Chapter 1

Introduction
Outline

- Introducing Prolog programs and queries
- Showing the advantages of declarative programming
- Illustrating shortcomings of Prolog
A Prolog Program

direct(frankfurt,san_fran_francisco).
direct(frankfurt,chicago).
direct(san_francisco,honolulu).
direct(honolulu,maui).

collection(X, Y) :- direct(X, Y).
collection(X, Y) :- direct(X, Z), collection(Z, Y).
Queries (I)

direct(frankfurt,san_francisco).
direct(frankfurt,chicago).
direct(san_francisco,honolulu).
direct(honolulu,maui).

collection(X, Y) :- direct(X, Y).
collection(X, Y) :- direct(X, Z), connection(Z, Y).

| ?- collection(frankfurt, maui).
yes
direct(frankfurt,san_francisco).
direct(frankfurt,chicago).
direct(san_francisco,honolulu).
direct(honolulu,maui).

connection(X, Y) :- direct(X, Y).
connection(X, Y) :- direct(X, Z), connection(Z, Y).

| ?- connection(san_francisco, X).  
X = honolulu ;  
X = maui ;  
no
Queries (III)

direct(frankfurt,san_francisco).
direct(frankfurt,chicago).
direct(san_francisco,honolulu).
direct(honolulu,maui).

connection(X, Y) :- direct(X, Y).
connection(X, Y) :- direct(X, Z), connection(Z, Y).

| ?- connection(maui, X).

no
An Important Data Structure: Lists

\[ a_1, \ldots, a_n \]

\[ \text{head} \mid \text{tail} \]

\[ \text{list} = [\text{apples} \mid [\text{pears, plums}]] \]

member(X, [X | List]).
member(X, [Y | List]) :- member(X, List).

member_both(X, L1, L2) :- member(X, L1), member(X, L2).

?– member_both(X, [apples, pears, plums], [peaches, plums, pears]).
X = pears ;
X = plums ;
no
An Imperative Program for Comparison

type List = array[1..n] of integer;
procedure members(a, b : List; var c : List; var x : integer);
    var i, j, k : integer;
begin
    k := 1;
    for i := 1 to n do
        for j := 1 to n do
            if a[i] = b[j] then begin
                c[k] := a[i]; k := k + 1
            end;
    x := k - 1
end;
Declarative Programs are Flexible

member(X, [X | List]).
member(X, [Y | List]) :- member(X, List).

member_both(X, L1, L2) :- member(X, L1), member(X, L2).

?- member_both(pears, [apples, pears, plums], [peaches, plums, pears]).
yes

?- member_both(apples, [apples, pears, plums], [peaches, X]).
X = apples
Declarative Programs are Flexible

\begin{align*}
\text{add}(X,0,X) &. \\
\text{add}(X,s(Y),s(Z)) &: \text{add}(X,Y,Z). \\
| ?- \text{add}(s(0),s(0),Z). \\
Z &= s(s(0)) \\
| ?- \text{add}(X,Y,s(s(0))). \\
X &= s(s(0)), \quad Y = 0; \\
X &= s(0), \quad Y = s(0); \\
X &= 0 \\
Y &= s(s(0))
\end{align*}

/* x + 0 = x */
/* x + y = z \rightarrow x + s(y) = s(z) */
Yet Another Declarative Specification

The square of 45 is 2025, and 20 + 25 is 45, isn't that strange? Find more pairs of numbers that exhibit this peculiarity!

```
solution(N, Z) :- between(1, 99, N),
                 Z is N*N,
                 Z >= 1000,
                 (Z // 100) + (Z mod 100) =:= N.
```

<table>
<thead>
<tr>
<th>?- solution(N, Z).</th>
</tr>
</thead>
<tbody>
<tr>
<td>N = 45, Z = 2025 ;</td>
</tr>
<tr>
<td>N = 55, Z = 3025 ;</td>
</tr>
<tr>
<td>N = 99, Z = 9801</td>
</tr>
</tbody>
</table>
Programming Languages

- Imperative Programming Languages
  - Declaration part defines possible states (of variables); statement part defines transformation on states
  - Close to von Neumann computer architecture
  - Description of how something is computed
  - Example: Java

- Declarative Programming Languages
  - Abstraction from states and state transformations
  - Direct formulation of mathematical objects (functions, relations, constraints)
  - Description of what is computed
  - Example: Prolog, Eclipse, Haskell, and Curry
Declarative Programming Languages

