

# Logic Programming

## *Introduction*

Michael Genesereth  
Computer Science Department  
Stanford University

**Lecture will begin at ~1:35 PDT.**

# Logic Programming (Spoiler Alert)

**Logic Programming** is a style of programming based on Symbolic Logic.

**Logic Program** is a collection of sentences encoded in the language of Symbolic Logic.

**Logic Programming Language** is a specific language for writing such programs.

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

# Imperative Programming

```
Editeur - [Java syntax test editor.jav]
File Edit Search Macro Tools Window Help
public class CreateObjectDemo {
    public static void main(String[] args) {
        // create a point object and two rectangle objects
        Point origin_one = new Point(23, 94);
        Rectangle rect_one = new Rectangle(origin_one, 100, 200);
        Rectangle rect_two = new Rectangle(50, 100);

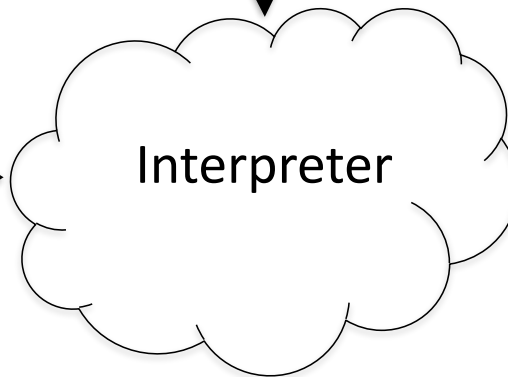
        // display rect_one's width, height, and area
        System.out.println("Width of rect_one: " + rect_one.width);
        System.out.println("Height of rect_one: " + rect_one.height);
        System.out.println("Area of rect_one: " + rect_one.area());

        // set rect_two's position
        rect_two.origin = origin_one;

        // display rect_two's position
        System.out.println("X Position of rect_two: " + rect_two.origin.x);
        System.out.println("Y Position of rect_two: " + rect_two.origin.y);

        // move rect_two and display its new position
        rect_two.move(40, 72);
    }
}
```

Inputs

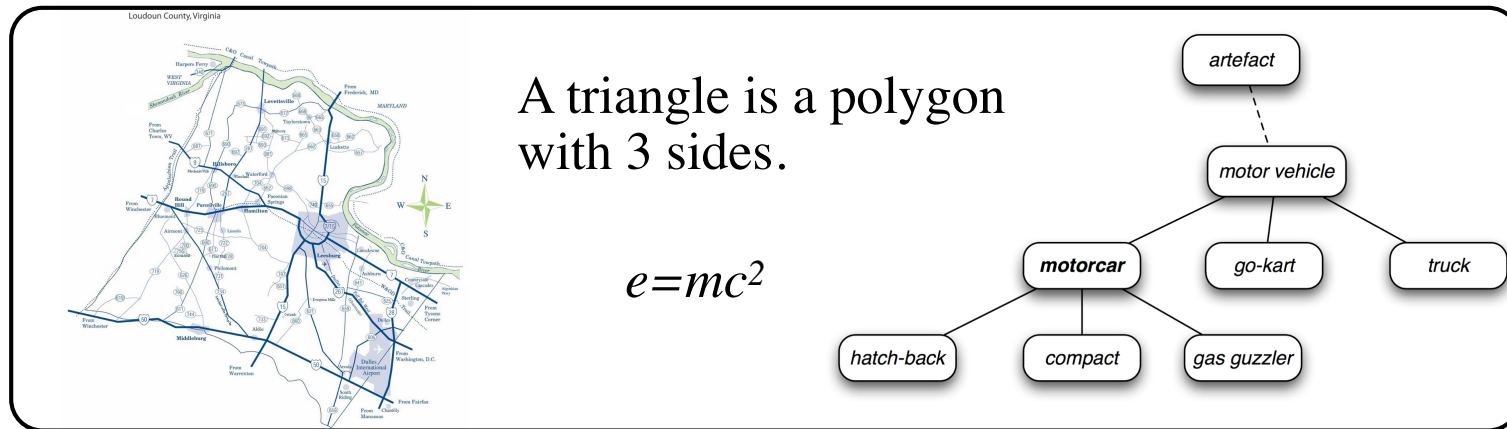


Interpreter

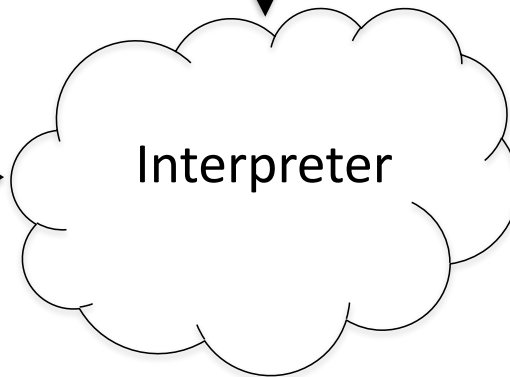


Outputs

# Declarative Programming



Inputs



Outputs

# Runnable Specifications

## **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**

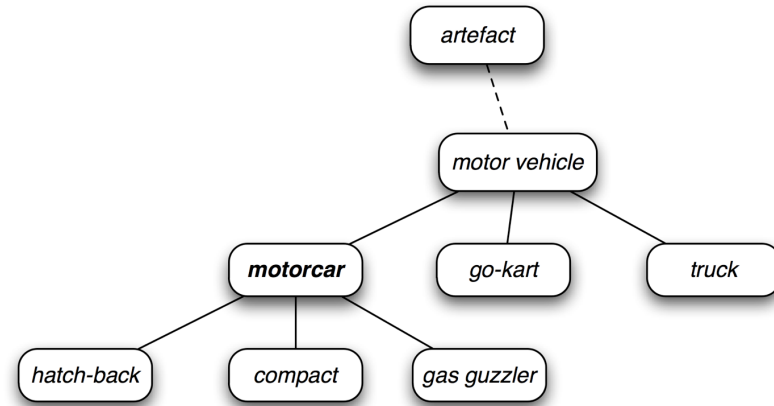
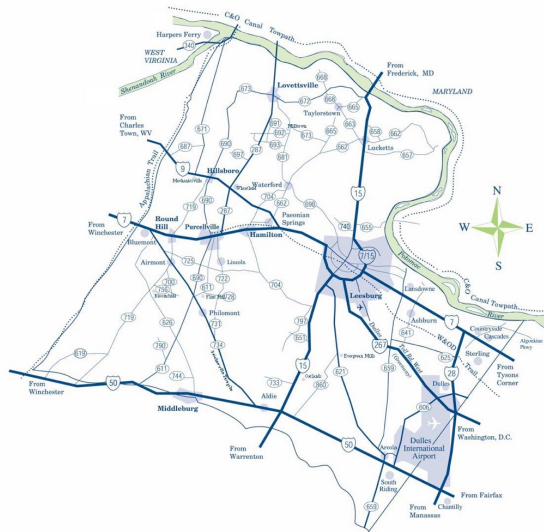
Can be **compiled** into traditional programs

