Day-1: Introduction to Logic Programming using Prolog-environment set up, first prolog programming.

* <http://artint.info/html/ArtInt_56.html>
* <https://www.doc.gold.ac.uk/~mas02gw/prolog_tutorial/prologpages/search.html>

Day-2: Family tree using Prolog Programming.

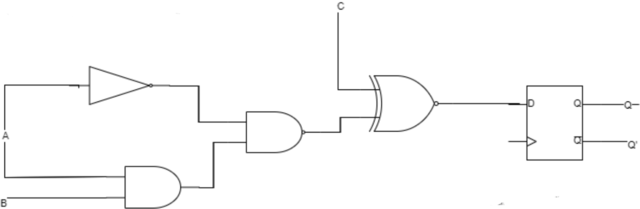
* <https://www.cefns.nau.edu/~edo/Classes/CS470-570_WWW/Assignments/Prog3-Logic/Testing-Assignment.html>
* <https://www.cse.unsw.edu.au/~billw/dictionaries/prolog/comparison.html>
* <https://www.tutorialspoint.com/prolog/prolog_relations.htm>

Day-3: Combinational and sequential circuit implementation using prolog

Logic gates:

* Facts: AND, OR, NOT
* Rules: XOR, NOR, NAND, Half Adder, Full adder, 4bit adder, 2-1 Mux.

Circuit implementation:



<https://www.geeksforgeeks.org/circuit-combinational-and-sequential-implementation-using-prolog/>

Day-4: Solving real life problems using prolog programming

**Here is an example program for a simple course scheduling system using Prolog programming:**

% Define the available courses

course(cs101).

course(cs102).

course(math101).

course(math102).

course(eng101).

course(eng102).

% Define the available time slots

time\_slot(monday, 9).

time\_slot(monday, 11).

time\_slot(tuesday, 9).

time\_slot(tuesday, 11).

time\_slot(wednesday, 9).

time\_slot(wednesday, 11).

time\_slot(thursday, 9).

time\_slot(thursday, 11).

time\_slot(friday, 9).

time\_slot(friday, 11).

% Define the available instructors

instructor(john).

instructor(jane).

instructor(bob).

% Define the course schedule predicate

schedule(Course, Instructor, Day, Time) :-

course(Course),

instructor(Instructor),

time\_slot(Day, Time).

**This program defines the available products and customers, as well as the sales history. It then defines a price predicate that calculates the price of a product based on the customer's budget, and a sales\_report predicate that returns a report of all sales that meet the customer's budget.**

% Define the available products

product(laptop, 1000).

product(desktop, 800).

product(printer, 200).

product(scanner, 150).

% Define the available customers

customer(john, 5000).

customer(jane, 3000).

customer(bob, 2000).

% Define the sales history

sale(john, laptop, 1).

sale(john, printer, 2).

sale(jane, desktop, 2).

sale(jane, scanner, 1).

sale(bob, laptop, 2).

% Define the price calculation predicate

price(Customer, Product, Quantity, Price) :-

customer(Customer, Budget),

product(Product, UnitPrice),

Price is UnitPrice \* Quantity,

Price =< Budget.

% Define the sales report predicate

sales\_report(Customer, Product, Quantity, Price) :-

sale(Customer, Product, Quantity),

price(Customer, Product, Quantity, Price).

**Exercise**

Here are some exercises for Prolog programming on real-life problems:

* Family Tree: Create a Prolog program to represent a family tree. Define predicates for parent, grandparent, sibling, and uncle/aunt. Use the program to answer questions like "Who is John's uncle?" or "Who are Jane's siblings?"
* Employee Scheduling: Create a Prolog program to generate employee schedules based on their availability and work requirements. Define predicates for employee, shift, day, and available. Use the program to generate schedules for a given week.

**Sample program:**

% Define the available employees

employee(john).

employee(jane).

employee(bob).

employee(sara).

% Define the available shifts

shift(morning).

shift(afternoon).

shift(evening).

% Define the available days

day(monday).

day(tuesday).

day(wednesday).

day(thursday).

day(friday).

% Define the employee availability

available(john, monday, morning).

available(john, tuesday, afternoon).

available(john, wednesday, evening).

available(john, thursday, morning).

available(john, friday, afternoon).

available(jane, monday, morning).

available(jane, tuesday, morning).

available(jane, wednesday, afternoon).

available(jane, thursday, evening).

available(jane, friday, morning).

available(bob, monday, afternoon).

available(bob, tuesday, evening).

available(bob, wednesday, morning).

available(bob, thursday, afternoon).

available(bob, friday, evening).

available(sara, monday, evening).

available(sara, tuesday, morning).

available(sara, wednesday, afternoon).

available(sara, thursday, evening).

available(sara, friday, morning).

% Define the work requirements

work(morning, 2).

work(afternoon, 3).

work(evening, 2).

% Define the schedule predicate

schedule(Employee, Day, Shift) :-

employee(Employee),

day(Day),

shift(Shift),

available(Employee, Day, Shift),

findall(Shift, available(Employee, Day, Shift), Shifts),

length(Shifts, ShiftCount),

work(Shift, RequiredShifts),

ShiftCount >= RequiredShifts.

