Chapter 1: Introduction
"What" vs "How"

Declarative vs Procedural Programming

Procedural programming
• The programmer has to specify how to get the output for the range of required inputs.
• The programmer must know the appropriate algorithm.

Declarative programming
• Requires a more descriptive style.
• The programmer must know what relationships hold between various entities.
Example: List Concatenation

**In procedural style:**

```plaintext
list procedure cat(list a, list b)
{
    list t = list u = copylist(a);
    while (t.tail != nil) t = t.tail;
    t.tail = b;
    return u;
}
```

**In declarative style:**

```prolog
cat([], L, L).
cat([H | T], L, [H | Z]) :- cat(T, L, Z).
```
Logic Programming

- A declarative style programming paradigm.
- Computation through logical deduction.
- Uses the language of logic to express data and programs.
- Most of current logic programming languages use first order logic (FOL).
- Prolog – the most popular logic programming language.
Historical Facts

1970-ies:

- **Bob Kowalski.**
  "Predicate Logic as a Programming Language".
  IFIP Congress, Stockholm

- **Alain Colmerauer** and his group.
  Interpreter of the first logic programming language Prolog.
  Marseille
Prolog

- Prolog is the main subject of this course
- Used in Artificial Intelligence, Natural Language Processing, Automated Reasoning, XML Querying...
- Exists in many dialects (Sicstus Prolog, SWI Prolog, Prolog IV, BinProlog, Ciao Prolog, Prolog LPA, Visual Prolog, YAP Prolog, Strawberry Prolog(...
- (Almost) all the dialects agree on the “core" part (ISO Standard for Prolog)
Prolog in Industrial Applications

Clarissa:

• A fully voice-operated procedure browser.
• Developed at NASA.
• Used on the International Space Station.
• Enables astronauts to give full attention to the task while they navigate through complex procedures using spoken commands.
• Implemented in SICStus Prolog.
Prolog in Industrial Applications

Some other solutions:

• **FleetWatch** – fully integrated operations control and schedules planning solution. Used by 21 international airlines, among them Comair (USA), Italian branch of Lauda Air, Malev (Hungary), DHL Europe, Asiana (South Korea), Hainan (China), Royal Jordanian, Kuwait Airways, Cimber Air (Denmark), etc.

• **SCORE** – a long term airport capacity management system for coordinated airports. Successfully Used at IATA Scheduling Conferences. Users in more than 20 countries.

• **ARGOS** – a Decision Support System (DSS) for enhancing Crisis Management for incidents with Chemical, Biological, Radiological, and Nuclear (CBRN) releases.
Contents

Basics of PROLOG

- Facts
- Questions
- Variables
- Conjunction
- Rules
PROLOG

Used to solve problems involving
• objects, and
• relationships between objects.
Example

John owns the book
- The relationship: *ownership*
- The objects: *book, John*

Directional:
- John owns the book
- **Not**: The book owns John
PROLOG

- Program can be thought of as a storehouse of facts and rules.
- Conversational Language: The user can ask questions about the set of facts and rules in the PROLOG program.
Sisters Example:

- A rule defining sisters and the facts about the people involved.
- The user would ask:
  - Are these two people sisters?
- The system would answer
  - yes (true) or no (false)
Declaring Facts about objects and their relationships.
Defining Rules about objects and their relationships.
Asking Questions about objects and their relationships.
Parts of Fact

- **lower-case**
- **relationship** (written first)
- **objects**
- **round brackets**
- **full stop at end**
- **separated by commas**

Example: `likes(john,mary)`
Order of Objects

likes(mary, john).

order defined by programmer

mary —likes— john

The fact says nothing about how john likes mary

john...no info...→ mary
Examples of Facts

Example
Gold is valuable.
\texttt{valuable(gold)}

Jane is a female.
\texttt{female(jane)}

John owns some gold.
\texttt{owns(john,gold)}

John is the father of Mary.
\texttt{father(john,mary)}

Are these expressions really facts? Is there anything missing?
Interpretation of Names

The name refers to an object.

- **Semantic Meaning:** Given by the programmer.
- **Syntactic Meaning:** a set of characters, as PROLOG sees it.
Interpretation of Names

- Name refers to an object.
- Name gold can refer to:
  - a particular lump of gold, or
  - the chemical element Gold having atomic number 79
- valuable(gold) can mean:
  - that particular lump of gold, named gold, is valuable, or
  - the chemical element Gold, named gold, is valuable.
- The programmer decides (in her usage) the meaning.
Fact Terminology

predicates

valuable(gold).

likes(mary, john).

play(john, mary, football).

arguments
Database

Definition
In PROLOG, database is a collection of facts.