# Runnable Specifications

A logic program  
is a  
**runnable specification.**

# Specialized Languages

Loudoun County, Virginia



Column Name	Condensed Type	Nullable
CITY_ID	int	NOT NULL
CITY_NAME	char(30)	NULL
COUNTRY	char(30)	NULL

Column Name	Condensed Type	Nullable
FLIGHT_ID	int	NOT NULL
RESERVATION_ID	int	NOT NULL
FLIGHT_DATE	datetime	NULL

Column Name	Condensed Type	Nullable
RESERVATION_ID	int	NOT NULL
NUM_OF_SEATS	int	NULL
PLURCHASED_FLAG	bit	NULL
CREDIT_CARD_NUM	char(30)	NULL
CREDIT_CARD_TYPE	int	NULL
CREDIT_CARD_MONTH	char(10)	NULL
CREDIT_CARD_YEAR	char(10)	NULL
RESERVATION_DATE	datetime	NULL
FIRST_NAME	char(30)	NULL
EMAIL	char(30)	NULL
RESERVATION_NOTIFIED	bit	NULL
PLURCHASE_NOTIFIED	bit	NULL
TRANSACTION_ID	bigint	NULL
USER_ID	int	NULL

Column Name	Condensed Type	Nullable
FLIGHT_ID	int	NOT NULL
FLIGHT_CODE_CHAR	char(10)	NULL
FROM_CITY_ID	int	NULL
TO_CITY_ID	int	NULL
NUM_OF_SEATS	int	NULL
DEP_TIME	datetime	NULL
ARR_TIME	datetime	NULL
FREQUENCY	char(20)	NULL
TICKET_PRICE	int	NULL

Column Name	Condensed Type	Nullable
CREDIT_CARD_NUM	char(30)	NULL
TRANSACTION_ID	varchar(50)	NULL
AMOUNT	float(53)	NULL

Column Name	Condensed Type	Nullable
CREDIT_CARD_NUM	char(10)	NOT NULL
CREDIT_CARD_TYPE	int	NOT NULL
CREDIT_CARD_MONTH	int	NOT NULL
CREDIT_CARD_YEAR	int	NOT NULL
CREDIT_CARD_BAL	int	NULL

Column Name	Condensed Type	Nullable
PASSENGER_ID	int	NOT NULL
RESERVATION_ID	int	NOT NULL
FIRST_NAME	char(20)	NULL
LAST_NAME	char(20)	NULL

Column Name	Condensed Type	Nullable
USER_ID	int	NOT NULL
USER_NAME	char(25)	NOT NULL
TRANSACTION_ID	bigint	NULL
TRANSACTION_TYPE	char(10)	NULL

Column Name	Condensed Type	Nullable
INVOICE_ID	int	NOT NULL
RESERVATION_ID	int	NULL
AMOUNT	int	NULL
DATE_PAID	datetime	NULL
PAYMENT_TYPE	char(20)	NULL
CREDIT_CARD_NUM	char(30)	NULL
CREDIT_CARD_TYPE	char(20)	NULL
CREDIT_CARD_MONTH	char(10)	NULL
CREDIT_CARD_YEAR	char(10)	NULL
TRANSACTION_ID	bigint	NULL
USER_ID	int	NULL
ccAuthorized	bit	NULL

A triangle is a polygon with 3 sides.

# Logic as a Specification Language

## Language

Declarative language

+

Highly expressive

Other declarative languages exist but statements in most of those languages can be translated to logical form.

## Interpreter

Automated Reasoners capable of drawing conclusions

Can take advantage of domain-dependent reasoners

but are also capable domain-independent reasoning



Benefits

# Programming Ease

*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.

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

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

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

# Agility

*Ability to respond to changing circumstances or goals*



# Versatility

*Ability to be used for multiple purposes*

## **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.

# McCarthy's Example of Versatility



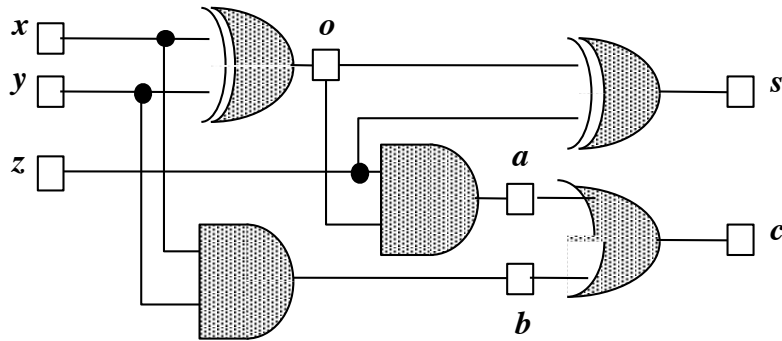
# McCarthy's Example of Versatility



Successes

# Engineering

Circuit:



Premises:

$$o \Leftrightarrow (x \wedge \neg y) \vee (\neg x \wedge y)$$

$$a \Leftrightarrow z \wedge o$$

$$b \Leftrightarrow x \wedge y$$

$$s \Leftrightarrow (o \wedge \neg z) \vee (\neg o \wedge z)$$

$$c \Leftrightarrow a \vee b$$

Applications:

Simulation

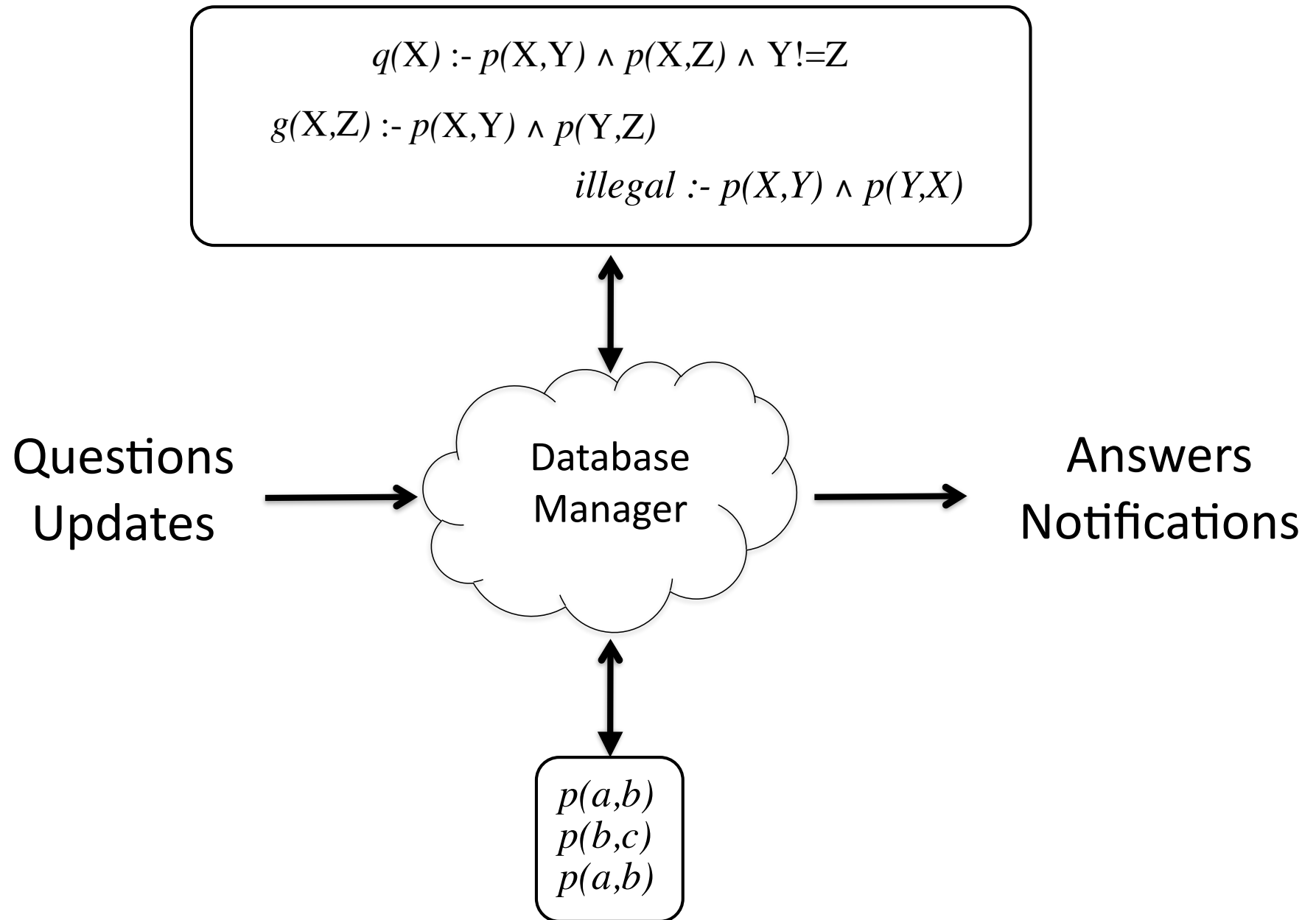
Configuration

Diagnosis

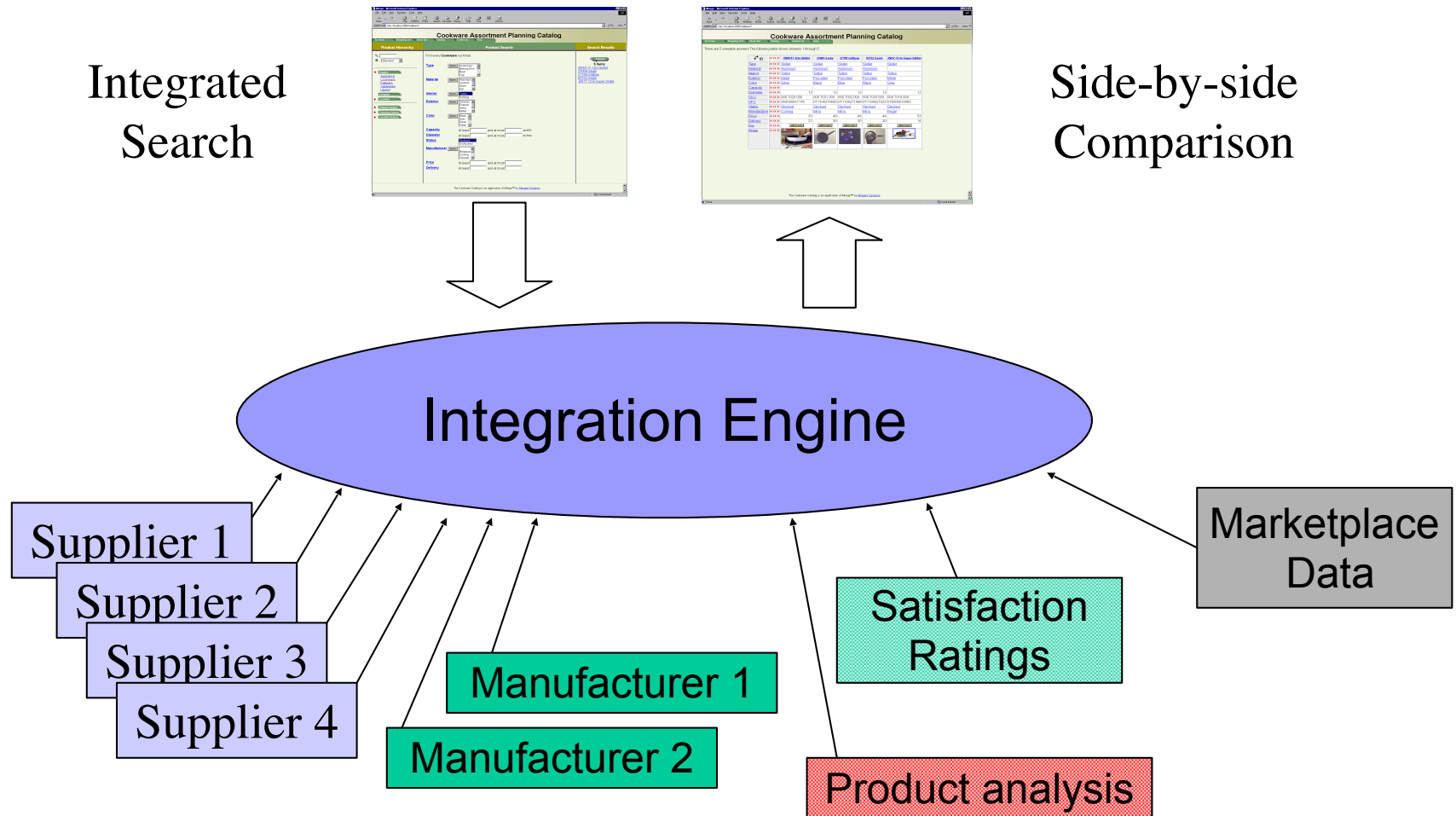
Test Generation



# Deductive Databases



# Data Integration

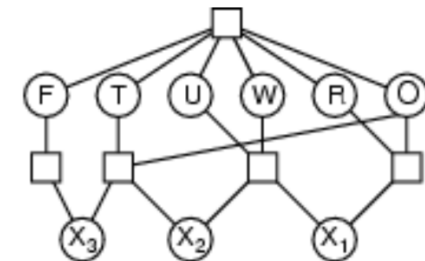


# Constraint Satisfaction

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

## Example: Cryptarithmic

$$\begin{array}{r} \text{TWO} \\ + \text{TWO} \\ \hline \text{FOUR} \end{array}$$



