

Logic Programming

Introduction

Michael Genesereth
Computer Science Department
Stanford University

Lecture will begin at ~1:35 PDT.

Programmed Computer System

```
Editeur - [java syntax test editor.java]
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(25, 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



Data Structures

Specifications versus Programs

Definitions
Assumptions
Goals



```
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);  
    }  
}
```



Traditional Program

Definitions
Assumptions
Goals

```
public class CreateObjectDemo {  
    public static void main(String[] args) {  
        // create a point object and two rectangle objects  
        Point origin_one = new Point(25, 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);  
    }  
}
```

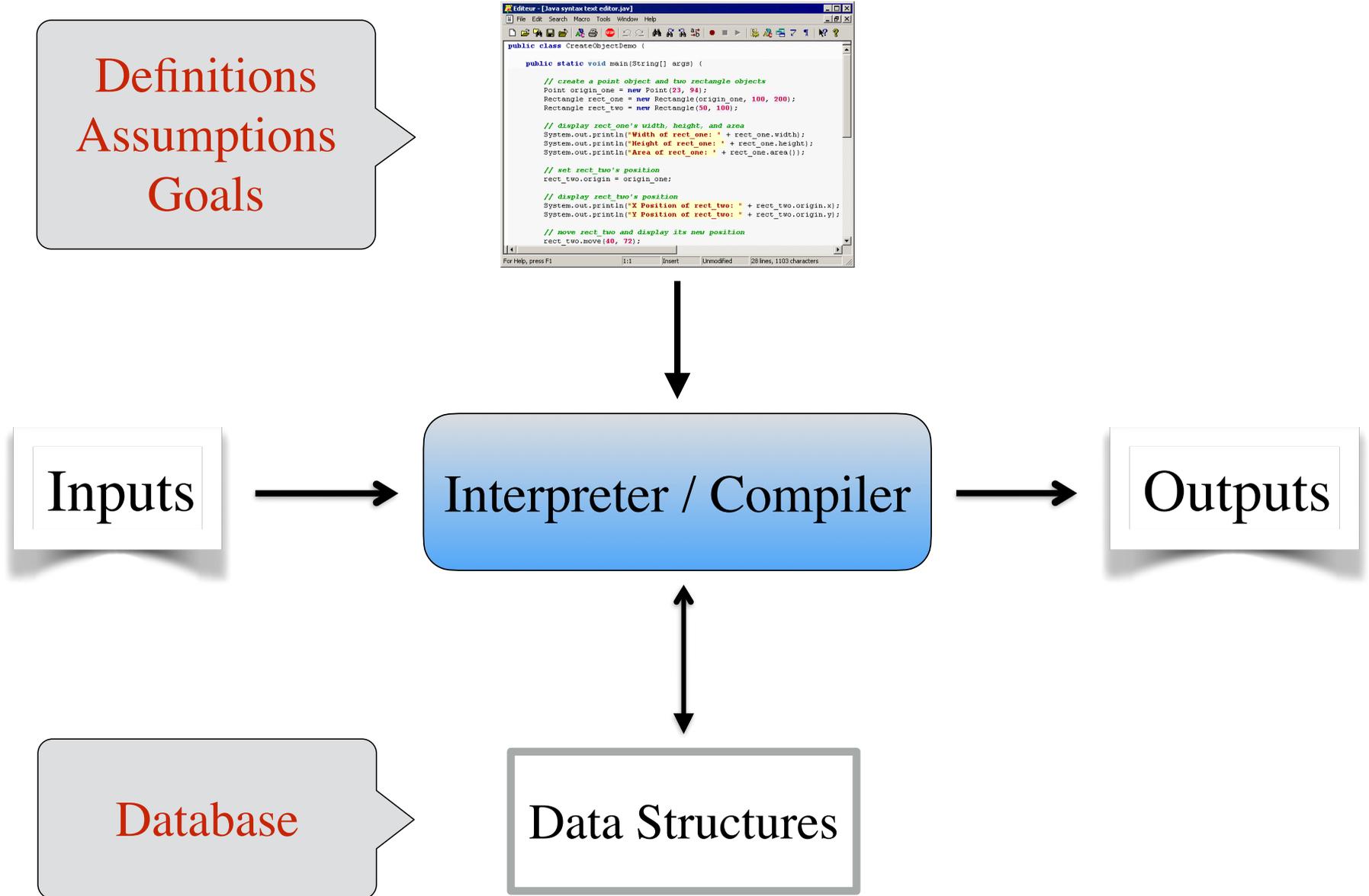
Inputs

Interpreter / Compiler

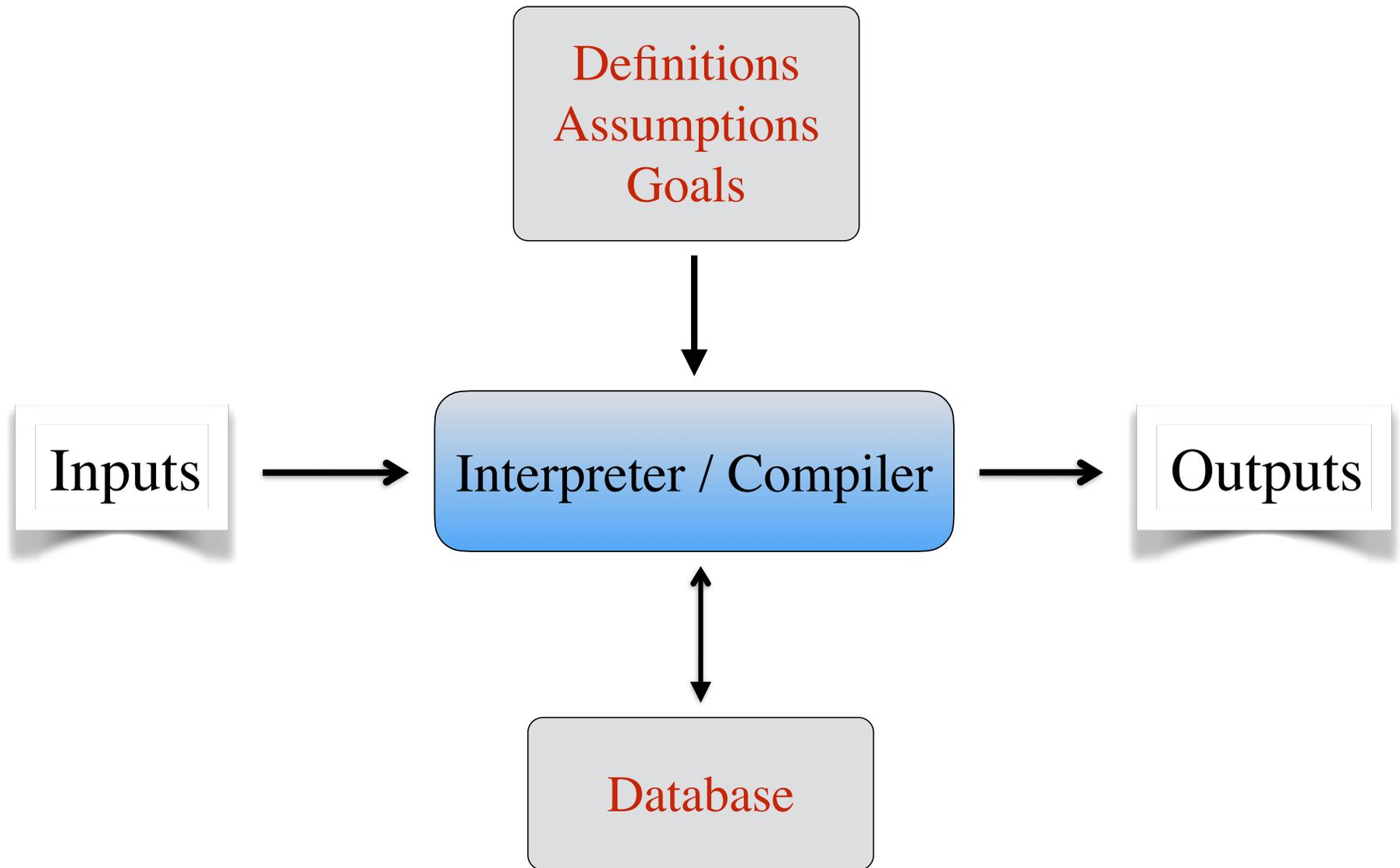
Outputs

Database

Data Structures



Logic Program



Specification = Program