- PROLOG draws its knowledge from these facts.
- The programmer is responsible for their accuracy.
Contents

Basics of PROLOG
Facts
Questions
Variables
Conjunction
Rules
Questions

- The database contains the facts from which the questions are answered.
- A Question can look exactly like a fact:
  - owns(mary,book).
- The difference is in which mode one is in
Questions

- In the interactive question mode (indicated by the question mark and dash): ?-

  - Meaning: If mary is interpreted as a person called Mary, and book is interpreted as some particular book, then

- ?- owns(mary,book).
  - means: Does Mary own the book?
Database Search

Example

Facts in the database:
likes(joe,fish).
likes(joe,mary).
likes(mary,book).
likes(john,book).

Questions:
?- likes(joe,money). no
?- likes(joe,mary). yes
?- king(john,france). no
Knowledge

The questions are always answered with respect to the database.

Example
Facts in the database:
\text{human}(\text{socrates}).
\text{human}(\text{aristotle}).
\text{athenian}(\text{socrates}).

Question:
Is Socrates Greek?
?- \text{greek}(\text{socrates})

The answer with respect to this database is \textbf{No}. 

Introduction 2/12/2015
Questions

Up until now questions just reflect exactly the database.

Does Mary like the book?
?- likes(mary,book).

More Interesting Question:
What objects does Mary like?

Variables.
Contents

Basics of PROLOG
Facts
Questions
Variables
Conjunction
Rules
Variables

Tiresome to ask about every object:
likes(john, this).
likes(john, that).

Better to ask:
What does John like?
or
Does John like $X$?
(i.e. use variables)
Question With Variables

Does John like $X$?

?- \texttt{likes(john,X)}.

or

?- \texttt{likes(john,SomethingThatJohnLikes)}.

$X$ and SomethingThatJohnLikes are variables.

Variable begins with a capital letter.
Database:
likes(john,flowers).

Question:
?- likes(john,X).

PROLOG answers:
X=flowers
Many Answers

Database:
likes(john,flowers).
likes(john,mary).
likes(paul,mary).

Question:
?-likes(john,X).

PROLOG answers:
X=flowers
and the user acknowledges
X=mary
and the user acknowledges
no
The first match is found: \( X = \text{flowers} \).

The user acknowledges.

- From that place on the next match is found (the search continues).
- From the place of the last instantiation no more match was found.
- Thus answer: no.
Contents

Basics of PROLOG

Facts

Questions

Variables

Conjunction

Rules
Conjunctions

More Complicated Relationships:

Does Mary like John and does John like Mary?

Both Conditions must be fulfilled.
Conjunctions

Database:

\[ \text{likes}(\text{mary}, \text{food}). \]
\[ \text{likes}(\text{mary}, \text{cola}). \]
\[ \text{likes}(\text{john}, \text{cola}). \]
\[ \text{likes}(\text{john}, \text{mary}). \]

Comma means Conjunction:

\[ ?- \text{likes}(\text{john}, \text{mary}), \text{likes}(\text{mary}, \text{john}). \]

Answer: no
A match for \text{likes}(\text{john}, \text{mary})
but none for \text{likes}(\text{mary}, \text{john})
Conjunctions with Variables

Is there anything that both mary and john like?

Find out what Mary likes and then see if John likes it.

?- likes(mary,X), likes(john,X).
Backtracking

- Find match for the first goal.
- Then see if matches the second.
- If not, find another match for the first.
- See if this matches the second.
- …etc.
Match First

?- likes(mary,X), likes(john,X)

likes(mary,food).
likes(mary, Cola).
likes(john, food).
likes(john, mary).
likes(john, Cola).

food
Match Second

?- likes(mary,X), likes(john,X)

likes(mary,food).
likes(mary,Cola).
likes(john,food).
likes(john,mary).

no not found

Introduction 2/12/2015
Backtrack

?- likes(mary,X), likes(john,X)

likes(mary,food).
likes(mary,Cola).
likes(john,Cola).
likes(john,mary).