- **Variables:**  $FTUWRO$
- **Domains:**  $\{0,1,2,3,4,5,6,7,8,9\}$
- **Constraints:** *Alldiff* ( $F,T,U,W,R,O$ )
  - $O + O = R + 10 \cdot X_1$
  - $X_1 + W + W = U + 10 \cdot X_2$
  - $X_2 + T + T = O + 10 \cdot X_3$
  - $X_3 = F, T \neq 0, F \neq 0$

$X_1, X_2, X_3$   
 $\{0,1\}$

# Worksheets

## Gates Information Network

[Home](#) [People](#) [Groups](#) [Classrooms](#) [Events](#) [Series](#) [Schedule](#) [Profile](#) [Dashboard](#)

Create a new Event.

**Title**

**Room**

**Date**

**Start Time**

**End Time**

**Duration**

**Owner** [Michael Genesereth](#)

**Webpage**

Comments and complaints to [action@logic.stanford.edu](mailto:action@logic.stanford.edu).

### SOPRI

#### Director & Officer Information

[Personal Information](#) | [Address](#) | [Degrees](#) | [Positions](#) | [Offices & Directorships](#) | [Affiliations](#) | [Family](#) | [Other Personal Info](#) | [Legal Proceedings](#) | [Compensation](#) | [Stock](#) | [Control](#) | [Business Affiliations](#) | [NYSE](#) | [Other](#) | [Audit Committee](#) | [Books & Records](#)

**Personal Information**

<b>First Name</b> <input type="text" value="e.g. William"/>	<b>Middle Name</b> <input type="text" value="e.g. Henry"/>	<b>Last Name</b> <input type="text" value="e.g. Gates"/>	<b>Suffix</b> <input type="text"/>
<b>Nick Name</b> <input type="text" value="e.g. Bill"/>			
<b>Date of Birth</b> Select Month: <input type="text"/> Select Day: <input type="text"/> Select Year: <input type="text"/>			
<b>Gender</b> Select Gender: <input type="text"/>			
<b>Biography</b> Enter your biography here. <input type="text"/>			

**Actions**

© 2010 Stanford University Logic Group

### DEPARTMENT OF COMPUTER SCIENCE

#### MSCS Program Sheet (2010-11)

Artificial Intelligence Primary Specialization

Name: [Charles Parnell Naut](#) Advisor:  Proposed date for degree conferral:  Date: 10/8/2010  
Student ID #:  Email: [cnaut@stanford.edu](mailto:cnaut@stanford.edu)  HCP?  Cotermin?

**GENERAL INSTRUCTIONS**

Before the end of your first quarter, you should complete the following steps. Detailed instructions are included in the [Guide to the MSCS Program Sheet](#) in your orientation packet (an online version is available at [cs.stanford.edu/degrees/mscs/programsheets/](http://cs.stanford.edu/degrees/mscs/programsheets/)):

- 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 sheet.
- Meet with your advisor and secure 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: [cs.stanford.edu/degrees/mscs/waivers/](http://cs.stanford.edu/degrees/mscs/waivers/). Do not enter anything in the "Units" column for courses taken elsewhere.

Note: If you are amending an old program sheet, enter "on file" in the approval column for courses that have already been approved.

Required:	Equivalent elsewhere (course number/title/institution)	Approval	Grade	Units
Logic, Automata and Complexity (✓CS 103)	<input type="text"/>	<input type="text"/>	<input type="text"/>	4
Probability (✓CS 109, ✓STATS 116, ✓CME 106, or ✓MS&E 220)	<input type="text"/>	<input type="text"/>	<input type="text"/>	
Algorithmic Analysis (✓CS 161)	<input type="text"/>	<input type="text"/>	<input type="text"/>	5
Computer Organization and Systems (✓CS 107)	<input type="text"/>	<input type="text"/>	<input type="text"/>	5
Principles of Computer Systems (✓CS 110)	<input type="text"/>	<input type="text"/>	<input type="text"/>	5

TOTAL UNITS USED TO SATISFY FOUNDATIONS REQUIREMENT:

Note: This total may not exceed 10 units.

7 Requirements Left Total Units: 10 Status: Draft

### Change Your Scoping

Country and Type of Business Implementation Focus Organizational Mapping **Scoping** Questions Results Confirmation

Previous Next Finish Cancel Save Search:  Go You Can Also Support

Show All Elements Expand All Your Location: Sales > Customer Invoicing

- Marketing
  - Market Development
  - Campaign Management
- Sales
  - Account and Activity Manage...
  - Product and Service Portfolio ...
  - New Business
  - Selling Products and Services
- Customer Invoicing**
  - Sales Planning
  - Service
    - Product and Service Portfolio ...
    - Entitlement Management
    - Customer Care
    - Field Service and Repair
  - Sourcing
  - Purchasing
  - Product Development
  - Supply Chain Setup Manage...
  - Supply Chain Planning and Cont...
  - Manufacturing, Warehousing, a...
  - Project Management

**Customer Invoicing**

<input type="checkbox"/> Sales and Service Invoicing	<input type="checkbox"/> External Invoicing	<input type="checkbox"/> Project Invoicing	<input checked="" type="checkbox"/> Miscellaneous Invoicing
<input checked="" type="checkbox"/> Communication for Customer Invoicing	<input checked="" type="checkbox"/> Analysis for Customer Invoicing		

Overview Relevance Constraints Notes

When you select certain combinations of elements, the system automatically selects additional elements. The elements that triggered the selection are listed below. You can click on these elements, selected by user or constraint, to navigate to them.

