

CSC 372, Spring 2025

# Prolog Introduction

Michelle Strout



February 20, 2025

# Plan

- Announcements

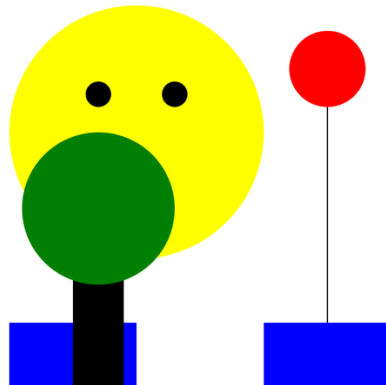
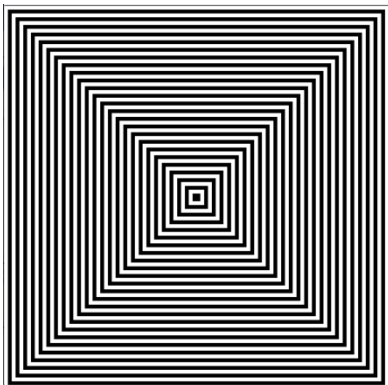
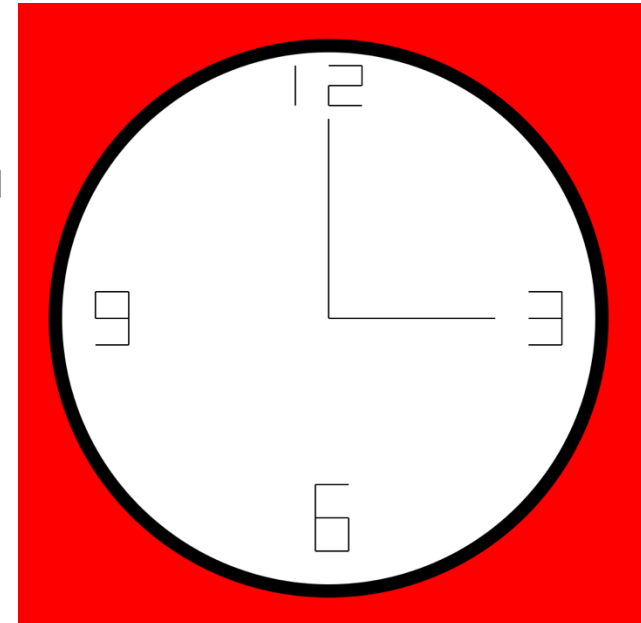
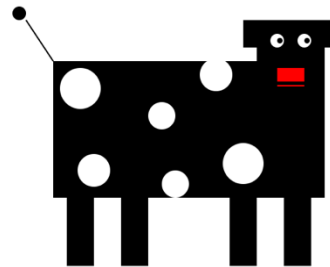
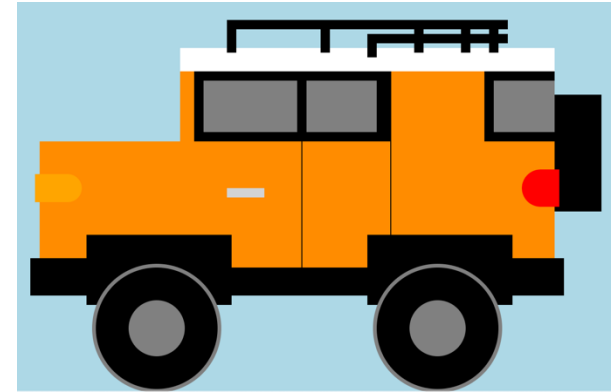
- SA4 MT1 Prolog is posted and due Wednesday Feb 26<sup>th</sup>
- LA2 grades will be out soon

- Last time

- Midterm 1

- Today

- Introduction to Prolog



# TopHat Questions

- Some questions about Prolog

# Outline for rest of today

- Motivation for Prolog
- Comparison of Prolog with other languages
- Hands on Prolog examples

*Credit for most of the following slides goes to Prof. Melanie Lotz*

# Motivation for Prolog

- A Prolog program is

- a collection of facts and rules that can be used to answer queries.

- Why developed?

- Natural language processing, for parsing sentences and reasoning about meaning.
- AI in the 1970s and 1980s relied heavily on symbolic reasoning, expert systems, and rule-based decision-making, all of which required some form of logical inference.
- Traditional procedural languages struggled with problems involving symbolic manipulation, logic inference, and rule-based reasoning.

- Why did it become popular?

- In 1980s, Japan's Fifth Generation Computer Systems (FGCS) was built around logic programming (Prolog).
- It was widely used in expert systems, NLP applications, and automated reasoning.

# Kinds of Applications Written in Prolog

- **Automated Reasoning**
  - Theorem Proving: Derives logical conclusions from axioms.
  - Type Inference: Infers types in functional programming.
  - Expert Systems: Supports decision-making (e.g., medical diagnosis).
- **Natural Language & Knowledge Representation**
  - Chatbots: Uses rule-based logic for conversations.
  - Parsing & Grammar Checking: Analyzes sentence structure.
  - Deductive Databases: Uses logic rules for query processing.
- **AI, Robotics, & Planning**
  - SAT Solver: Solves constraint satisfaction and optimization problems.
  - Sudoku Solver: Fills in grids using constraint logic.
  - Automated Planning: AI-driven task execution.
  - Constraint Solving: Optimizes pathfinding and scheduling.

