Prologue to Prolog 101
A Lecture for COSC-6421

Parke Godfrey
Fall 2004

Prologue

Goal: Convert you to the way of Prolog, especially you Lisp heathens (*or to introduce you to Prolog and its many merits*).

I. The Genesis of Prolog

II. Prolog, the language

III. The Merits of Prolog

IV. Why Prolog?

- **A.** Prolog vs. Lisp

- **B.** Why Prolog for AI?

V. The Cannibals-and-Missionaries Problem

VI. Homework

Theorem Proving

$\neg a \vee b$	$\neg b \vee \neg f \vee h$
$\neg a \vee c$	$\neg c \vee \neg d \vee h$
$\neg b \vee d \vee e$	$\neg e \vee \neg g \vee h$
$\neg c \vee f \vee g$	a

Prove h.

Search can be hard. Theorem proving can be hard.

A *Horn clause* has one or no positive atoms in it.

$$a \vee \neg b \vee \neg c$$

can be rewritten as

$$a \leftarrow b, c.$$

`Procedural = Declarative`

Logic can be used as a programming language!

Prolog, the language

1. Clauses, Facts, and Queries

Clause: $a \leftarrow b\_1, \ldots, b\_n.$

Fact: $a.$

Query: $\leftarrow a\_1, \ldots, a\_n.$

2. Matching (unification)

3. Built-in control

- Proof by refutation
- One inference rule: resolution
- Choosing clauses: first in list to match to last in list to match
- Choosing goals: from left-to-right in goal list

4. Meta-predicates

`setof`	`clause`	`var`
`assert`	`retract`	`not` “\+”
`univ` “=..”	`equivalent` “==”	*meta-variables!*

5. Search Prunning/Commit

`cut` “!”

Grandmothers and Grandfathers

$$\begin{aligned} \mathit{grandmother}\,(GM,\ X) \leftarrow\ & \mathit{mother}\,(GM,\ P), \\ & \mathit{parent}\,(P,\ X). \end{aligned}$$

$$\begin{aligned} \mathit{grandfather}\,(GF,\ X) \leftarrow\ & \mathit{father}\,(GF,\ P), \\ & \mathit{parent}\,(P,\ X). \end{aligned}$$

$$\begin{aligned} &\mathit{parent}\,(M,\ X) \leftarrow \mathit{mother}\,(M,\ X). \\ &\mathit{parent}\,(F,\ X) \leftarrow \mathit{father}\,(F,\ X). \end{aligned}$$

$$\begin{array}{ll} \mathit{mother}\,(\mathit{judith},\ \mathit{parke}). & \mathit{father}\,(\mathit{blan},\ \mathit{parke}). \\ \mathit{mother}\,(\mathit{ruby},\ \mathit{judith}). & \mathit{father}\,(\mathit{alvin},\ \mathit{judith}). \\ \mathit{mother}\,(\mathit{lallage},\ \mathit{blan}). & \mathit{father}\,(\mathit{albert},\ \mathit{blan}). \end{array}$$

$$\begin{array}{ll} \leftarrow \mathit{grandmother}\,(G,\ \mathit{parke}). & \leftarrow \mathit{grandmother}\,(\mathit{lallage},\ X). \end{array}$$

$$\begin{array}{ll} G = \mathit{ruby}; & X = \mathit{parke}; \\ G = \mathit{lallage}; & \mathit{no} \\ \mathit{no} & \end{array}$$

Why Prolog?

Prolog vs. Lisp (a sibling rivalry)

- the not-invented-here syndrom
- relational vs. functional

Why Prolog for AI?

- easy to write meta-programs
  - Prolog is its own meta-language!
  - `code = data`
- is an “interpreted” language
  - good debugging facilities
  - needed for meta-programming
- based on the recursion paradigm
- no typing!
- Prolog is based on first-order logic

  `Logic is good for AI.`
- is *declarative*

  (not prescriptive)

The Merits of Prolog

Neat Features of Prolog

- **Non-determinism (backtracking)**
  - Can find alternate answers/solutions for free!
- **Invertability**
  - Call any predicate with any instantiation pattern!

    (Well, sometimes . . .)
- **Unification**
  - Pattern matching for free!
- **Built-in Search**
  - A free refutation proof system.
  - Specs *are* executable. (Well, kind of . . .)