Miscellaneous Invoicing has been set to in scope

"Miscellaneous Invoicing" within "Customer Invoicing"

Selected by constraint

"External Invoicing" <b>New</b>	within "Customer Invoicing"
De-selected by user	
"Project Invoicing" <b>Will appear twice</b>	within "Customer Invoicing"

<http://logicprogramming.stanford.edu/examples/programsheets/demonstration.html>

# Business Rules



# Computational Law

**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

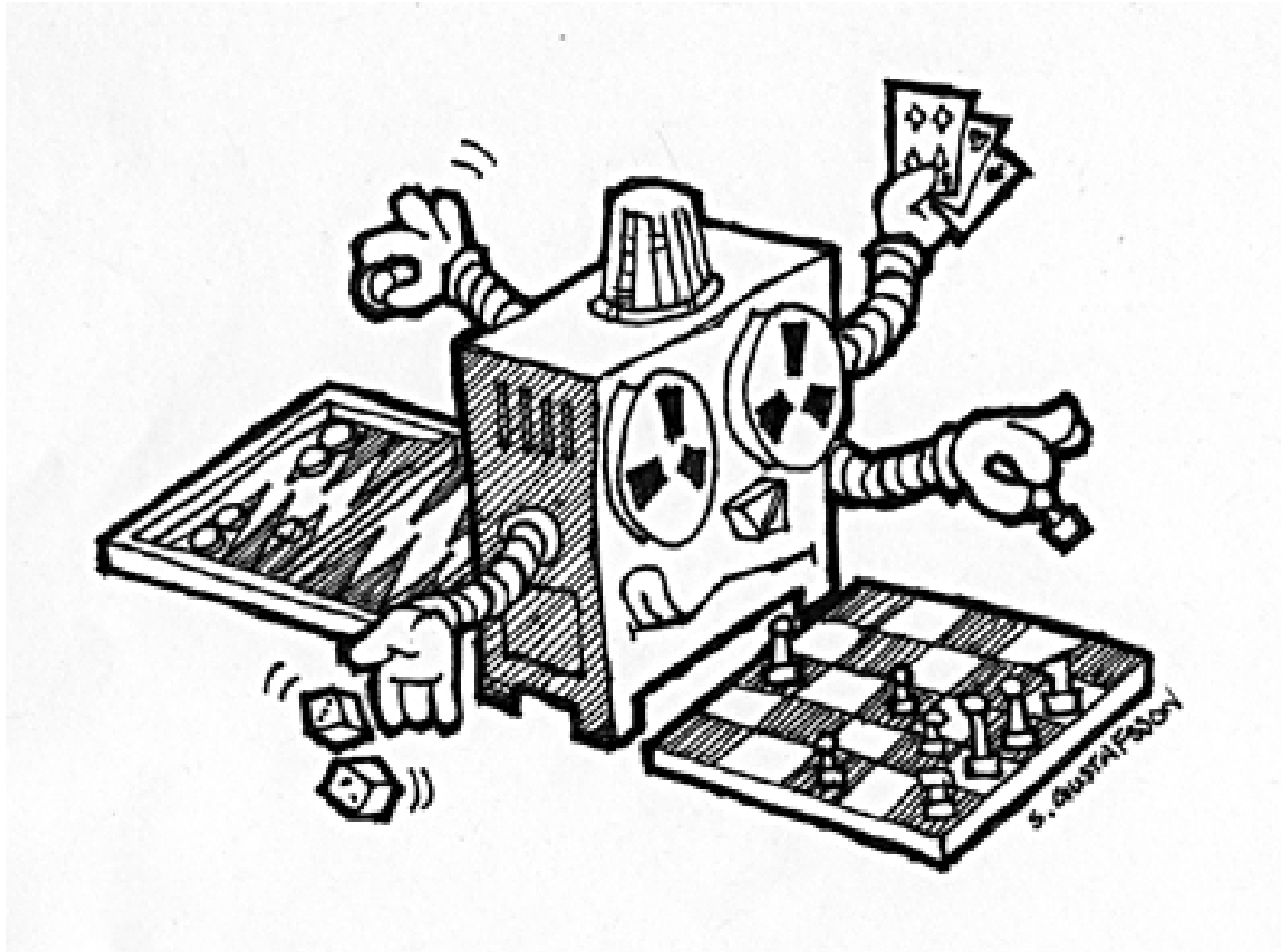
Analysis of regulations for overlap, consistency, etc.

