Logic Programming

Introduction

Michael Genesereth
Computer Science Department
Stanford University
Logic Programming is a style of programming in which programs take the form of sentences in the language of Symbolic Logic.

Logic Program is a collection of such sentences.

Logic Programming Language is a language for writing such programs.

Logic Programming System is a computer system that manages the creation, modification, and/or execution of logic programs.
Imperative Programming

![Diagram showing inputs, interpreter, and outputs]

**Inputs** → **Interpreter** → **Outputs**
A triangle is a polygon with 3 sides.

\[ e=mc^2 \]
Runnable Specifications

A declarative program is basically a runnable specification.

**Specification**
- What we believe about the *application area*
- What we want to know or to achieve in *application area*
- With no arbitrary decisions
- With no concern for internal processing details

**Runnable**
- Can be directly *interpreted*
- Can be *compiled* into traditional programs
  (Think automatic programming.)
Easier to create and modify than traditional programs.

Programmers can get by with little or no knowledge of the capabilities of systems executing those programs.

There is no need to make arbitrary choices.

Easier to learn logic programming than traditional programming.

Oddly, expert computer programmers often have more trouble with logic programming than novices.
Composability

No arbitrary choices

Computer can union multiple programs.

Computer can optimize programs.
Agility
Definition

A person $X$ is the grandparent of a person $Z$ if and only if there is a person $Y$ such that $X$ is the parent of $Y$ and $Y$ is the parent of $Z$.

Uses

Determine whether art is the parent of Cal.
Determine all of the grandchildren of Art.
Compute the grandparents of Cal.
Compute all grandparent-grandchildren pairs.
McCarthy’s Example of Versatility
Why Logic

Language
General / Domain-independent - no built-in assumptions
Highly expressive

Other data languages are easier for humans to use
Other definitional languages exist
But all can be converted to logical statements

Interpreter
Automated Reasoners capable of drawing conclusions
Can take advantage of domain-dependent reasoners
Logic

init(cell(1,1,b))
init(cell(1,2,b))
init(cell(1,3,b))
init(cell(2,1,b))
init(cell(2,2,b))
init(cell(2,3,b))
init(cell(3,1,b))
init(cell(3,2,b))
init(cell(3,3,b))
init(control(x))

legal(P,mark(X,Y)) :-
true(cell(X,Y,b)) &
true(control(P))
legal(x,noop) :-
true(control(o))
legal(o,noop) :-
ture(control(x))

next(cell(M,N,P)) :-
does(P,mark(M,N))
next(cell(M,N,Z)) :-
does(P,mark(M,N)) &
true(cell(M,N,Z)) & Z#b
next(cell(M,N,b)) :-
does(P,mark(J,K)) &
true(cell(M,N,b)) &
(M#J | N#K)
next(control(x)) :-
true(control(o))
next(control(o)) :-
true(control(x))
terminal :- line(P)
terminal :- ~open

goal(x,100) :- line(x)
goal(x,50) :- draw
goal(x,0) :- line(o)
goal(o,100) :- line(o)
goal(o,50) :- draw
goal(o,0) :- line(x)

diagonal(P) :-
true(cell(1,1,P)) &
true(cell(2,2,P)) &
true(cell(3,3,P))
diagonal(P) :-
true(cell(1,3,P)) &
true(cell(2,2,P)) &
true(cell(3,1,P))
diagonal(P) :-
true(cell(1,3,P)) &
true(cell(2,2,P)) &
true(cell(3,1,P))
line(P) :- row(M,P)
line(P) :- column(N,P)
line(P) :- diagonal(P)
open :- true(cell(M,N,b))
draw :- ~line(x) &
~line(o)
Applications
Circuit:

Premises:

\[ o \iff (x \land \neg y) \lor (\neg x \land y) \]

\[ a \iff z \land o \]

\[ b \iff x \land y \]

\[ s \iff (o \land \neg z) \lor (\neg o \land z) \]

\[ c \iff a \lor b \]

Applications:

Simulation

Configuration

Diagnosis

Test Generation
### Constraint Satisfaction

<table>
<thead>
<tr>
<th></th>
<th>owners</th>
<th>broods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anita</td>
<td>Barbara</td>
<td>Douglas</td>
</tr>
<tr>
<td>2006</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>2007</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>2008</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>2009</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>2010</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Max</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Molly</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Riley</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Samson</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Shadow</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

