# CSc 372

# Comparative Programming Languages

## 18 : Prolog — Basics

Department of Computer Science
University of Arizona

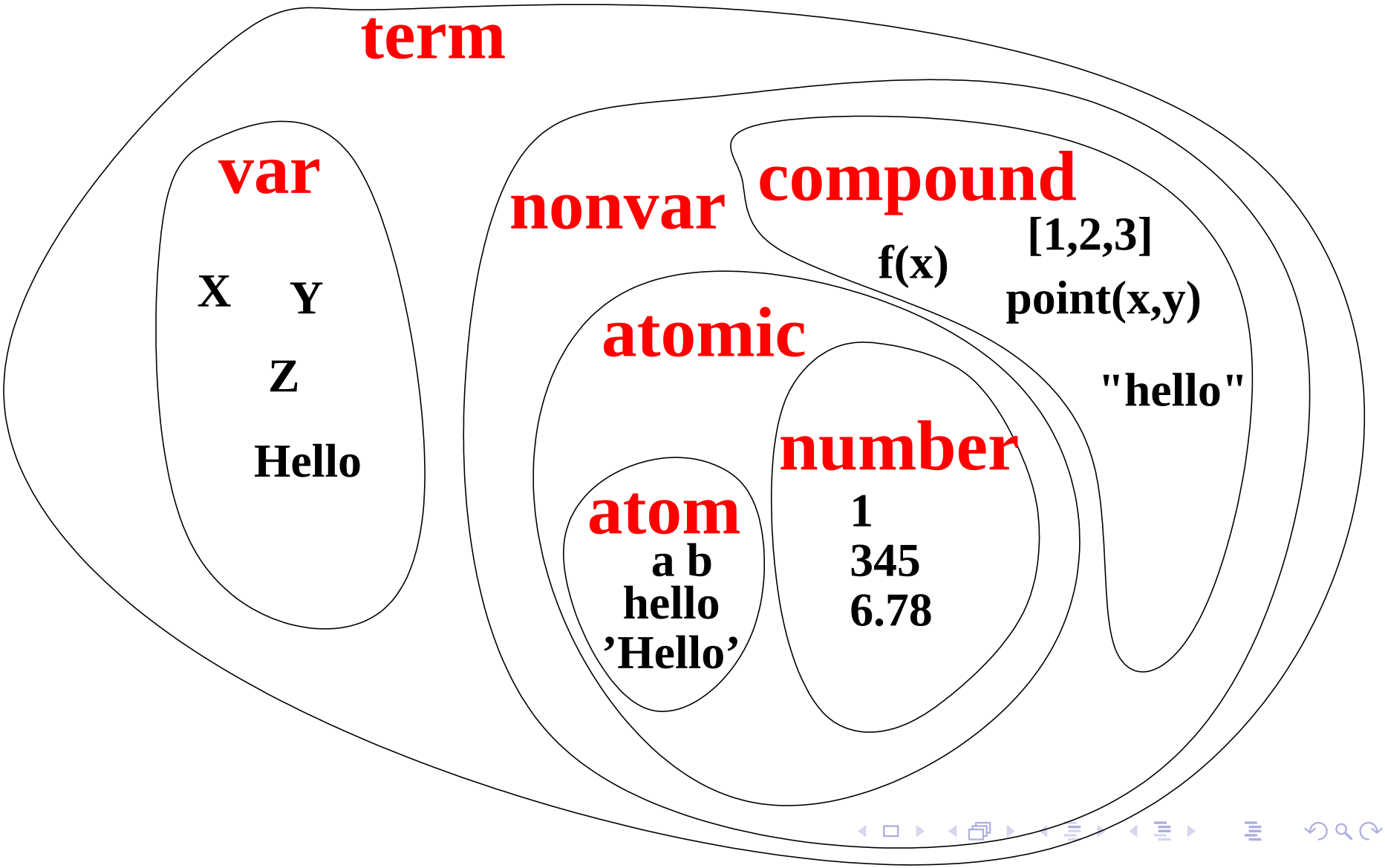
collberg@gmail.com

Copyright © 2011 Christian Collberg

# Prolog Types

- The term is Prolog's basic data structure.
- Everything is expressed in the form of a term. This includes programs and data.
- Prolog has four basic types of terms:
  1. variables start with an uppercase letter;
  2. compound terms are lists, strings, and structures;
  3. atoms start with a lower-case letter;
  4. numbers.