<http://logicprogramming.stanford.edu/examples/portico/demonstration.html>

# General Game Playing



# General Game Playing

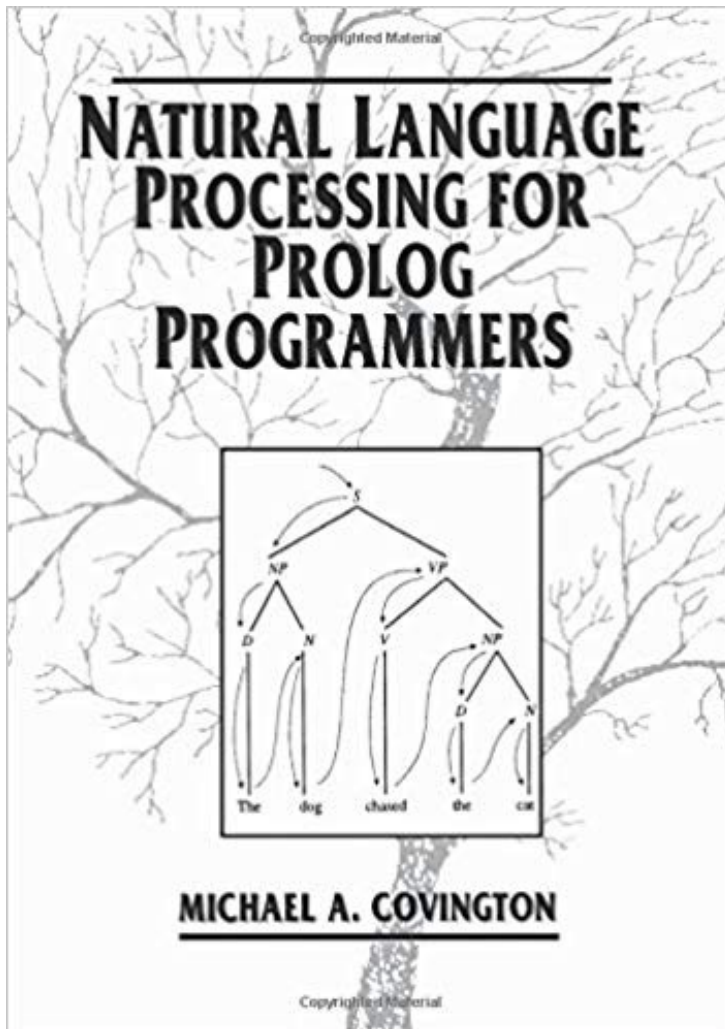


<http://logicprogramming.stanford.edu/examples/nineboard/demonstration.html>



# Non-Successes

# Natural Language Processing



## Lecture Notes

### PROLOG AND NATURAL-LANGUAGE ANALYSIS

Fernando C.N. Pereira  
and  
Stuart M. Shieber

**CSLI** CENTER FOR THE STUDY  
OF LANGUAGE  
AND INFORMATION

# Theorem Proving



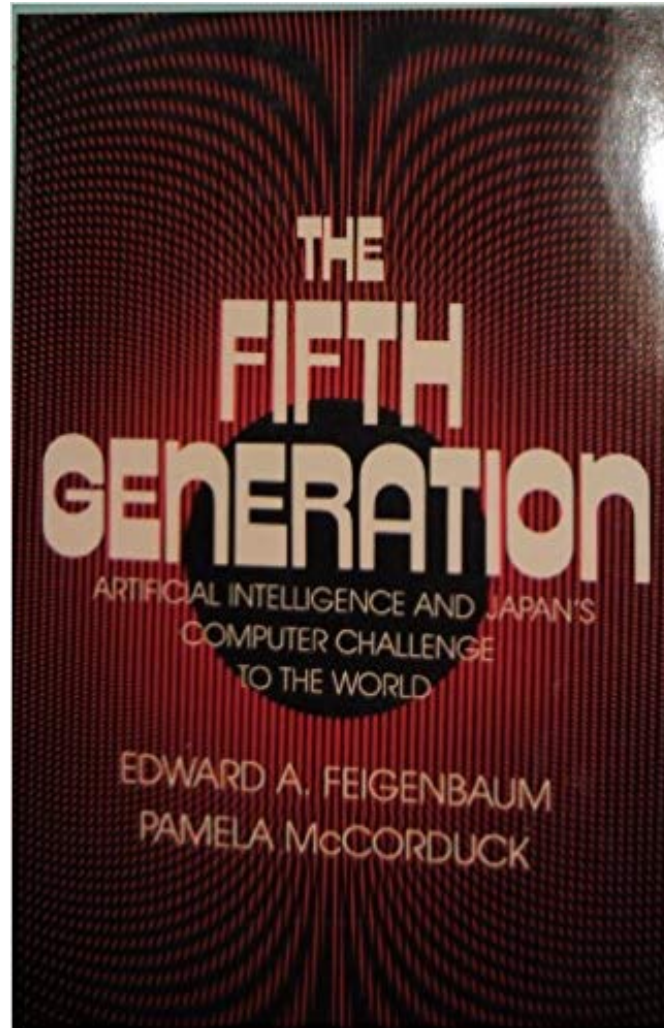
**PTTP**

means

Prolog Technology Theorem  
Prover

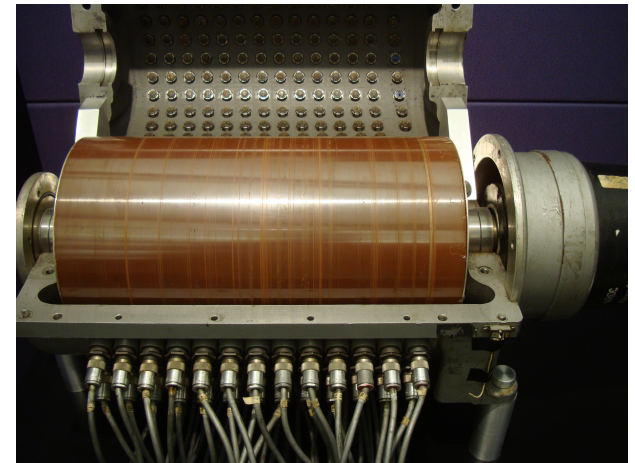
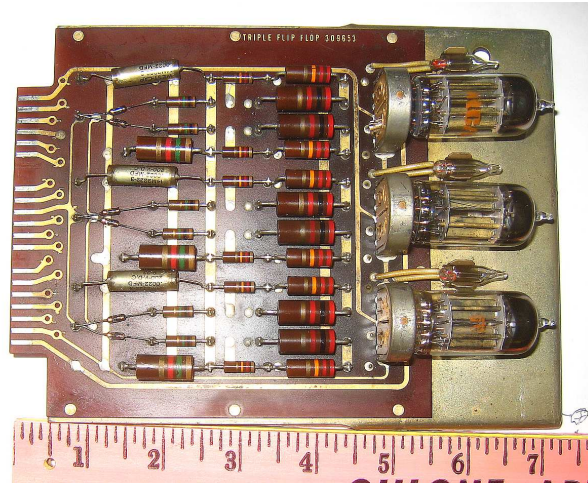
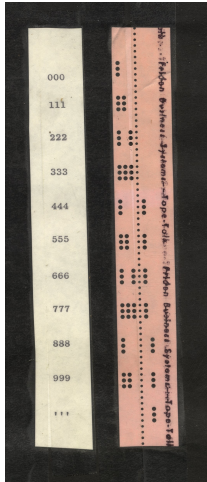
by [acronymsandslang.com](https://www.acronymsandslang.com)

