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 system that manages the creation, modification, and/or execution of logic programs.
Imperative Programming

Inputs → Interpreter → Outputs

```
public class ImperativeExample {
    public static void main(String[] args) {
        // Create a point object and two rectangle objects
        Point origin = new Point(0, 0);
        Rectangle rect1 = new Rectangle(origin, 10, 20);
        System.out.println("Area of rect1: " + rect1.area);
        // Display rect1's width, height, and area
        System.out.println("Width of rect1: " + rect1.width);
        System.out.println("Height of rect1: " + rect1.height);
        // Set rect1's position
        rect1.setOrigin(new Point(5, 5));
        // Display rect1's new position
        System.out.println("New position of rect1: " + rect1.getOrigin());
    }
}
```
A triangle is a polygon with 3 sides.

\[ e = mc^2 \]
Runnable Specifications

A declarative program is basically a **runnable specification**.

**Specification**
- Says what we believe
- Says what we want
- With no extraneous information or arbitrary decisions

**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.
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
Agility
No arbitrary choices

Computer can union multiple programs.

Computer can optimize programs.
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) :-
  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)

row(M,P) :-
  true(cell(M,1,P)) &
  true(cell(M,2,P)) &
  true(cell(M,3,P))
column(N,P) :-
  true(cell(1,N,P)) &
  true(cell(2,N,P)) &
  true(cell(3,N,P))
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))

draw :- ~line(x) &
  ~line(o)
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
Applications
Circuit:

Premises:

\[ o \leftrightarrow (x \land \neg y) \lor (\neg x \land y) \]
\[ a \leftrightarrow z \land o \]
\[ b \leftrightarrow x \land y \]
\[ s \leftrightarrow (o \land \neg z) \lor (\neg o \land z) \]
\[ c \leftrightarrow a \lor b \]

Applications:

Simulation
Configuration
Diagnosis
Test Generation

Conclusion:

\[ x \land y \Rightarrow \neg c \]
### Constraint Satisfaction

#### Owners vs. Broods

<table>
<thead>
<tr>
<th>Year</th>
<th>Anita</th>
<th>Barbara</th>
<th>Douglas</th>
<th>Fernando</th>
<th>Ginger</th>
<th>beagle</th>
<th>bulldog</th>
<th>chow chow</th>
<th>great dane</th>
<th>maltese</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>2007</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>2008</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>2009</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>2010</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
</tbody>
</table>

#### Sudoku

```

<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>1</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
</tbody>
</table>
```

---

15
Deductive Databases

\begin{align*}
q(X) & : \neg 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)
\end{align*}

Questions
Updates

Answers
Notifications

Database Manager

\begin{align*}
p(a,b) \\
p(b,c) \\
p(a,b) \\
p(p(a,b))
\end{align*}
Worksheets

Gates Information Network

<table>
<thead>
<tr>
<th>Home</th>
<th>People</th>
<th>Groups</th>
<th>Classrooms</th>
<th>Events</th>
<th>Series</th>
<th>Schedule</th>
<th>Profile</th>
<th>Dashboard</th>
</tr>
</thead>
</table>

Title | Create a new Event. |
Room | Gates 200 |
Date | 2016-10-06 |
Start Time | 2 |
End Time | 2 |
Duration | |
Owner | Michael Geneseech |
Webpage | |

Comments and complaints to意识形态@stanford.edu.

Gates Info Network

Department of Computer Science
MCS Program Sheet (2010-11)

| Name: Charles Parthenaquet | Email: parthenaquet@stanford.edu |
| Advisor: | Proposed date for degree conferral: 10/6/2010 |
| Student ID: | 3896 | |

General Instructions:
Before the end of your first quarter, you should complete the following steps. Detailed instructions are included in the Guide to the MCS Program Sheet. In your orientation packet, an online version is available at https://www.stanford.edu/degrees/maconferral/.

- Complete this program sheet by filling in the number, name, and units of each course you intend to use for your degree.
- Create a course schedule showing the year and quarter in which you intend to take each course in your program.
- Meet with your advisor and review the necessary signatures on the program sheet.

Foundations Requirement:
You must satisfy the requirements listed in each of the following areas; all courses taken elsewhere must be approved by your advisor on a foundation course waiver form. Required documents for waiving a course include course descriptions, syllabi, and textbook lists. These documents can be organized here: https://www.stanford.edu/degrees/maconferral/.

Note: If you are amending an old program sheet, entering “en” in the approval column for courses that have already been approved.

<table>
<thead>
<tr>
<th>Requirement Title</th>
<th>Equivalent credit (maximum number of units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Units Used to Satisfy Foundations Requirements</td>
<td>10</td>
</tr>
</tbody>
</table>

Equivalent Credit (maximum number of units):

- Total Units: 10
- Status: Draft

Director & Officer Information

<table>
<thead>
<tr>
<th>Personal Information</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>Last Name</td>
<td>Suffix</td>
</tr>
<tr>
<td>Nick Name</td>
<td>Email</td>
<td>Tel.</td>
</tr>
<tr>
<td>Degree</td>
<td>GPA</td>
<td>Graduation Year</td>
</tr>
</tbody>
</table>

Biography:

[Enter your biographical information here]

Actions:

[Save Changes]

© 2010 Stanford University Logic Group

Change Your Scoping

[Diagram showing various categories and options such as Sales, Customer Invoicing, Project Invoicing, etc.]

New | Will appear twice

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.
General Game Playing
Natural Language Processing

Lecture Notes

PROLOG AND NATURAL-LANGUAGE ANALYSIS

Fernando C.N. Pereira
and
Stuart M. Shieber

Center for the Study of Language and Information
Theorem Proving

PTTP

means

Prolog Technology Theorem Prover

by acronymsandslang.com
Japan’s Fifth Generation Project
History
IBM 360
Assembly Language

Assembly Language

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

Assembler + Linker

Machine Language

100101011001
010011111011
111010101101
010101010101

Processor

Programmer
Higher Level Languages

Fortran for Scientists and Engineers
Stephen J. Chapman

COMMON LISP
A Gentle Introduction to Symbolic Computation
David S. Touretzky
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
This course
Types of Logic Programming:

Basic Logic Programming
Dynamic Logic Programming
Constraint Systems
General Logic Programming (i.e. Prolog)
Answer Set Programming
Inductive Logic Programming (i.e.. Progol)

Languages:

Datalog
Epilog
LPS
Prolog
Progol
Schedule

April       2  Introduction
9  Basic Logic Programming
16  Basic Logic Programming
23  Basic Logic Programming
30  Dynamic Logic Programming

May        7  Dynamic Logic Programming
14  Constraint Systems
21  Constraint Systems
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
10% for Assignment 1
20% for each of Assignments 2, and 3, and 4
30% for Assignment 5 (Final 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
Logic Programming vs. Machine Learning
Logic Programming and Machine Learning
Maybe