. We represent a labeled leaf as \texttt{leaf(label)}, and a tree as \texttt{tree(left subtree, right subtree)} (recursively).

(a) Define a binary predicate \texttt{mirror} which succeeds when given two trees which are mirror images of one another. Invoking the predicate with a variable as a second argument should produce a mirror image of the tree in the first argument.

(b) Define a unary predicate \texttt{look_for_elephant} which searches for an elephant on a given tree.

Solution:

(a) The predicate \texttt{mirror} is defined recursively. For the base case, a leaf its own mirror image. To produce the mirror image of a complex tree, mirror each of the two main subtrees, and switch their location:
\texttt{mirror(leaf(X), leaf(X)).}
\texttt{mirror(tree(X,Y), tree(Y1,X1)) :- mirror(X,X1), mirror(Y,Y1).}

(b) Again, a recursive definition. For the base case, a leaf labeled \texttt{elephant} is what we need. Then, a tree has an elephant if either of its main subtrees has one (hence two separate rules, one for each subtree):
\texttt{look_for_elephant(leaf(elephant)). look_for_elephant(tree(X,_)) :- look_for_elephant(X). look_for_elephant(tree(_,X)) :- look_for_elephant(X).}