Outline

Prolog

Running Prolog at the school:

1. boot a computer using Linux;
2. start a terminal;
3. type `sicstus` in it

and the Sictus Prolog will start.
Running Prolog on your home computer/laptop:

- Download SWI Prolog and follow instructions.

Example of a Prolog program

- Terms:
  - constants, variables (start with an upper-case letter or an underscore);
  - compound: name(arg_1,. . . ,arg_k);
  - Ground terms: terms with no variables
- Clauses:
  - Rules: Head :- $Goal_1$, . . . , $Goal_k$.
  - Facts: Head.
    i.e. a rule without any goals or body
  - Goals: :- $Goal_1$, . . . , $Goal_k$.
    i.e. a rule without a head.

Examples of clauses

```
parent(john,juliet).
:- parent(john,X).
parent(X,juliet).
greater_than(succ(X),zero).
```

Predicate definition: collection of rules with the same predicate (head name).

```
ancestor(X,Y) :- mother(X,Y).
ancestor(X,Y) :- father(X,Y).
ancestor(X,Y) :- ancestor(X,Z), ancestor(Z,Y).
...
```

Examples of clauses

```
parent(john,juliet).
:- parent(john,X).
parent(X,juliet).
greater_than(succ(X),zero).
```

Predicate definition: collection of rules with the same predicate (head name).

```
ancestor(X,Y) :- mother(X,Y).
ancestor(X,Y) :- father(X,Y).
ancestor(X,Y) :- ancestor(X,Z), ancestor(Z,Y).
...
```

Examples of clauses

```
parent(john,juliet).
:- parent(john,X).
parent(X,juliet).
greater_than(succ(X),zero).
```

Predicate definition: collection of rules with the same predicate (head name).

```
ancestor(X,Y) :- mother(X,Y).
ancestor(X,Y) :- father(X,Y).
ancestor(X,Y) :- ancestor(X,Z), ancestor(Z,Y).
...
```

Rules

Meaning of a rule `Head :- Goal`$_1$`,...,Goal`$_n$:

- If `Goal`$_1$ and `Goal`$_2$ and ... and `Goal`$_k$ all hold, then `Head` holds.

Program: a sequence of clauses.

```
ancestor(X,Y) :- father(X,Y).
father(X,Y) :- parent(X,Y), male(X).
parent(john,juliet).
male(john).
```

Queries:

```
:- ancestor(john,juliet).
:- father(john,juliet).
:- parent(john,juliet),male(john).
```

Rules

Meaning of a rule `Head :- Goal`$_1$`,...,Goal`$_n$:

- If `Goal`$_1$ and `Goal`$_2$ and ... and `Goal`$_k$ all hold, then `Head` holds.

Program: a sequence of clauses.

```
ancestor(X,Y) :- father(X,Y).
father(X,Y) :- parent(X,Y), male(X).
parent(john,juliet).
male(john).
```

Queries:

```
:- ancestor(john,juliet).
:- father(john,juliet).
:- parent(john,juliet),male(john).
```

Rules

Meaning of a rule `Head :- Goal`$_1$`,...,Goal`$_n$:

- If `Goal`$_1$ and `Goal`$_2$ and ... and `Goal`$_k$ all hold, then `Head` holds.

Program: a sequence of clauses.

```
ancestor(X,Y) :- father(X,Y).
father(X,Y) :- parent(X,Y), male(X).
parent(john,juliet).
male(john).
```

Queries:

```
:- ancestor(john,juliet).
:- father(john,juliet).
:- parent(john,juliet),male(john).
```

Prolog program with recursion

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z),ancestor(Z,Y).
parent(chaz,john).
parent(john,juliet).
:- ancestor(chaz,juliet).
:- parent(chaz,john),ancestor(john,juliet).
:- ancestor(john,juliet).
:- parent(john,juliet).
```

Goal with variables: find a substitution for the variables that makes this goal derivable:

```
:- ancestor(chaz,X).
```

Prolog program with recursion

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z),ancestor(Z,Y).
parent(chaz,john).
parent(john,juliet).
:- ancestor(chaz,juliet).
:- parent(chaz,john),ancestor(john,juliet).
:- ancestor(john,juliet).
:- parent(john,juliet).
```

Goal with variables: find a substitution for the variables that makes this goal derivable:

```
:- ancestor(chaz,X).
```

How does Prolog answer goals?

Search Strategy: process subgoals left-to-right, top-to-bottom (but see later. . .)

Use the trace facility of Prolog.

How does Prolog answer goals?

Search Strategy: process subgoals left-to-right, top-to-bottom (but see later. . .)

Use the trace facility of Prolog.

Built-in predicates

Built-in predicates which perform evaluation:

- operators: `+`, `*`, `-`, `/`
- comparison: `<`, `>`, `<=`, `>=`
- equality, inequality: `=`, `==`, `\==`
  - `X = 2 * 3 * 7`
  - `42 = 2 * 3 * 7`
- invoke evaluation:
  - `42 is 2 * 3 * 7`
  - `X is 2 * 3 * 7`