% Define the schedule report predicate

schedule\_report(Day) :-

day(Day),

write('--- Schedule for '), write(Day), writeln(' ---'),

schedule(Employee, Day, Shift),

write(Employee), write(' - '), write(Shift), writeln(' shift'),

fail.

schedule\_report(\_).

* Music Recommendation: Create a Prolog program to recommend music to a user based on their preferences. Define predicates for artist, genre, and like. Use the program to recommend music that the user might enjoy based on their favorite artists and genres.

**Sample program:**

% Define the available artists

artist(beatles).

artist(beyonce).

artist(johnny\_cash).

artist(taylor\_swift).

artist(ed\_sheeran).

artist(rihanna).

% Define the available genres

genre(rock).

genre(pop).

genre(country).

genre(rap).

% Define the user preferences

like(john, beatles, rock).

like(john, ed\_sheeran, pop).

like(john, taylor\_swift, pop).

like(jane, beyonce, pop).

like(jane, rihanna, pop).

like(jane, johnny\_cash, country).

like(jim, johnny\_cash, country).

like(jim, ed\_sheeran, pop).

like(sara, rihanna, pop).

like(sara, beyonce, pop).

like(sara, ed\_sheeran, pop).

% Define the recommend predicate

recommend(User, Artist) :-

like(User, \_, Genre),

artist(Artist),

genre(Genre),

like(User, Artist, Genre).

% Define the recommend report predicate

recommend\_report(User) :-

like(User, \_, \_),

write('--- Recommendations for '), write(User), writeln(' ---'),

recommend(User, Artist),

write(Artist), writeln(' - recommended'),

fail.

recommend\_report(\_).

* Food Allergy Checker: Create a Prolog program to check if a given meal contains any allergens for a user with specific food allergies. Define predicates for food, ingredient, and allergen. Use the program to check if a given meal is safe for the user to eat.

**Sample Program:**

% Define the food, ingredient, and allergen predicates

food(pizza).

food(soup).

food(salad).

ingredient(pizza, dough).

ingredient(pizza, tomato\_sauce).

ingredient(pizza, cheese).

ingredient(soup, broth).

ingredient(soup, vegetables).

ingredient(salad, lettuce).

ingredient(salad, tomatoes).

allergen(milk).

allergen(wheat).

allergen(gluten).

allergen(shellfish).

% Define the predicate to check if a given meal is safe for a user with specific food allergies

is\_safe\_to\_eat(Meal, Allergies) :-

findall(Ingredient, (ingredient(Meal, Ingredient), allergen(Allergy), allergic\_to(Allergy, Allergies), \+ safe\_for\_allergic(Ingredient, Allergy)), UnsafeIngredients),

length(UnsafeIngredients, 0).

% Define the helper predicates for checking allergies

allergic\_to(Allergy, [Allergy|\_]).

allergic\_to(Allergy, [\_|RestAllergies]) :- allergic\_to(Allergy, RestAllergies).

safe\_for\_allergic(Ingredient, Allergy) :-

\+ allergen(Allergy) ; \+ ingredient\_has\_allergen(Ingredient, Allergy).

ingredient\_has\_allergen(Ingredient, Allergy) :-

ingredient(Ingredient, Ingredients),

allergen(Allergy),

member(Allergy, Ingredients).

* Travel Planner: Create a Prolog program to help plan a trip based on the user's preferences and budget. Define predicates for destination, budget, activity, and recommend. Use the program to recommend destinations and activities that fit the user's preferences and budget.

**Sample program:**

% Define destinations and their corresponding costs

destination(paris, 1000).

destination(london, 1200).

destination(tokyo, 1500).

destination(new\_york, 1300).

destination(los\_angeles, 1100).

destination(sydney, 1400).

% Define activities and their corresponding costs

activity(sightseeing, 50).

activity(shopping, 100).

activity(food\_tasting, 75).

activity(beach, 25).

activity(hiking, 30).

activity(museum, 40).

% Define the budget predicate

budget(X) :- X >= 1000.

% Define the recommend predicate

recommend(Destination, Activity) :-

destination(Destination, Cost),

activity(Activity, ActivityCost),

TotalCost is Cost + ActivityCost,

budget(Budget),

TotalCost =< Budget.

% Example usage:

% recommend(Destination, Activity).

% This will recommend a Destination and Activity that fit the user's preferences and budget.

These exercises can be customized and expanded upon to create more complex Prolog programs for a variety of real-life problems.

Day-5: Graph search using Prolog Programming

* Finding path in a graph: chapter 4: page-86 [2]

% Define the graph

edge(a, b).

edge(a, c).

edge(b, d).

edge(c, d).

edge(c, e).

edge(d, e).

% Define the path predicate

path(Start, End, Path) :-

dfs(Start, End, [Start], Path).

% Define the depth-first search predicate

dfs(Node, End, Visited, Path) :-

Node == End,

reverse(Visited, Path).

dfs(Node, End, Visited, Path) :-

edge(Node, Next),

not(member(Next, Visited)),

dfs(Next, End, [Next|Visited], Path).

Day-6: Introduction to neural network using python.

Book references:

1. The Art of Prolog: Advanced Programming Techniques, Second Edition.
2. (International Computer Science Series) Ivan Bratko - Prolog programming for artificial intelligence-Pearson Education Canada (2012)