TECHNISCHE UNIVERSITÄT DRESDEN

Computational Logic ∴ Group

Hannes Strass (based on slides by Michael Thielscher)

Faculty of Computer Science, Institute of Artificial Intelligence, Computational Logic Group

Introduction

Lecture 1, 9th Oct 2023 // Foundations of Logic Programming, WS 2023/24

Overview

Prolog Programs

Queries

Advantages of Declarative Programs

Shortcomings of Prolog

How to Use a Prolog System

Prolog Programs

A Prolog Program

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

Facts

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

Rules

Queries

Queries (1)

```
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(frankfurt, maui).

yes
```

Queries (2)

```
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 (3)

```
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

$[a_1, \ldots, a_n]$ [apples,pears,plums]
$[head \mid tail]$ [apples | [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
```

Advantages of Declarative Programs

An Imperative Program for Comparison

```
function members(a: number[], b: number[]): number[] {

    let c: number[] = [];
    let i, j: number;

    for (i = 0; i < a.length; i += 1) {
        for (j = 0; j < b.length; j += 1) {
            if (a[i] == b[j]) {
                c.push(a[i]);
            }
        }
    }

    return c;
}
```

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

```
add(X,0,X).
add(X,s(Y),s(Z)) :- add(X,Y,Z).

| ?- add(s(0),s(0),Z).

Z = s(s(0))

| ?- add(X,Y,s(s(0))).

X = s(s(0)),
Y = 0 ;

X = s(0),
Y = s(0) ;

X = 0,
Y = s(s(0))
```

Yet Another Program

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.
```

```
| ?- solution(N, Z).

N = 45, Z = 2025 ;
N = 55, Z = 3025 ;
N = 99, Z = 9801
```

Quiz: Program and Queries

Quiz

Consider this program:

```
mother(mary, harry).        mother(edna, wilma).
father(eddy, harry).        father(eddy, wilma).

siblings(X, Y) :- mother(M, X), mother(M, Y).
siblings(X, Y) :- father(F, X), father(F, Y).

...
```

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 languages: C/C++, Java, Python
- ▷ Declarative Programming Languages
  - abstraction from states and state transformations
  - direct formulation of mathematical objects (functions, relations, constraints)
  - description of *what* is computed
  - Example languages: Haskell, Prolog, Scala, Curry, Eclipse Prolog

Declarative Programming Languages

- Functional Programming Languages
  Example language: Haskell
- Logic Programming Languages
  Example language: Prolog
- Integrated (Functional-logic) Programming Languages
  Example language: Curry
- Constraint Logic Programming Languages
  Example language: Eclipse Prolog

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, to develop, and to prove properties about.
- The output of a logic program is a logical consequence of the program.
- Flexibility of logic programs.

Shortcomings of Prolog

Shortcomings of Prolog: Termination (1)

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

direct(san_francisco,san_francisco).

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

| ?- connection(san_francisco, X).
X = honolulu ;
X = san_francisco ;
X = maui ;
X = honolulu ;
...
```

Shortcomings of Prolog: Termination (2)

```
direct(san_francisco,san_francisco).

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 = san_francisco ;
X = honolulu ;
X = san_francisco ;
X = honolulu ;
...
```

Shortcomings of Prolog: Termination (3)

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

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

| ?- connection(san_francisco, X).
X = maui ;
X = maui ;
X = maui ;
...
```

Shortcomings of Prolog: Termination (4)

```
direct(san_francisco,san_francisco).

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

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


| ?- connection(san_francisco, X).
```

?

Shortcomings of Prolog: Occur Check

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 it truly declarative?

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

```
between(X, Y, Z) :- X =< Z, Z =< Y.
```

This is the “simplest” usable specification:

```
between(X, Y, Z) :- X =< Y, Z is X.
between(X, Y, Z) :- X < Y, X1 is X+1, between(X1, Y, Z).
```

How to Use a Prolog System

How to Use a Prolog System (1)

```
irz601: > cat 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 (2)

```
X = s(s(0)),
Y = 0 ? ;

X = s(0),
Y = s(0) ? ;

X = 0,
Y = s(s(0)) ? ;

no
| ?- halt.
```

Conclusion

Summary

- Prolog programs consist of **facts** and **rules**.
- We use Prolog by asking **queries** to programs.
- Answers to queries can be Boolean (yes/no) …
- … or given by variable assignments.
- Prolog programs are **declarative** (to a certain extent).

Suggested action points:

- Prolog interpreters are freely available – install one now, e.g. on ubuntu:

```
sudo apt install swi-prolog
```

- … or other platforms: `https://www.swi-prolog.org/download/stable`
- Try out the examples of this lecture.