- Logic Programming Languages
  Example language: Prolog

- Constraint Logic Programming Languages
  Example language: Eclipse

- Functional Programming Languages
  Example language: Haskell

- Integrated (Functional-logic) Programming Languages
  Example language: Curry
Advantages of Declarative Programming

- Specifications are programs
- The computation mechanism is not part of the program
- “Thinking” declaratively is easier than “thinking” procedurally
- Declarative programs are therefore much simpler to understand, develop, and verify
- The output of a logic program is a logical consequence of the program
- Logic programs are flexible
Shortcomings of Prolog: Termination (I)

direct (frankfurt, san_francisco).
direct (frankfurt, chicago).
direct (san_francisco, honolulu).
direct (honolulu, maui).
direct (san_francisco, san_francisco).

collection(X, Y) :- direct(X, Y).
collection(X, Y) :- direct(X, Z), collection(Z, Y).

| ?- collection(san_francisco, X).
X = honolulu ;
X = san_francisco ;
X = maui ;
X = honolulu ;
...
Shortcomings of Prolog: Termination (II)

direct(san_francisco,san_francisco).
direct(frankfurt,san_francisco).
direct(frankfurt,chicago).
direct(san_francisco,honolulu).
direct(honolulu,maui).

collection(X, Y) :- direct(X, Y).
collection(X, Y) :- direct(X, Z), connection(Z, Y).

| ?- collection(san_francisco, X).
X = san_francisco ;
X = honolulu ;
X = san_francisco ;
X = honolulu ;
...
Shortcomings of Prolog: Termination (III)

\[
\text{direct}(\text{frankfurt}, \text{san}\_\text{francisco}). \\
\text{direct}(\text{frankfurt}, \text{chicago}). \\
\text{direct}(\text{san}\_\text{francisco}, \text{honolulu}). \\
\text{direct}(\text{honolulu}, \text{maui}). \\
\text{direct}(\text{san}\_\text{francisco}, \text{san}\_\text{francisco}).
\]

\[
\text{connection}(X, Y) :- \text{direct}(X, Z), \text{connection}(Z, Y).
\text{connection}(X, Y) :- \text{direct}(X, Y).
\]

| ?- \text{connection}(\text{san}\_\text{francisco}, X).  \\
X = \text{maui} ;  \\
X = \text{maui} ;  \\
X = \text{maui} ;  \\
...
Shortcomings of Prolog: Termination (IV)

direct(san_francisco,san_francisco).
direct(frankfurt,san_francisco).
direct(frankfurt,chicago).
direct(san_francisco,honolulu).
direct(honolulu,maui).

collection(X, Y) :- direct(X, Z), collection(Z, Y).
collection(X, Y) :- direct(X, Y).

? = collection(san_francisco, X).

?
Shortcomings of Prolog: “Occur Check” - Failure

A person $x$ and the mother of $x$ can never be the same.

```
mystery :- same_person(X, mother_of(X)).
same_person(Z, Z).

| ?- mystery.

yes
```
Shortcomings of Prolog: Is Prolog Truly Declarative?

This rule can only be “called” if all three arguments are numbers:

\[
\text{between}(X, Y, Z) : - \ X =< \ Z, \ Z =< Y.
\]

This is the “simplest” usable specification:

\[
\text{between} \ (X, Y, Z) : - \ X =< Y, \ Z \text{ is } X. \\
\text{between} \ (X, Y, Z) : - \ X < Y, \ X1 \text{ is } X+1, \ \text{between}(X1, Y, Z).
\]
How to Use a Prolog System (I)

% add-program in file add.pl:
add(X,0,X).
add(X,s(Y),s(Z)) :- add(X,Y,Z).

irz601:~> sicstus
SICStus 3 #5: Fri Nov 1 15:49:55 MET 1996
| ?- [add].
{consulting/usr/users/ith/ak15/add.pl...}
{/usr/users/ith/ak15/add.pl consulted, 0 msec 352 bytes}

yes
| ?- add(X,Y,s(s(0))).
How to Use a Prolog System (II)

\[
\begin{align*}
X &= s(s(0)), \\
Y &= 0 \ ? \ ; \\
\end{align*}
\]

\[
\begin{align*}
X &= s(0), \\
Y &= s(0) \ ? \ ; \\
\end{align*}
\]

\[
\begin{align*}
X &= 0, \\
Y &= s(s(0)) \ ? \ ; \\
\end{align*}
\]

no

| ?- halt. |
Objectives

- Introducing Prolog programs and queries
- Showing the advantages of declarative programming
- Illustrating shortcomings of Prolog