# Prolog Types...



# Prolog Numbers

- Most Prolog implementations support infinite precision integers. This is not true of GNU Prolog!
- The built-in operator is evaluates arithmetic expressions:

```
| ?- X is 6*7.
X = 42
| ?- X is 6.0*7.0.
X = 42.0
| ?- X is 60000000000000*700000000000000.
X = 1
```

# Prolog Arithmetic Expressions

- An infix expression is just shorthand for a structure:

```
| ?- X = +(1,*(2,3)).
X = 1+2*3
| ?- X = 1+2*3.
X = 1+2*3
| ?- X is +(1,*(2,3)).
X = 7
| ?- X is 1+2*3.
X = 7
```

- X = 1*2 means "make the variable X and 1*2 the same". It looks like an assignment, but it's what we call unification. More about that later.

# Prolog Atoms

- Atoms are similar to `enums` in C.
- Atoms start with a lower-case letter and can contain letters, digits, and underscore (_).

```
| ?- X = hello.
X = hello
| ?- X = hE_l_l_o99.
X = hE_l_l_o99
```

# Prolog Variables

- Variables start out uninstantiated, i.e. without a value.
- Uninstantiated variables are written _number:

```
| ?- write(X).
_16
```

- Once a Prolog variable has been instantiated (given a value), it will keep that value.

```
| ?- X=sally.
X = sally
| ?- X=sally, X=lisa.
no
```

# Prolog Variables. . .

- When a program backtracks over a variable instantiation, the variable again becomes uninstantiated.

```
| ?- (X=sally; X=lisa), write(X), nl.
sally
X = sally ?  ;

lisa
X = lisa
```

# Prolog Programs

- A Prolog program consists of a database of facts and rules:

```
likes(lisa,chocolate).
likes(lisa,X) :- tastes_like_chocolate(X).
```

- :- is read if.
- :- is just an operator, like other Prolog operators. The following are equivalent:

```
likes(lisa,X) :- boy(X),tastes_like_choc(X).

:-(likes(lisa,X),
   (boy(X),tastes_like_chok(X))).
```

# Prolog Programs...

- Prolog facts/rules can be overloaded, wrt their arity.
- You can have a both a rule `foo()` and a rule `foo(X)`:

```
| ?- [user].
foo.
foo(hello).
foo(bar,world).
foo(X,Y,Z) :-
    Z is X + Y.
<ctrl-D>
```

```
| ?- foo.
yes
| ?- foo(X).
X = hello
| ?- foo(X,Y).
X = bar
Y = world
| ?- foo(1,2,Z).
Z = 3
```

# Standard predicates

- `read(X)` and `write(X)` read and write Prolog terms.
- `nl` prints a newline character.

```
| ?- write(hello),nl.
hello

| ?- read(X), write(X), nl.
hello.
hello
```

# Standard predicates...

- `write` can write arbitrary Prolog terms:

```
| ?- write(hello(world)),nl.
hello(world)
```

- Note that `read(X)` requires the input to be syntactically correct and to end with a period.

```
| ?- read(X).
foo).
uncaught exception:  error
```

# Unification/Matching

- The `=`-operator tries to make its left and right-hand sides the same.
- This is called unification or matching.
- If Prolog can't make `X` and `Y` the same in `X = Y`, matching will fail.

```
| ?- X=lisa, Y=sally, X = Y.
no
| ?- X=lisa, Y=lisa, Z = X, Z = Y.
X = lisa
Y = lisa
Z = lisa
```

- We will talk about this much more later.

# Backtracking

- Prolog will try every possible way to satisfy a query.
- Prolog explores the search space by using backtracking, which means undoing previous computations, and exploring a different search path.

# Backtracking. . .

- Here's an example:

```
| ?- [user].
girl(sally).
girl(lisa).
pretty(lisa).
blonde(sally).
| ?- girl(X),pretty(X).
X = lisa
| ?- girl(X),pretty(X),blonde(X).
no
| ?- (X=lisa; X=sally), pretty(X).
X = lisa
```

- We will talk about this much more later.

# Māori Family Relationships

John Foster (in *He Whakamaarama – A New Course in Māori*) writes:

> *Relationship is very important to the Māori. Social seniority is claimed by those able to trace their whakapapa or genealogy in the most direct way to illustrious ancestors. Rights to shares in land and entitlement to speak on the marae may also depend on relationship. Because of this, there are special words to indicate elder or younger relations, or senior or younger branches of a family.*

- Māori is the indigenous language spoken in New Zealand. It is a polynesian language, and closely related to the language spoken in Hawaii.

