Logic and Functional Programming

Lecture 1

Dr. Cristian-Paul Bara

Computer Science Department
Faculty of Mathematics and Computer Science,
Babeş Bolyai University,
Cluj-Napoca, Romania
2025

Who am I?

- Cristian-Paul Bara

Who am I?

- Cristian-Paul Bara
- Ph.D. @ University of Michigan

Who am I?

- Cristian-Paul Bara
- Ph.D. @ University of Michigan
- Senior Data Scientist @ Bosch

Table of Contents

Programming Paradigms

- What is a Programming Paradigm?

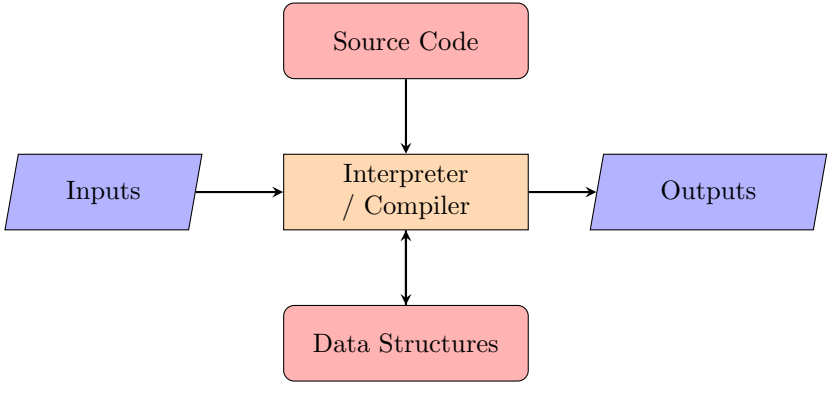
Programming Paradigms

- What is a Programming Paradigm?
- What programming Paradigms do you Know?

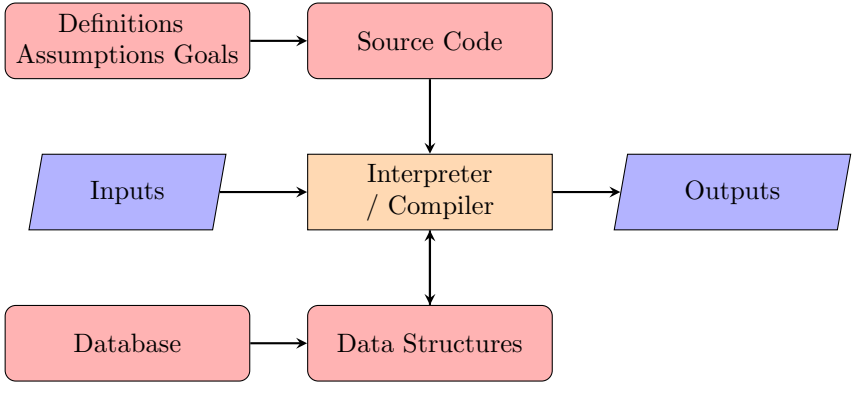
Programming Paradigms

- What is a Programming Paradigm?
- What programming Paradigms do you Know?
- Why one and not another?

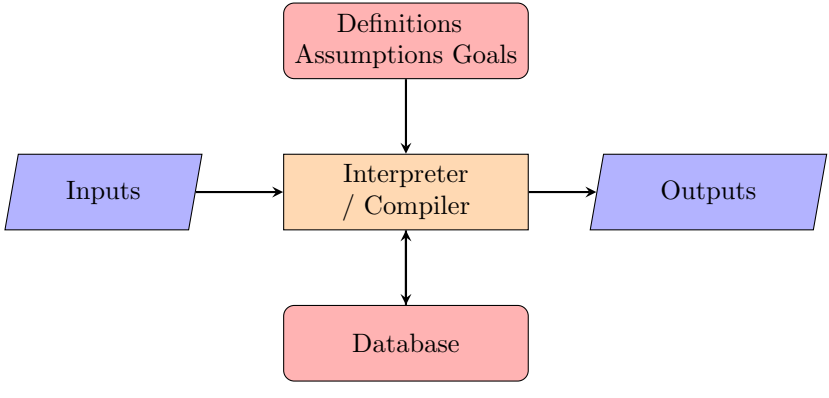
Logic as a programing Paradigm



Logic as a programing Paradigm



Logic as a programing Paradigm



Logic as a Specification Language

Language of Logic

- Domain Independent
- Highly Expressive

Logic Interpreters / Compilers

- Automated Reasoners capable of drawing conclusions
- Can take advantage of domain-dependent reasoners but are also capable domain-independent reasoning

Logic as a Specification Language

- Database Programming (Datalog, SQL)
- Classical Logic Programming (Prolog)
- Dynamic Logic Programming (Epilog, LPS)

- Constraint Satisfaction
- Program Synthesis

- Answer Set Programming (ASP)
- Probabilistic Logic
- Inductive Logic Programming (Progol)

Traditional Programming

Benefits

- Efficiency
- Lots of traditional programmers
- Well established software engineering practices

Disadvantages

- Creation, maintenance expensive and time-consuming
- Different programs for different tasks
- Difficult to explain results
- Programs not comprehensible to ordinary users

Why Logic Programming? - Ease of Creation

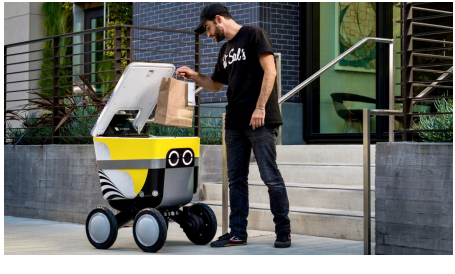
Logic Programs are relatively easy to create.