Cola
Rules

• How to express that John likes all people?
• Listing all people?
  • likes(john, alfred).
  • likes(john, bertrand).
  • likes(john, charles).
  • likes(john, david).
  • etc.
• Not feasible. More compact way: Using rules. John likes any object provided it is a person.
Rule Examples

- Rules state Dependence:

  I use an umbrella if there is rain.

- Rules Define:

  X is a bird if X is an animal and X has feathers.
Formulating Rules

- John likes anyone who likes cola.
- John likes any Something if it likes cola.
- John likes X if X likes cola.
Rule Syntax

\[ \text{likes} (\text{joh}n, X) : \leftarrow \text{likes} (X, \text{Cola}) \]

- head
- rule delimiter (if)
- body
Variable Scope

The occurrences of X within a rule refer to the same object:

likes(john,X):- likes(X, cola), likes(X, food).
likes(john, mary):- likes(mary, cola), likes(mary, food).
likes(john, adam):- likes(adam, cola), likes(adam, food).
Royal Parents

Example

The parents of X are Y and Z.
Y is the mother.
Z is the father.

Database:

\text{male(albert).}
\text{male(edward).}
\text{female(alice).}
\text{female(victoria).}
\text{parents(edward,victoria,albert).}
\text{parents(alice,victoria,albert).}
Sisters

Example

X is a sister of Y if:

- X is female,
- X has parents M and F,
- Y has parents M and F.

Rule:

\[ \text{sister}(X,Y) : \neg \text{female}(X), \]
\[ \text{parents}(X,M,F), \]
\[ \text{parents}(Y,M,F). \]
Question:

?- sister(alice, edward).

The question (goal) matches the head of the rule, if one replaces X with alice and Y with edward.

The instance of the body becomes new goal: female(alice), parents(alice, M, F), parents(ederward, M, F).
Is Alice Edward’s Sister?

(1) male(albert).
(2) male(edward).
(3) female(alice).
(4) female(victoria).
(5) parents(edward,
    victoria, albert).
(6) parents(alice,
    victoria, albert).
(7) sister(X,Y):-
    female(X),
    parents(X,M,F),
    parents(Y,M,F).

7 sister(alice,edward)
    X0=alice,
    Y0=edward

3 female(alice),
    parents(alice,M0,F0),
    parents(edward,M0,F0).

6 parents(alice,M0,F0),
    parents(edward,M0,F0).

5 parents(edward,victoria,albert).

Introduction  2/12/2015
Is Alice Edward’s Sister?

(1) male(albert).
(2) male(Edward).
(3) female(alice).
(4) female(victoria).
(5) parents(Edward, victoria, albert).
(6) parents(alice, victoria, albert).
(7) sister(X,Y):-
    female(X),
    parents(X,M,F),
    parents(Y,M,F).

7 sister(alice,Edward)
    X0=alice, Y0=Edward

3 female(alice),
    parents(alice, M0, F0),
    parents(Edward, M0, F0).

6 parents(alice, M0, F0),
    parents(Edward, M0, F0).

5 parents(Edward, victoria, albert).

Introduction  2/12/2015
Who’s Sister Is Alice?

(1) male(albert).
(2) male(Edward).
(3) female(alice).
(4) female(victoria).
(5) parents(Edward, victoria, albert).
(6) parents(alice, victoria, albert).
(7) sister(X, Y):-
    female(X),
    parents(X, M, F),
    parents(Y, M, F).

7 sister(alice, X)
    ↓
X0=alice, Y0=X

3 female(alice),
    parents(alice, M0, F0),
    parents(X, M0, F0).

6 parents(alice, M0, F0),
    parents(X, M0, F0).
    ↓
M0=victoria
F0=albert

5 parents(X, victoria, albert).
  ↓
X=Edward

Answer: X = Edward.

Introduction 2/12/2015
Useful Links

- **SWI-Prolog:**

- **SWI-Prolog Editor (Windows, SWI-x86 only)**

- **Prolog mode for (X)Emacs:**
  - [http://turing.ubishops.ca/home/bruda/emacs-prolog/](http://turing.ubishops.ca/home/bruda/emacs-prolog/)

- **Prolog newsgroup:**
  - [http://groups.google.com/groups?group=comp.lang.prolog](http://groups.google.com/groups?group=comp.lang.prolog)

- **Links from SWI-Prolog Web page:**
  - [http://www.swi-prolog.org/Links.html](http://www.swi-prolog.org/Links.html)