# Comparison of Prolog with other PLs

Feature	Prolog (Logic Programming)	Python/Java (Imperative/OOP)	SML (Functional)
<b>Paradigm</b>	Logic-based (declarative)	Imperative & Object-Oriented	Functional
<b>Execution Model</b>	Uses <b>backtracking</b> and <b>unification</b> to find solutions	Step-by-step execution of commands	Evaluates expressions via function calls
<b>Control Flow</b>	Controlled by <b>rules &amp; queries</b>	Controlled by loops, conditionals, and method calls	Controlled by function calls and recursion
<b>Pattern Matching</b>	Based on <b>unification</b> (works in multiple directions)	Limited (e.g., Java switch, Python match)	Strict pattern matching in function definitions
<b>Use Cases</b>	AI, rule-based systems, expert systems, NLP	General-purpose (web, data science, software dev)	Mathematical computation, type inference, compiler design
<b>Mutability</b>	Mostly immutable (variables assigned once)	Mutable state common	Mostly immutable (functional purity)

From ChatGPT 2/20/25

# Using Prolog

- Use the docker container from SA1

```
// start docker desktop

// start the container
docker run -it -v $(pwd):/workspace chapel_sml_prolog

devuser@d918ee370200:~$ cd /workspace

devuser@d918ee370200:/workspace$ swipl
...
For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?-
apropos(Word) .

?-
```



# Prolog Basics

- Everything in Prolog is a term
- A term is
  - a constant,
  - a variable,
  - or a compound term.
- A compound term consists of
  - Functor – The name of the term, which must be an atom (a lowercase identifier or quoted string).
  - Arity – The number of arguments the term takes.
  - Arguments – The values or subterms inside the parentheses, which can be atoms, numbers, variables, or other compound terms.

- **A constant is**

- an integer (e.g., 123, -123),
- a real number (e.g., 3.14, -3.14e-4),
- or an atom
  - starts with a lowercase letter and has 0 or more letters, digits, or underscores after it
  - treated like a constant
  - examples: fred, ml, parent, \*, =,

- A variable starts with an uppercase letter or an underscore (the ones starting with \_ get special treatment).
  - X
  - Child
  - \_123
- A compound term has an atom followed by a list of terms.
  - x(y, z)
  - parent(adam, Child)

# Question on Prolog Basics

- In the compound terms below, identify each piece as one of the following
  - a constant integer
  - a constant atom
  - a variable
  - a functor/predicate name, which is an atom
  - arity
- Compound Terms
  - `x(y, z)`
  - `parent(adam, Child)`
  - `likes(alice, hobby(reading))`
- Prolog identifies terms by Functor/Arity, e.g., `parent/2` or `hobby/1`.

# Prolog Unification

- Unification is pattern matching.
- Two terms unify if there is some way of binding their variables that makes them identical.
  - `parent(adam, Child)` and `parent(adam, seth)` unify by binding `Child` (a variable) to `seth` (an atom)
  - `friend(alice, X)` and `friend(Y, bob)` unify by binding `X` to `bob` and `Y` to `alice`.

# How Prolog Works

- The language system maintains a collection of facts and rules of inference, a kind of internal database that can change as the system runs.
- A Prolog program is a set of data for this database.
- The simplest item in the database is a fact, which is just a term followed by a period. (e.g. `parent(adam, seth).`)
- A rule tells the system how to prove something.
- A goal is a term to be proved.

# Example: Building a family tree

- **Baggins family tree**
  - [https://tolkiengateway.net/wiki/Baggins\\_family](https://tolkiengateway.net/wiki/Baggins_family)

# Exercise

- Try the following queries:
  - $3 = 3.$
  - $3 = 4.$
  - $X = 3.$
  - $3 = 1 + 2.$
  - $X = 1 + 2.$
  - $1 + 2 = 1 + 2.$
  - $1 + 2 = 3 + 0.$
  - $X + 2 = 1 + 2.$
  - $X + 1 = 1 + 2.$
  - $1 + 2 = X.$
  - $1 + 2 = 3.$
- What does  $=$  mean/do in Prolog?



# Exercise

- Try the following queries:
  - 3 is 3.
  - 3 is 4.
  - X is 3.
  - 3 is  $1 + 2$ .
  - X is  $1 + 2$ .
  - $1 + 2$  is  $1 + 2$ .
  - $1 + 2$  is  $3 + 0$ .
  - $X + 2$  is  $1 + 2$ .
  - $X + 1$  is  $1 + 2$ .
  - $1 + 2$  is X.
  - $1 + 2$  is 3.
- What does is mean/do in Prolog?

# Exercise

- Try the following queries:
  - $3 ::= 3.$
  - $3 ::= 4.$
  - $X ::= 3.$
  - $3 ::= 1 + 2.$
  - $X ::= 1 + 2.$
  - $1 + 2 ::= 1 + 2.$
  - $1 + 2 ::= 3 + 0.$
  - $X + 2 ::= 1 + 2.$
  - $X + 1 ::= 1 + 2.$
  - $1 + 2 ::= X.$
  - $1 + 2 ::= 3.$
- What does  $::=$  mean/do in Prolog?

# Terms that have infix notation

+

-

\*

/

>

<

>=

=<

- Let's use these in some examples

# Exercise

- Try out the following queries. Write down the results.
  - $X = [1,2,3,4]$ .
  - $[X | Y] = [1,2,3,4]$ .
  - $[X,Y | Z] = [1,2,3,4]$ .
  - $[_ , _ , _ | X] = [1,2,3,4]$ .
  - $[A,B,C,D | E] = [1,2,3,4]$ .
- Explain what each of the following mean:  $[]$ ,  $|$ ,  $_$

# Exercise

- Predict the results of the following queries.
  - $[A, B] = [1, 2]$ .
  - $[A, B, C] = [1, 2]$ .
  - $[A, B \mid C] = [1, 2]$ .
  - $[\_ \mid A] = [1, 2]$ .