# Māori Terms of Address

| Māori | English |
|---|---|
| au | I |
| tipuna, tupuna | grandfather, grandmother, grandparent, ancestor |
| tiipuna | grandparents |
| matua taane | father |
| maatua | parents |
| paapaa | father |
| whaea, maamaa | mother |
| whaea kee | aunt |
| kuia | grandmother, old lady |
| tuakana | older brother of a man, older sister of a woman |
| teina | younger brother of a man, younger sister of a woman |

# Māori Terms of Address...

| Māori | English |
|---|---|
| tungaane | woman's brother (older or younger) |
| tuahine | man's sister (older or younger) |
| kaumaatua | elder (male) |
| mokopuna | grandchild (male or female) |
| iraamutu | niece, nephew |
| taane | husband, man |
| hunaonga | daughter-in-law, son-in-law |
| tamaahine | daughter |
| tama | son |
| tamaiti | child (male or female) |
| tamariki | children |
| wahine | wife, woman |
| maataamua | oldest child |

# Māori Terms of Address...

| Māori | English |
|---|---|
| pootiki | youngest child |
| koroheke, koro, koroua | old man |
| whaiapo | boyfriend, girlfriend[1] |
| kootiro | girl |
| tamaiti taane | boy |
| whanaunga | relatives |

[1] Literally: "What you follow at night"

# The Whanau

- A program to translate between English and Māori must take into account the differences in terms of address between the two languages.
- Write a Prolog predicate `calls(X,Y,Z)` which, given a database of family relationships, returns **all** the words that `X` can use to address or talk about `Y`.

```
?- calls(aanaru, hata, Z).
   Z = tuakana ;
   Z = maataamua ;
   no

?- calls(aanaru, rapeta, Z).
   Z = teina ;
   no
```

# The Whanau...

- Whanau is Māori for family.
- Below is a table showing an extended Māori family.

| Name | Sex | Father | Mother | Spouse | Born |
|---|---|---|---|---|---|
| Hoone | male | unknown | unknown | Rita | 1910 |
| Rita | female | unknown | unknown | Hone | 1915 |
| Ranginui | male | unknown | unknown | Reremoana | 1915 |
| Reremoana | female | unknown | unknown | Ranginui | 1916 |
| Rewi | male | Hoone | Rita | Rahia | 1935 |
| Rahia | female | Ranginui | Reremoana | Rewi | 1940 |
| Hata | male | Rewi | Rahia | none | 1957 |
| Kiri | female | Rewi | Rahia | none | 1959 |

# The Whanau...

| Name | Sex | Father | Mother | Spouse | Born |
|---|---|---|---|---|---|
| Hiniera | female | Rewi | Rahia | Pita | 1960 |
| Aanaru | male | Rewi | Rahia | none | 1962 |
| Rapeta | male | Rewi | Rahia | none | 1964 |
| Mere | female | Rewi | Rahia | none | 1965 |
| Pita | male | unknown | unknown | Hiniera | 1960 |
| Moeraa | female | Pita | Hiniera | none | 1986 |
| Huia | female | Pita | Hiniera | none | 1987 |
| Irihaapeti | female | Pita | Hiniera | none | 1988 |

# The Whanau Program — Database Facts

- We start by encoding the family as facts in the Prolog database.

```
% person(name,   sex,      father,mother,spouse, birth-year).

person(hoone,    male,     unkn1, unkn5, rita,      1910).
person(rita,     female,   unkn2, unkn6, hoone,     1915).
person(ranginui,male,      unkn3, unkn7, reremoana,1915).
person(reremoana, female,unkn4, unkn8, ranginui, 1916).

person(rewi,     male,     hoone, rita, reremoana, 1935).
person(rahia,    female,ranginui,reremoana, rita, 1916).

person(hata,     male,     rewi,  rahia,  none,     1957).
person(kiri,     female,   rewi,  rahia   none,     1959).
```