#### Sudoku Puzzle

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>1</th>
<th></th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
q(X) :- p(X,Y) \land p(X,Z) \land Y!\neq Z

\[ g(X,Z) :- p(X,Y) \land p(Y,Z) \]

\[ \text{illegal} :- p(X,Y) \land p(Y,X) \]
Data Integration

Integrated Search

Side-by-side Comparison

Integration Engine

Supplier 1
Supplier 2
Supplier 3
Supplier 4

Manufacturer 1
Manufacturer 2

Marketplace Data

Satisfaction Ratings

Product analysis
Business Rules
Computational Law is that branch of legal informatics concerned with the mechanization of legal reasoning.

Automated Legal Reasoning Systems
Legal analysis of specific cases
Planning for compliance in specific cases
Analysis of regulations for overlap, consistency, etc.

http://build.symbium.com
General Game Playing

http://ggp.stanford.edu
Theorem Proving

PTTP

means

Prolog Technology Theorem Prover

by acronymsandslang.com
Japan’s Fifth Generation Project

THE FIFTH GENERATION
ARTIFICIAL INTELLIGENCE AND JAPAN'S COMPUTER CHALLENGE TO THE WORLD

EDWARD A. FEIGENBAUM
PAMELA MCCORDUCK
History
IBM 360

Figure 4. Card Codes and Graphics for 64-Character Set
Assembly Language

```plaintext
mov ecx, ebx
mov esp, edx
mov edx, r9d
mov rax, rdx
```

Assembler + Linker

```
100101011001
0100111111011
1110101011101
0101010101010
```

Processor
Higher Level Languages
The main advantage we expect the advice taker to have is that its behavior will be improvable merely by making statements to it, telling it about its … environment and what is wanted from it.

- John McCarthy 1958
McCarthy's paper belongs in the Journal of Half-Baked Ideas ...
... The gap between McCarthy's general programme and its execution ... seems to me so enormous that much more has to be done to persuade me that even the first step in bridging the gap has already been taken.

- Yehoshua Bar-Hillel 1958
The potential use of computers by people to accomplish tasks can be “one-dimensionalized” into a spectrum representing the nature of the instruction that must be given the computer to do its job. Call it the **what-to-how spectrum**.

At one extreme of the spectrum, the user supplies his intelligence to instruct the machine with precision exactly how to do his job step-by-step. ... At the other end of the spectrum is the user with his real problem. ... He aspires to communicate what he wants done ... without having to lay out in detail all necessary subgoals for adequate performance.

- Ed Feigenbaum 1974
Bob Kowalski

Programming in Prolog
Using the ISO Standard
Fifth Edition
Springer
This course
Types of Logic Programming:
  Database Programming
  General Logic Programming
  Dynamic Logic Programming
  Constraint Systems
  Answer Set Programming
  Inductive Logic Programming (i.e. Progol)

Languages:
  Datalog
  Prolog
  Epilog
  LPS
  Progol
Schedule

April  2  Introduction
  9  Datalog
 16  Datalog
 23  Prolog
 30  Prolog

May  7  Epilog
 14  Epilog
 21  Constraints
 28  Advanced Logic Programming

June  4  Project Reports
Sets

\[ \{a, b, c\} \cup \{b, c, d\} = \{a, b, c, d\} \]

\[ a \in \{a, b, c\} \]

\[ \{a, b, c\} \subseteq \{a, b, c, d\} \]

Functions and Relations

\[ f(a, b) = c \]

\[ r(a, b, c) \]
CS 106 or equivalent
Numerical Grade
- 15% for each of Assignments 1, 2, 3, 4
- 40% for the Term Project

Letter Grade
- Based on numerical grade (see above)
- *No* curve - independent of number of students
- A, B, C, D distributed ~uniformly over 70% - 100%

Extra Credit
- Supports raising grade a fraction for those near cutoffs
- Discretionary
http://cs151.stanford.edu
Logic Programming

Preface

Sets, Functions, Relations

Unit 1 - Introduction

Lesson 1 - Introduction
Lesson 2 - Datasets

Unit 2 - Basic Logic Programs

Lesson 3 - View Definitions
Lesson 4 - Simple Examples
Lesson 5 - Composite Objects
Lesson 6 - Metaknowledge
Lesson 7 - Implementation
Lesson 8 - Optimization