- Requires **little work**. The specification is the program; no need to make choices about data structures and algorithms.
- Specification authors can get by with **few assumption**s about the capabilities of systems executing those programs.
- **Easier to learn** logic programming than traditional programming. Think spreadsheets.

Oddly, expert computer programmers often have more trouble with logic programming than novices.

Why Logic Programming? - Adaptability

Easy to deal with changing circumstances.



Why Logic Programming? - Versatility

Easy to use for multiple tasks.

Sample Program: A person X is the grandparent of a person Z if and only if there is a person Y such that X is the parent of Y and Y is the parent of Z.

Uses

- Determine whether John is the grandparent of Dan.
- Determine all of the grandchildren of John.
- Compute the grandparents of Dan.
- Compute all grandparent-grandchildren pairs.

Why Logic Programming? - Successesful Applications

- Logic circuit Simulation, Configuration, Diagnosis, and Test Generation
- Interactive Web Pages
- Business Rules and Workflows (SAP, Oracle, IBM)
- Computational Law (Planning for, and Analysing Compliance)
- General Game Playing

Why Logic Programming? - Successesful Applications

Why Logic Programming? - Not so Successesful Applications ... yet ... maybe?

- Natural Language Processing
- Theorem Proving

Why Logic Programming? - Still relevant!

- Google releases Mangle (2023)
- A language meant for *Deductive Database Programming*
- Built over Datalog
- Runs by applying Recursion and Rules over Gnowledge Graphs

Installation

- Google "install swi-prolog on [your-OS]"
- Ubuntu:

```
$ sudo apt-add-repository ppa:swi-prolog/stable
$ sudo apt update
$ sudo apt install swi-prolog
```

- Windows: https://www.swi-prolog.org/download/stable/bin/swipl-9.2.7-1.x64.exe.envelope
- MacOS: https://www.swi-prolog.org/download/stable/bin/swipl-9.2.7-1.fat.dmg

Facts

- Syntax: `relation(object1, object2, ...).`
- Names of properties/relationships begin with lower case letters.
- The relationship name appears as the first term.
- Objects appear as comma-separated arguments within parentheses.
- A period "." must end a fact.
- Objects also begin with lower case letters. They also can begin with digits (like 1234), and can be strings of characters enclosed in quotes e.g. color(penink, 'red').
- This is also called a predicate or clause.

Facts

```
father(joe,paul).
father(joe,mary).
father(joe,hope).
mother(jane,paul).
mother(jane,mary).
mother(jane,hope).
```

Rules

- Syntax: `r1(o1, o2, ...)  :- r2(q1, q2,...), r3(...), ...  .`
- A rule is a predicate expression that uses logical implication `(:-)` to describe a relationship among facts.
- This sentence is interpreted as: left_hand_side if right_hand_side.
- The left_hand_side is restricted to a single, positive, literal, which means it must consist of a positive atomic expression. It cannot be negated and it cannot contain logical connectives.
- This notation is known as a Horn clause.
- The Horn clause calculus is equivalent to the first-order predicate calculus.

Rules

```
parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).
sibling(X,Y) :- parent(Z,X),parent(Z,Y),X\=Y.
child(X,Y) :- parent(Y,X).
female(mary).
female(hope).
female(X) :- mother(X,_).
male(paul).
male(X) :- father(X,_).
son(X,Y) :- parent(Y,X),male(X).
daughter(X,Y) :- parent(Y,X),female(X).
```

Goals

- Syntax: `?- clause .`
- The Prolog interpreter responds to queries about the facts and rules represented in its database.
- The database is assumed to represent what is true about a particular problem domain.
- In making a query you are asking Prolog whether it can prove that your query is true.
- The interpreter will answer YES if the clause is true, and NO otherwise

Goals

```
?- father(joe,paul).
true.
?- father(joe,X).
X = paul ;
X = mary ;
X = hope.
?- son(X, joe).
X = paul ;
false.
?- daughter(X,joe).
X = mary ;
X = hope ;
false.
```

Sections of a Prolog program.

```
$ cat family.pl
father(joe,paul).
father(joe,mary).
father(joe,hope).
mother(jane,paul).
mother(jane,mary).
mother(jane,hope).
female(mary).
female(hope).
female(X) :- mother(X,_).
male(paul).
male(X) :- father(X,_).
```

```
$ swipl
?- consult('family.pl').
true.
?- male(X).
X = paul ;
X = joe ;
X = joe ;
X = joe.
?- female(X).
X = mary ;
X = hope ;
X = jane ;
X = jane ;
X = jane.
```