# The Whanau Program — Database Facts...

```
% person(name,  sex,     father,mother,spouse,   birth-year)
person(hiniera, female,  rewi,  rahia,  pita,    1960).
person(anaru,   male,    rewi,  rahia,  none,    1962).
person(rapeta,  male,    rewi,  rahia,  none,    1964).
person(mere,    female,  rewi,  rahia,  none,    1965).
person(pita,    male,    unkn9, unkn10, hiniera,1960).

person(moeraa,  female,   hiniera,  pita, none, 1986).
person(huia,    female,   hiniera,  pita, none, 1987).
person(irihaapeti, female,hiniera,  pita, none, 1988).
```

# Whanau — Auxiliary predicates

- We introduce some auxiliary predicates to extract information from the database.

```
% Auxiliary predicates
gender(X, G) :- person(X, G, _, _, _, _).
othergender(male, female).
othergender(female, male).
female(X) :- gender(X, female).
male(X)   :- gender(X, male).
```

# Whanau — Family Relationships

- We next write some predicates that computes common family relationships.

```
% Is Y the <operator> of X?
wife(X, Y)     :- person(X, male, _, _, Y, _).
husband(X, Y)  :- person(X, female, _, _, Y, _).
spouse(X, Y)   :- wife(X, Y).
spouse(X, Y)   :- husband(X, Y).
parent(X, Y)   :- person(X, _,Y, _, _, _).
parent(X, Y)   :- person(X, _,  _, Y, _, _).
son(X, Y)      :- person(Y, male, X, _, _, _).
son(X, Y)      :- person(Y, male, _, X, _, _).
daughter(X, Y):- person(Y, female, X, _, _, _).
daughter(X, Y):- person(Y, female, _, X, _, _).
child(X, Y)    :- son(X, Y).
child(X, Y)    :- daughter(X, Y)
```

# Whanau — Family Relationships...

- Some of the following are left as an exercise:

```
% Is X older than Y?
older(X,Y) :-
      person(X, _, _, _, _,Xyear),
      person(Y, _, _, _, _,Yyear),
      Yyear > Xyear.

% Is Y a sibling of X of the gender G?
sibling(X, Y, G) :- <left as an exercise>.

% Is Y one of X's older siblings of gender G?
oldersibling(X,Y,G) :- <left as an exercise>.

% Is Y one of X's older/younger siblings of either gender?
oldersibling(X,Y) :- <left as an exercise>.
```

# Whanau — Family Relationships...

```
youngersibling(X,Y) :- <left as an exercise>.

% Is Y an ancestor of X of gender G?
ancestor(X,Y,G) :- <left as an exercise>.

% Is Y an older relative of X of gender G?
olderrelative(X,Y,G) :-
   ancestor(X, Y, G).
olderrelative(X,Y,G) :-
   ancestor(X, Z, _),
   sibling(Y, Z, G).

% Is Y a sibling of X of his/her opposite gender?
siblingofothersex(X, Y) :- <left as an exercise>.
```

# The Whanau Program — Calls

- We can now finally write the predicate `calls(X,Y,T)` which computes all the ways `T` in which `X` can address `Y`.

```
% Me.
calls(X, X, au).

% Parents.
calls(X,Y,paapaa) :- person(X, _,Y, _, _, _).
calls(X,Y,maamaa) :- person(X, _, _,Y, _, _).

% Oldest/youngest sibling of same sex.
calls(X, Y, tuakana) :-
   gender(X, G), eldestsibling(X, Y, G).
calls(X, Y, teina) :-
   gender(X, G), youngestsibling(X, Y, G).
```

# The Whanau Program — Calls...

```
% Siblings of other sex.
calls(X, Y, tungaane) :- <left as an exercise>.
calls(X, Y, tuahine) :- <left as an exercise>.
calls(X, Y, tipuna) :-  <left as an exercise>.

% Sons and daughters.
calls(X, Y, tama)   :-  <left as an exercise>.
calls(X, Y, tamahine) :-  <left as an exercise>.

% Oldest/youngest child.
calls(X, Y, maataamua) :- <left as an exercise>.
calls(X, Y, pootiki) :- <left as an exercise>.

% Child-in-law.
calls(X, Y, hunaonga) :-  <left as an exercise>.

% Grandchild
```

# Readings and References

- Read Clocksin-Mellish, Chapter 2.

# Summary

# Prolog So Far

- Prolog terms:
  - atoms (`a, 1, 3.14`)
  - structures

    `guitar(ovation, 1111, 1975)`
- Infix expressions are abbreviations of "normal" Prolog terms:

| infix | prefix |
|---|---|
| `a + b` | `+(a, b)` |
| `a + b* c` | `+(a, *(b, c))` |