# Japan's Fifth Generation Project



# History

# LGP-30





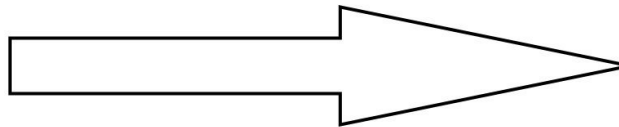
# Assembly Language

Assembly Language

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

Programmer

Assembler + Linker



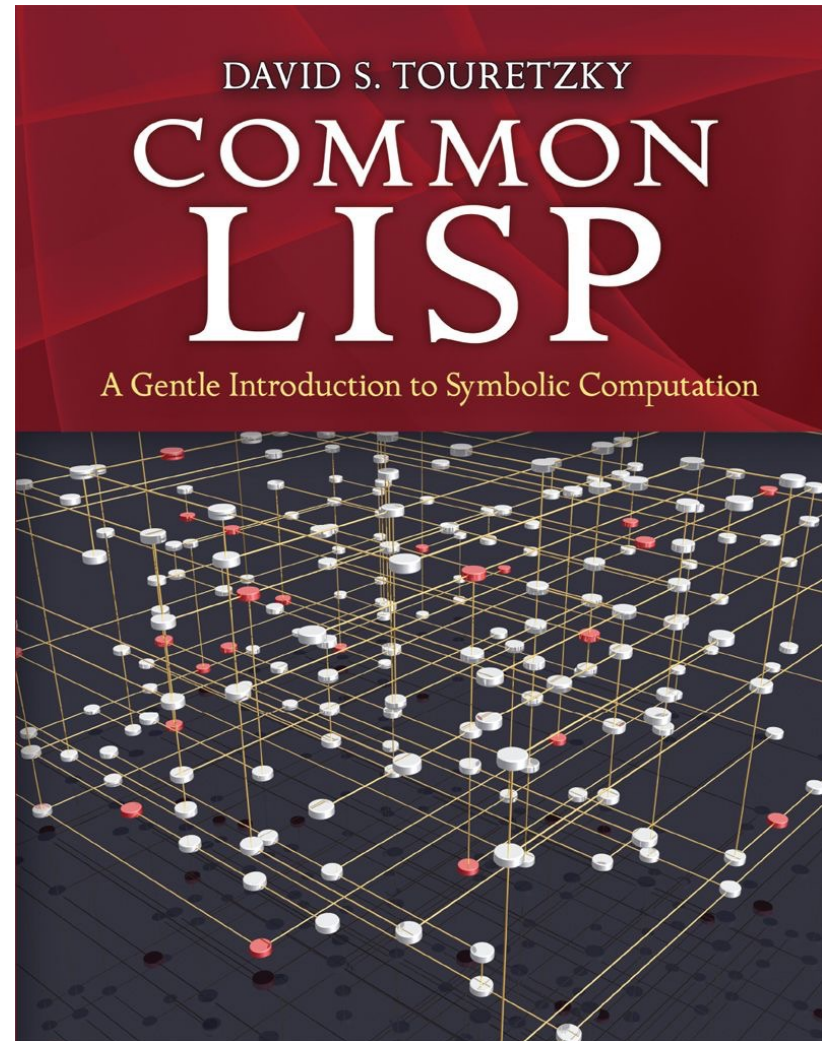
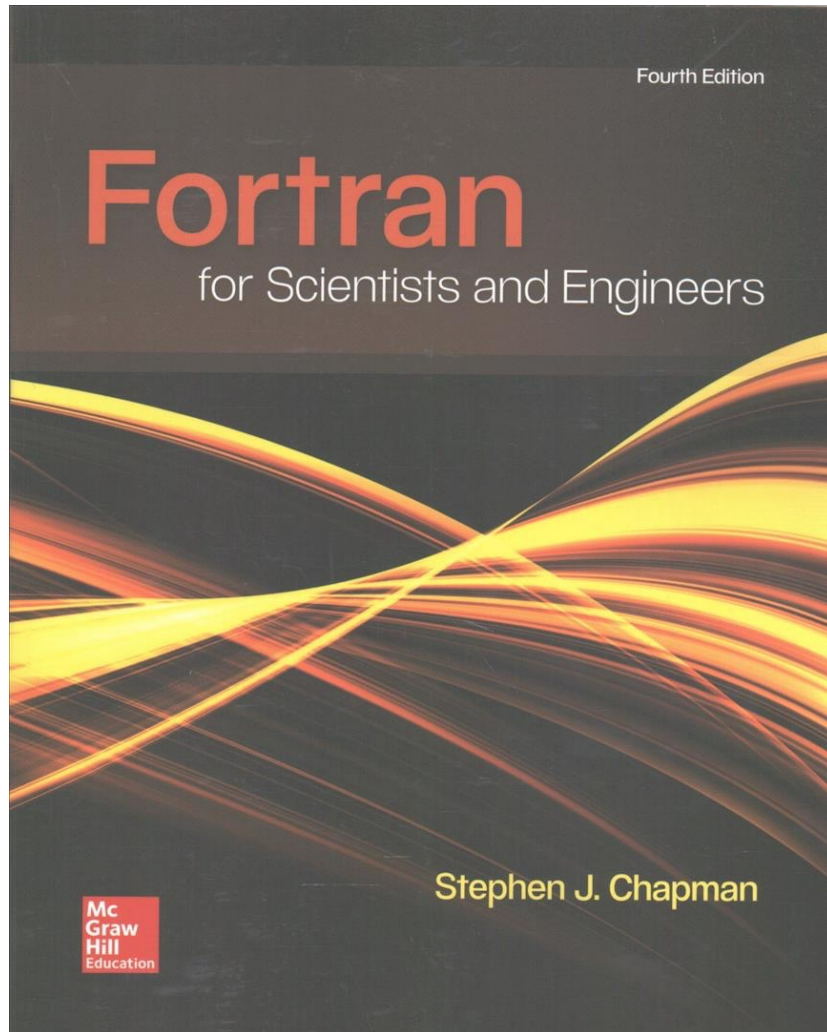
Machine Language

```
100101011001  
010011111011  
111010101101  
01010101010
```