    Do not have to write one's own search mechanism for every problem.
- **Built-in database features**
  - `assert` and `retract`

Meta-Predicates

a.k.a. Extra-Logical Predicates

setof/findall

$\leftarrow \mathit{setof}\,(GM,\ \mathit{grandmother}\,(GM,\ \mathit{parke}),\ GMs).$

$GMs = [\mathit{lallage},\ \mathit{ruby}];$

no

assert

$\leftarrow \mathit{student}\,(X).$

no

$\leftarrow \mathit{assert}\,(\mathit{student}\,(\mathit{parke})).$

yes

$\leftarrow \mathit{student}\,(X).$

$X = \mathit{parke};$

no

meta-variables

$\mathit{exec\_list}\,([X|Xs]) \leftarrow X,\ \mathit{exec\_list}\,(Xs).$

$\mathit{exec\_list}\,([\,]).$

Executable Specifications

`Program = Logic + Control`

A goal of logic programming is to be able to execute specifications as code.

In Prolog, the *control* mechanism is built in.

Problem with Specs

`Some specs are more equal than others.`

$$\begin{aligned} \mathit{sort}\,(As,\ Zs) \leftarrow\ & \mathit{same\_length}\,(As,\ Zs),\\ & \mathit{perm}\,(As,\ Zs),\\ & \mathit{ordered}\,(Zs). \end{aligned}$$

$$\begin{aligned} \mathit{perm}\,(As,\ [A\,|Zs]) \leftarrow\ & \mathit{choose}\,(A,\ As,\ \mathit{Rest}),\\ & \mathit{perm}\,(\mathit{Rest},\ Zs). \end{aligned}$$
$$\mathit{perm}\,([\,],\ [\,]).$$

$$\mathit{same\_length}\,([\_\,|As],\ [\_\,|Zs]) \leftarrow \mathit{same\_length}\,(As,\ Zs).$$
$$\mathit{same\_length}\,([\,],\ [\,]).$$

$$\mathit{choose}\,(A,\ [A\,|As],\ As).$$
$$\mathit{choose}\,(A,\ [B\,|As],\ [B\,|Zs]) \leftarrow \mathit{choose}\,(A,\ As,\ Zs).$$

$$\mathit{ordered}\,([A,\ B\,|As]) \leftarrow A < B,\ \mathit{ordered}\,([B\,|As]).$$
$$\mathit{ordered}\,([A]).$$
$$\mathit{ordered}\,([\,]).$$

Problem with Specs [cont.]

A better sort of sort.

```
sort ([A|As], Zs) ← divide_list (A, As, Fs, Ls),
                    sort (Fs, OrdFs),
                    sort (Ls, OrdLs),
                    append (OrdFs, [A|OrdLs], Zs).
sort ([], []).

divide_list (A, [F|As], [F|Fs], Ls) ←
    A > F,
    divide_list (A, As, Fs, Ls).
divide_list (A, [L|As], Fs, [L|Ls]) ←
    A =< L,
    divide_list (A, As, Fs, Ls).
divide_list (A, [], [], []).
```

Pragmatics

\+	is	*not*	
,	is	*and*	
;	is	*or*	(also used to enumerate answers)
!	is	*cut*	
:-	is	*if*	(←)

[*Head*|*Tail*] is a list.

Head is the first term in list. (*car* for you Lispites)

Tail is the first term in list. (*cdr* for you Lispites)

[*First*, *Second*|*Tail*] is valid notation too. [] is the empty list.

[*First*, *Second*, *Third*] is a completely enumerated list.

Variables names always start CAPITALIZED.

Constants begin with lowercase (or are single-quoted).

How do you load clauses from a file?

In the Prolog session, type: *consult* (⟨*filename*⟩).

Every clause (rule, query, or fact) must end in a period!

Books on Prolog

Prolog Books (On reserve in AVW Library)

[1] W. F. Clocksin and C. S. Mellish. *Programming in Prolog.* Springer-Verlag, Berlin, third, revised and extened edition, 1987.

[2] L.S. Sterling and E.Y. Shapiro. *The Art of Prolog.* MIT Press, 1986.

Manuals

The *SICSTUS* Manual.

Logic for Problem Solving

[1] R.A. Kowalski. *Logic for Problem Solving.* Artificial Intelligence Series. North-Holland, New York, 1979.

[2] Nils J. Nilsson. *Principles of Artificial Intelligence.* Morgan Kaufmann Publishers Incorporated, 1980.

Books on Logic Programming

[1] John W. Lloyd. *Foundations of Logic Programming.* Symbolic Computation—Artificial Intelligence. Springer-Verlag, Berlin, second edition, 1987.

[2] Jorge Lobo, Jack Minker, and Arcot Rajasekar. *Foundations of Disjunctive Logic Programming.* M.I.T. Press, Cambridge, Massachusetts, 1992.