**Puzzle 1**

**Problem Statement**

1. Dogs like bones.
2. Dogs eat everything they like.
3. Max is a dog.
4. (Conclusion) Max eats bones.

**1. Predicate Form**

**Predicates:**

* Dog(x) : x is a dog.
* Likes(x, y) : x likes y.
* Eats(x, y) : x eats y.
* Bone(y) : y is a bone.
* x = Max : Max is an individual constant.

**Logical Representation:**

1. ∀x (Dog(x) → Likes(x, Bone))
2. ∀x ∀y (Dog(x) ∧ Likes(x, y) → Eats(x, y))
3. Dog(Max)
4. Goal: Eats(Max, Bone)

**2. Clause Form**

1. ¬Dog(x) ∨ Likes(x, Bone)
2. ¬Dog(x) ∨ ¬Likes(x, y) ∨ Eats(x, y)
3. Dog(Max)
4. Goal: ¬Eats(Max, Bone)

**3. Assumptions and Goal**

**Assumptions:** The above clauses (1-3).  
**Goal:** To derive a contradiction from the negation of the conclusion (¬Eats(Max, Bone)).

**4. Prover9 Proof**

**Input File for Prover9:**

-Dog(x) | Likes(x, bone).

-Dog(x) | -Likes(x, y) | Eats(x, y).

Dog(max).

**Goal:**

Eats(max, bone).

**Expected Proof:**

Prover9 will infer:

1. From clause 3, Max is a dog.
2. Using clause 1, since Max is a dog, Max likes bones.
3. Using clause 2, since Max is a dog and likes bones, Max eats bones.  
   The conclusion "Max eats bones" will be derived, leading to a contradiction when ¬Eats(Max, Bone) is assumed.

**Actual Proof:**

1 Eats(max,bone) # label(non\_clause) # label(goal). [goal].

2 Dog(max). [assumption].

3 -Dog(x) | Likes(x,bone). [assumption].

4 -Dog(x) | -Likes(x,y) | Eats(x,y). [assumption].

5 -Likes(max,x) | Eats(max,x). [resolve(2,a,4,a)].

6 -Eats(max,bone). [deny(1)].

7 -Likes(max,bone). [resolve(5,b,6,a)].

8 Likes(max,bone). [resolve(2,a,3,a)].

9 $F. [resolve(7,a,8,a)].

**Puzzle 2**

**Problem Statement**

1. Every bird sleeps in some tree.
2. Every loon is a bird, and every loon is aquatic.
3. Every tree in which any aquatic bird sleeps is beside some lake.
4. Anything that sleeps in anything that is beside any lake eats fish.
5. (Conclusion) Every loon eats fish.

**1. Predicate Form**

**Predicates:**

* Bird(x) : x is a bird.
* Loon(x) : x is a loon.
* Aquatic(x) : x is aquatic.
* Tree(y) : y is a tree.
* SleepsIn(x, y) : x sleeps in y.
* Beside(y, z) : y is beside z.
* Lake(z) : z is a lake.
* Eats(x, y) : x eats y.
* Fish(y) : y is fish.

**Logical Representation:**

1. ∀x (Bird(x) → ∃y (Tree(y) ∧ SleepsIn(x, y)))
2. ∀x (Loon(x) → Bird(x) ∧ Aquatic(x))
3. ∀x ∀y (Aquatic(x) ∧ SleepsIn(x, y) → ∃z (Lake(z) ∧ Beside(y, z)))
4. ∀x ∀y ∀z (SleepsIn(x, y) ∧ Beside(y, z) ∧ Lake(z) → Eats(x, Fish))
5. Goal: ∀x (Loon(x) → Eats(x, Fish))

**2. Clause Form**

1. ¬Bird(x) ∨ Tree(f(x))
2. ¬Bird(x) ∨ SleepsIn(x, f(x))
3. ¬Loon(x) ∨ Bird(x)
4. ¬Loon(x) ∨ Aquatic(x)
5. ¬Aquatic(x) ∨ ¬SleepsIn(x, y) ∨ Lake(f(x, y))
6. ¬Aquatic(x) ∨ ¬SleepsIn(x, y) ∨ Beside(y, f(x, y))
7. ¬SleepsIn(x, y) ∨ ¬Beside(y, z) ∨ ¬Lake(z) ∨ Eats(x, Fish)
8. ¬Loon(x) ∨ Eats(x, Fish)

**3. Assumptions and Goal**

**Assumptions:** The above clauses (1-7).  
**Goal:** To derive clause 8 (¬Loon(x) ∨ Eats(x, Fish)).

**4. Prover9 Proof**

**Input File for Prover9:**

-Bird(x) | Tree(f1(x)).

-Bird(x) | SleepsIn(x, f1(x)).

-Loon(x) | Bird(x).

-Loon(x) | Aquatic(x).

-Aquatic(x) | -SleepsIn(x, y) | Lake(f2(x, y)).

-Aquatic(x) | -SleepsIn(x, y) | Beside(y, f2(x, y)).

-SleepsIn(x, y) | -Beside(y, z) | -Lake(z) | Eats(x, fish).

**Goal:**

-Loon(x) | Eats(x, fish).

**Expected Proof:**

Prover9 will:

1. Use clause 3 to infer that loons are birds.
2. Use clauses 1 and 2 to show that every bird (and thus every loon) sleeps in some tree.
3. Use clauses 5 and 6 to show that every tree in which an aquatic bird sleeps is beside some lake.
4. Use clause 7 to conclude that anything sleeping beside a lake eats fish.
5. Derive the conclusion that every loon eats fish.

**Actual Proof:**

1 -Loon(x) | Eats(x,fish) # label(non\_clause) # label(goal). [goal].

2 -Loon(x) | Bird(x). [assumption].

4 -Bird(x) | SleepsIn(x,f1(x)). [assumption].

5 Loon(c1). [deny(1)].

6 -Loon(x) | Aquatic(x). [assumption].

8 -Loon(x) | SleepsIn(x,f1(x)). [resolve(2,b,4,a)].

9 Aquatic(c1). [resolve(5,a,6,a)].

10 -Aquatic(x) | -SleepsIn(x,y) | Lake(f2(x,y)). [assumption].

11 -Aquatic(x) | -SleepsIn(x,y) | Beside(y,f2(x,y)). [assumption].

12 SleepsIn(c1,f1(c1)). [resolve(8,a,5,a)].

13 -SleepsIn(x,y) | -Beside(y,z) | -Lake(z) | Eats(x,fish). [assumption].

14 -SleepsIn(c1,x) | Lake(f2(c1,x)). [resolve(9,a,10,a)].

15 -SleepsIn(c1,x) | Beside(x,f2(c1,x)). [resolve(9,a,11,a)].

16 -Beside(f1(c1),x) | -Lake(x) | Eats(c1,fish). [resolve(12,a,13,a)].

17 -Eats(c1,fish). [deny(1)].

18 -Beside(f1(c1),x) | -Lake(x). [resolve(16,c,17,a)].

19 Lake(f2(c1,f1(c1))). [resolve(14,a,12,a)].

20 -Beside(f1(c1),f2(c1,f1(c1))). [resolve(18,b,19,a)].

21 Beside(f1(c1),f2(c1,f1(c1))). [resolve(15,a,12,a)].

22 $F. [resolve(20,a,21,a)].

**Summary**

Both puzzles are solvable using Prover9 by:

* Representing the problem in predicate logic.
* Converting predicates to clause form.
* Running the proof with the assumptions and negated goal to derive a contradiction, confirming the conclusion.