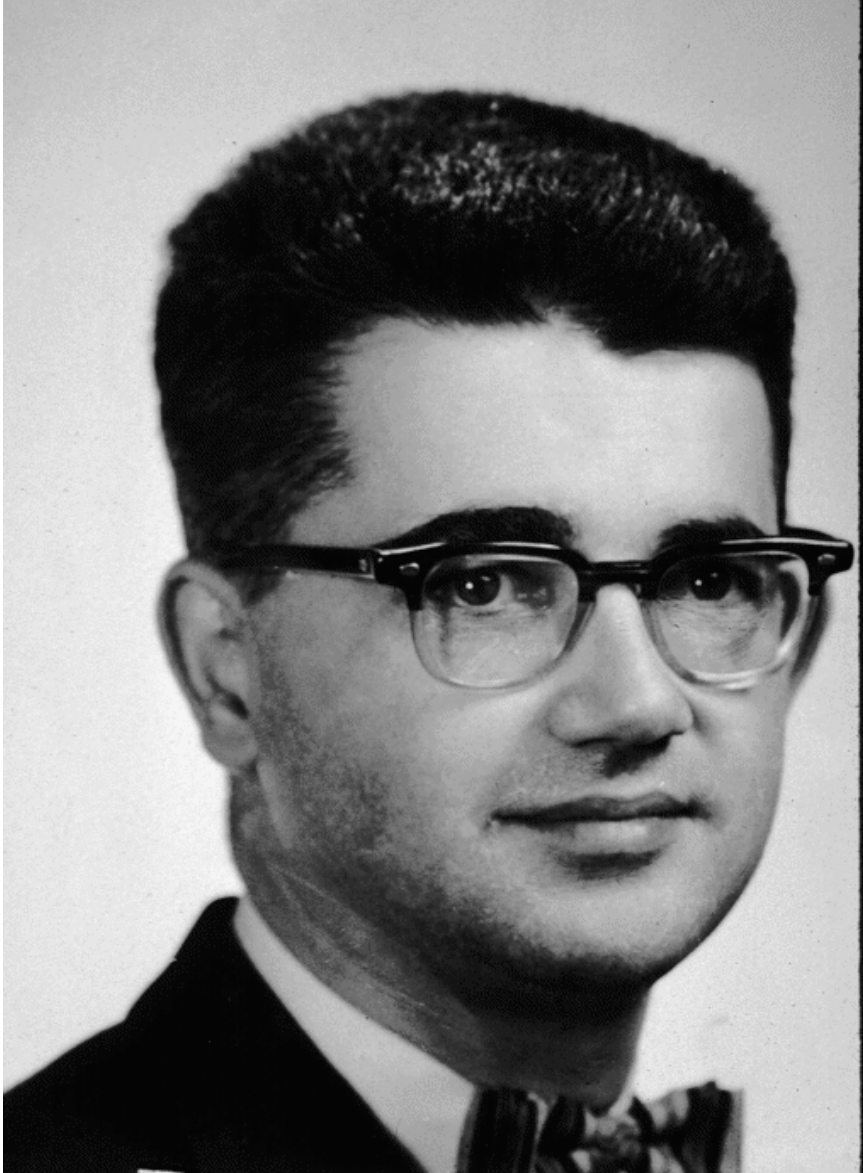
Processor



# Higher Level Languages



# John McCarthy



*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

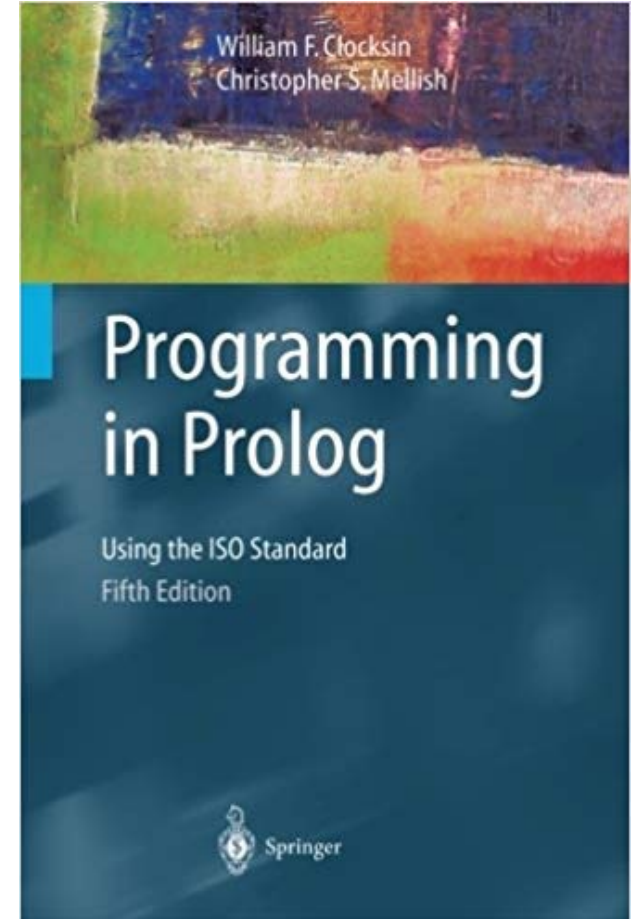
# Ed Feigenbaum



*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*

# Alain Colmerauer and Bob Kowalski



This course

# Types of Logic Programming

Types of Logic Programming:

Database Programming

Classical Logic Programming

Dynamic Logic Programming

Constraint Systems

Answer Set Programming

Inductive Logic Programming (i.e. Progol)

Languages:

Datalog

Prolog

Epilog

Golog

Progol

# Schedule

Mar 29	Introduction	May 3	Action Definitions
31	Datasets	5	Dynamic Systems
		10	Database Management
Apr 5	Queries	12	Worksheets
7	Examples		
12	Query Evaluation	17	Applications
14	Query Optimization	19	Applications
		24	Project Reports
19	View Definitions	26	Project Reports
21	Query Optimization	31	Project Reports
26	Simple Examples		
28	Lists, Sets, Trees		

# Mathematical Background

## 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

$$f(a, b) = c$$

$$r(a, b, c)$$



# Programming Background

CS 106 or equivalent

# Teams

## Composition

3 people each (2 or 4 okay with *good* reason)

## Names:

Pansy Division

The Pumamen

Team Camembert

Mighty Bourgeoisie

Greedy Bastards

Red Hot Chili Peppers

/\*v\*\

X Æ A-12

Michael Genesereth

# Grades

## **Numerical Score**

10% for each of Assignments 1, 2, 3, 4,5

50% 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

# Textbook

Series ISSN: 1939-4608

## *SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING*

**Series Editors:** Ronald J. Brachman, *Jacobs Technion-Cornell Institute at Cornell Tech*  
Francesca Rossi, *AI Ethics Global Leader, IBM Research AI*  
Peter Stone, *University of Texas at Austin*

### Introduction to Logic Programming

Michael Genesereth, *Stanford University*  
Vinay K. Chaudhri, *Stanford University*

“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*

#### ABOUT SYNTHESIS

This volume is a printed version of a work that appears in the *Synthesis Digital Library of Engineering and Computer Science*. Synthesis books provide concise, original presentations of important research and development topics, published quickly, in digital and print formats.



MORGAN & CLAYPOOL PUBLISHERS  
store.morganclaypool.com



GENESERETH • CHAUDHRI

INTRODUCTION TO LOGIC PROGRAMMING

MORGAN & CLAYPOOL



MORGAN & CLAYPOOL PUBLISHERS

# Introduction to Logic Programming

Michael Genesereth  
Vinay K. Chaudhri

*SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING*

Ronald J. Brachman, Francesca Rossi, and Peter Stone, *Series Editors*

<http://cs151.stanford.edu>



# Introduction to Logic Programming

*What  
versus  
How*

[Lessons](#)

[Sierra](#)

[Tools](#)

[Examples](#)

[Zoom](#)

[Piazza](#)

The following syllabus lists all of the materials of the course. Note that there are interactive exercises at the ends of the chapters in the course textbook. (Click on the exercise numbers to go to the exercise pages.) These exercises are an essential part of the course, and you will benefit from tackling them. Some are easier than others, but you should attempt them all. Do the exercises! Do The Exercises!! DO THE EXERCISES!!!

## Color Code

Black - Lecture Slides

[Blue - Readings](#)

[Red - Assignments](#)

Grey - Comment

## Introduction (Week 1)

[Lecture 1 - Introduction](#)

[Lecture 2 - Datasets](#)

[Chapter 1 - Introduction](#)

[Chapter 2 - Datasets](#)

[Programs With Common Sense](#)

[Logic Programming](#)

[Assignment 1.1 - Datasets in Sierra](#)

[Assignment 1.2 - Game State](#)

[Assignment 1.3 - Triples](#)

[Project](#)

