**19. Student-Teacher-Subject Database**

prolog

Copy

Download

student(john, cs101).

student(sarah, cs101).

student(mike, math202).

teacher(dr\_smith, cs101).

teacher(dr\_jones, math202).

% Query: student(Name, Code), teacher(Teacher, Code).

**20. Planets Database**

prolog

Copy

Download

planet(mercury, rocky, 0.39).

planet(venus, rocky, 0.72).

planet(earth, rocky, 1.0).

planet(mars, rocky, 1.52).

planet(jupiter, gas\_giant, 5.20).

planet(saturn, gas\_giant, 9.58).

% Query: planet(Name, Type, Distance).

**21. Towers of Hanoi**

prolog

Copy

Download

hanoi(1, Start, End, \_) :-

write('Move top disk from '), write(Start), write(' to '), write(End), nl.

hanoi(N, Start, End, Via) :-

N > 1,

M is N - 1,

hanoi(M, Start, Via, End),

hanoi(1, Start, End, \_),

hanoi(M, Via, End, Start).

% Query: hanoi(3, left, right, center).

**22. Birds That Can Fly**

prolog

Copy

Download

can\_fly(penguin, no).

can\_fly(sparrow, yes).

can\_fly(ostrich, no).

can\_fly(eagle, yes).

can\_fly(kiwi, no).

% Query: can\_fly(Bird, yes).

**23. Family Tree**

prolog

Copy

Download

parent(john, mary).

parent(john, bob).

parent(mary, ann).

parent(mary, tom).

parent(bob, lisa).

male(john).

male(bob).

male(tom).

female(mary).

female(ann).

female(lisa).

father(Father, Child) :-

parent(Father, Child),

male(Father).

mother(Mother, Child) :-

parent(Mother, Child),

female(Mother).

% Query: father(Father, Child).

**24. Dieting System Based on Disease**

prolog

Copy

Download

diet(diabetes, low\_sugar).

diet(hypertension, low\_sodium).

diet(obesity, low\_calorie).

diet(anemia, iron\_rich).

recommend\_diet(Disease, Diet) :-

diet(Disease, Diet).

% Query: recommend\_diet(Disease, Diet).

**25. Monkey Banana Problem**

prolog

Copy

Download

state(atdoor, onfloor, atwindow, hasnot).

state(atwindow, onfloor, atwindow, hasnot).

state(atwindow, onbox, atwindow, hasnot).

state(atwindow, onbox, atwindow, has).

move(state(middle, onfloor, middle, hasnot), grasp, state(middle, onfloor, middle, has)).

move(state(P, onfloor, P, H), climb, state(P, onbox, P, H)).

move(state(P1, onfloor, P1, H), push(P1, P2), state(P2, onfloor, P2, H)).

move(state(P1, onfloor, B, H), walk(P1, P2), state(P2, onfloor, B, H)).

canget(state(\_, \_, \_, has)).

canget(State1) :-

move(State1, \_, State2),

canget(State2).

% Query: canget(state(atdoor, onfloor, atwindow, hasnot)).

**26. Fruit and Color with Backtracking**

prolog

Copy

Download

fruit\_color(apple, red).

fruit\_color(banana, yellow).

fruit\_color(grape, purple).

fruit\_color(orange, orange).

fruit\_color(apple, green). % Some apples are green

% Query: fruit\_color(Fruit, Color).

**27. Best First Search**

prolog

Copy

Download

% This is a simplified implementation

best\_first\_search(Start, Goal) :-

bfs([node(Start, [])], Goal, Path),

reverse(Path, ReversedPath),

write('Path: '), write(ReversedPath).

bfs([node(Goal, Path)|\_], Goal, [Goal|Path]).

bfs([node(State, Path)|Rest], Goal, Solution) :-

findall(node(NextState, [State|Path]),

(move(State, NextState), \+ member(NextState, Path)),

Children),

append(Rest, Children, NewQueue),

bfs(NewQueue, Goal, Solution).

% Requires defining move/2 for your specific problem

**28. Medical Diagnosis**

prolog

Copy

Download

symptom(fever, flu).

symptom(cough, flu).

symptom(fever, cold).

symptom(sneezing, cold).

symptom(headache, migraine).

symptom(nausea, migraine).

diagnose(Symptoms, Diagnosis) :-

findall(D, (member(S, Symptoms), symptom(S, D)), Diagnoses),

list\_to\_set(Diagnoses, PossibleDiagnoses),

member(Diagnosis, PossibleDiagnoses).

% Query: diagnose([fever, cough], D).

**29. Forward Chaining**

prolog

Copy

Download

% Knowledge base

rule(has\_wings, can\_fly).

rule(can\_fly, is\_bird).

rule(lays\_eggs, is\_bird).

rule(is\_bird, is\_animal).

% Forward chaining

forward\_chain(Facts, NewFacts) :-

findall(Conclusion,

(member(Fact, Facts),

rule(Fact, Conclusion),

\+ member(Conclusion, Facts)),

NewFacts),

NewFacts \= [].

infer\_all(Facts, AllFacts) :-

forward\_chain(Facts, NewFacts),

append(Facts, NewFacts, UpdatedFacts),

infer\_all(UpdatedFacts, AllFacts).

infer\_all(Facts, Facts).

% Query: infer\_all([has\_wings], AllFacts).

**30. Backward Chaining**

prolog

Copy

Download

% Knowledge base

rule(is\_bird, [has\_wings, can\_fly]).

rule(is\_bird, [lays\_eggs]).

rule(can\_fly, [has\_wings]).

backward\_chain(Goal, KnownFacts, Proof) :-

member(Goal, KnownFacts),

Proof = [Goal].

backward\_chain(Goal, KnownFacts, [Goal|Subproofs]) :-

rule(Goal, Subgoals),

backward\_chain\_list(Subgoals, KnownFacts, Subproofs).

backward\_chain\_list([], \_, []).

backward\_chain\_list([H|T], KnownFacts, [HProof|TProof]) :-

backward\_chain(H, KnownFacts, HProof),

backward\_chain\_list(T, KnownFacts, TProof).

% Query: backward\_chain(is\_bird, [has\_wings], Proof).

**32. Pattern Matching**

prolog

Copy

Download

match([], []).

match([H|T], [H|T2]) :- match(T, T2).

match([\_|T], [\_|T2]) :- match(T, T2).

% Query: match([a,b,c], [a,X,Y]).

**33. Count Vowels**

prolog

Copy

Download

vowel(a). vowel(e). vowel(i). vowel(o). vowel(u).

count\_vowels([], 0).

count\_vowels([H|T], Count) :-

(vowel(H) ->

count\_vowels(T, SubCount),

Count is SubCount + 1

;

count\_vowels(T, Count)

).

% Query: count\_vowels("hello world", Count).