Logic Programming

Introduction

Michael Genesereth
Computer Science Department
Stanford University

Lecture will begin at ~1:35 PDT.
Programmed Computer System

Inputs → Interpreter / Compiler → Outputs

Data Structures
Specifications versus Programs

Definitions
Assumptions
Goals
Traditional Program

Definitions
Assumptions
Goals

Inputs → Interpreter / Compiler → Outputs

Database

Data Structures
Logic Program

Definitions
Assumptions
Goals

Inputs

Interpreter / Compiler

Database

Outputs
A logic program is effectively a **runnable specification.**
Logic as a Specification Language

Language of Logic

Domain Independent
+
Highly expressive

Logic Interpreters / Compilers
Automated Reasoners capable of drawing conclusions
Can take advantage of domain-dependent reasoners
but are also capable domain-independent reasoning
Types of Logic Programming

Database Programming (Datalog, SQL)
Classical Logic Programming (Prolog)
Dynamic Logic Programming (Epilog, LPS)

Constraint Satisfaction
Program Synthesis

Answer Set Programming (ASP)
Probabilistic Logic
Inductive Logic Programming (Progol)
Why Logic Programming
Benefits

- Efficiency
- Lots of traditional programmers
- Well established software engineering practices

Disadvantages

- Creation, maintenance expensive and time-consuming
- Different programs for different tasks
- Difficult to explain results
- Programs not comprehensible to ordinary users
Ease of Creation

Logic Programs are relatively easy to create.

Requires **little work**. The specification is the program; no need to make choices about data structures and algorithms.

Specification authors can get by with **few assumptions** about the capabilities of systems executing those programs.

**Easier to learn** logic programming than traditional programming. Think spreadsheets.

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

*Easy to deal with changing circumstances*
Easy to use for multiple tasks

Sample Program
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 grandparent of Cal.
Determine all of the grandchildren of Art.
Compute the grandparents of Cal.
Compute all grandparent-grandchildren pairs.
McCarthyy’s Example
McCarthy’s Example
McCarthy’s Example
The building is illegal

The shadow line is 262 cm

The allowable shadow is 240 cm

262 > 240

The parcel is in zone R-1

The allowable shadow in R-1 is 240 cm

The building is 462 cm high

The building is 200 cm from the boundary

462 - 200 = 262
Why was my building plan rejected?

*Your plan is illegal because your shadow line (262 cm) exceeds the allowable shadow (240 cm).*

What is my shadow line?

*Your shadow line (262 cm) is the maximum intrusion into the yard of a side neighbor determined by a 45 degree line from the highest point of the building.*

What is the allowable shadow line?

*Your parcel is in zone R-1 and in zone R-1, the maximum shadow that can be cast on a side neighbor is 240 cm.*
Successes
Circuit: Description:

Applications:
Simulation
Configuration
Diagnosis
Test Generation

Description:

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

\[ q(X) :\neg p(X,Y) \land p(X,Z) \land Y \neq Z \]

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

\[ \text{illegal} :\neg p(X,Y) \land p(Y,X) \]

Questions

Updates

Database Manager

Answers

Notifications
Business Rules and Workflow

ERp SYSTEM

Financial Management
Customer Relationship Management
Supply Chain Management
Human Resource Management
Manufacturing Resource Planning
Computational Law is that branch of legal informatics concerned with the mechanization of legal reasoning.

Automated Compliance Management
- Legal analysis of specific cases
- Planning for compliance in specific cases
- Analysis of regulations for overlap, consistency, etc.

Portico
General Game Playing
General Game Playing

Pelican Hunters
Non-Successes
PTTP

means

Prolog Technology Theorem Prover

by acronymsandslang.com
Japan’s Fifth Generation Project
History
LGP-30 (1GL)
IBM 360

Figure 4. Card Codes and Graphics for 84-Character Set
Assembly Language (2GL)

Assembly Language

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

Assembler + Linker

Machine Language

100101011001
010011111011
111010101101
010101010101

Programmer

Processor
Higher Level Languages (3GL)
Symbolic Processing Languages (3GL)
Imperative Programming Languages
Declarative Programming Languages

Declarative vs. Imperative
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
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
If code is the problem, the only possible answer is to eliminate the coding by building systems directly from their specifications.

- Val Huber, 1997
This course
<table>
<thead>
<tr>
<th>April</th>
<th>2</th>
<th>Introduction</th>
<th>May</th>
<th>7</th>
<th>Operation Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>Datasets</td>
<td></td>
<td>9</td>
<td>Model Management</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Queries</td>
<td></td>
<td>14</td>
<td>Reactive Worksheets</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Query Examples</td>
<td></td>
<td>16</td>
<td>Semantic Worksheets</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>Query Evaluation</td>
<td></td>
<td>21</td>
<td>Constraint Satisfcation</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>Query Optimization</td>
<td></td>
<td>23</td>
<td>Program Synthesis</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>View Definitions</td>
<td></td>
<td>28</td>
<td>Extensions and Advances</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>View Evaluation</td>
<td></td>
<td>30</td>
<td>Past Projects</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>Simple Examples</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>May</td>
<td>2</td>
<td>Lists, Sets, Trees</td>
<td>Jun</td>
<td>4</td>
<td>Project Reports</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>Project Reports</td>
</tr>
</tbody>
</table>
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

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

\[r(a, b, c)\]
Grades

Numerical Score
  15% for each of Assignments 1, 2, 3, 4
  40% for the Term Project

Reported Grade
  Based on numerical score (see above)
  *No* curve - independent of number of students
  Satisfactory = 70% and above

Extra Credit
  Added to score before determining Reported Grade
  Discretionary
"This is a book for the 21st century: presenting an elegant and innovative perspective on logic programming. Unlike other texts, it takes datasets as a fundamental notion, thereby bridging the gap between programming languages and knowledge representation languages; and it treats updates on an equal footing with datasets, leading to a sound and practical treatment of action and change." – Bob Kowalski, Professor Emeritus, Imperial College London

"In a world where Deep Learning and Python are the talk of the day, this book is a remarkable development. It introduces the reader to the fundamentals of traditional Logic Programming and makes clear the benefits of using the technology to create runnable specifications for complex systems." – Son Cao Tran, Professor in Computer Science, New Mexico State University

"Excellent introduction to the fundamentals of Logic Programming. The book is well-written and well-structured. Concepts are explained clearly and the gradually increasing complexity of exercises makes it so that one can understand easy notions quickly before moving on to more difficult ideas." – George Younger, student, Stanford University
http://cs151.stanford.edu