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

Traditional 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



Versatility

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.

McCarthy's Example



McCarthy's Example



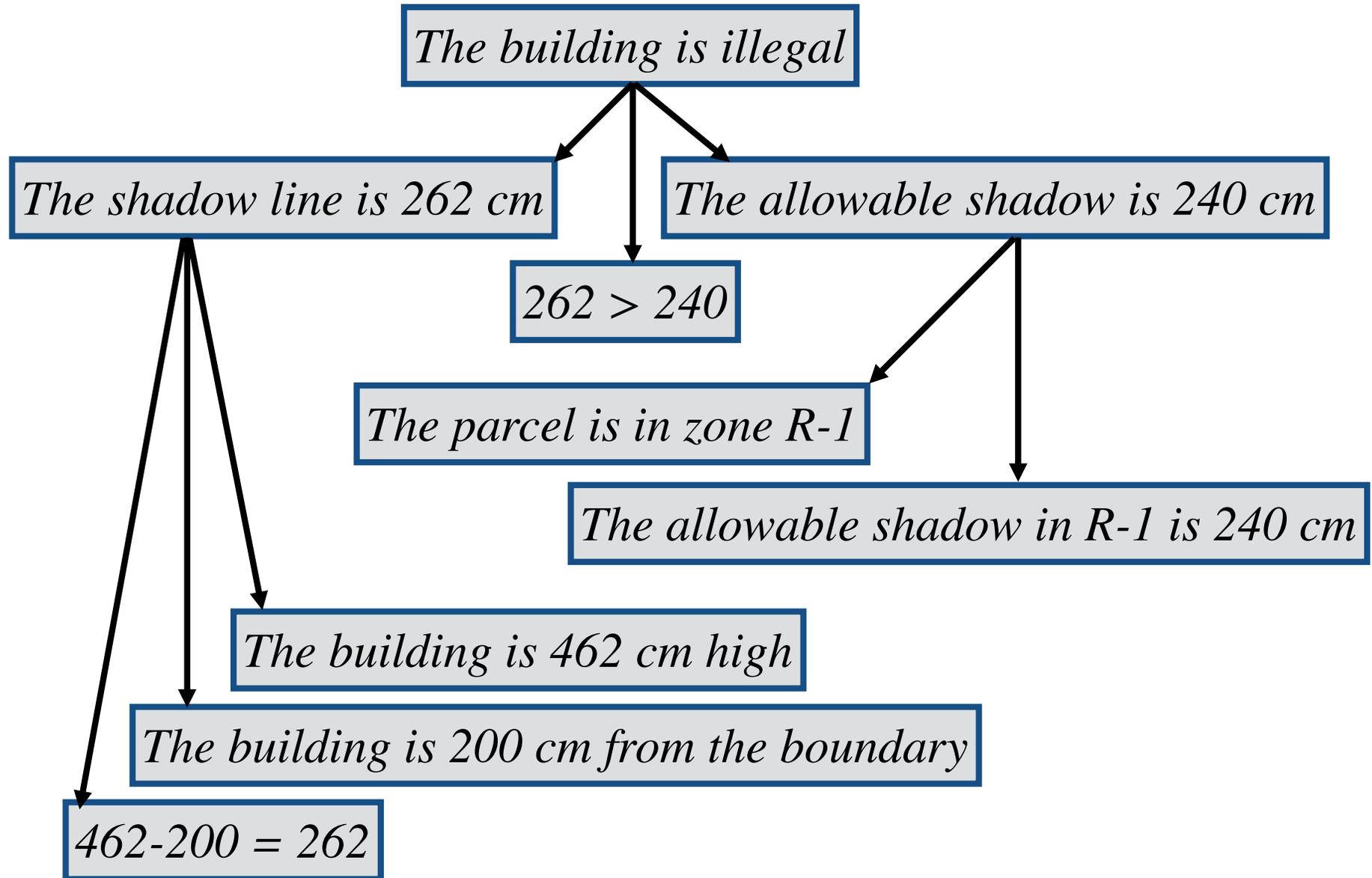
McCarthy's Example



©DESIGNALIKIE

©DESIGNALIKIE

Explanation



Explanations of Results

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.

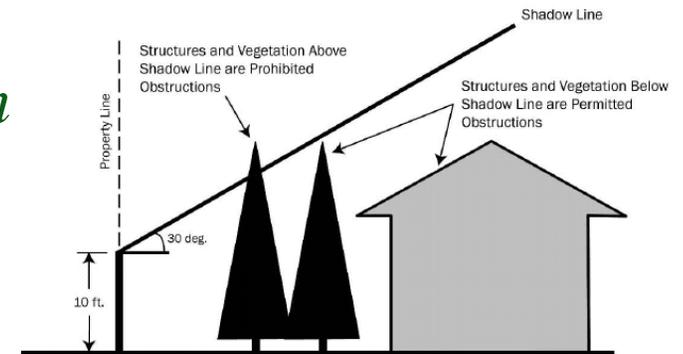


Figure 4. Permitted and Prohibited Obstructions

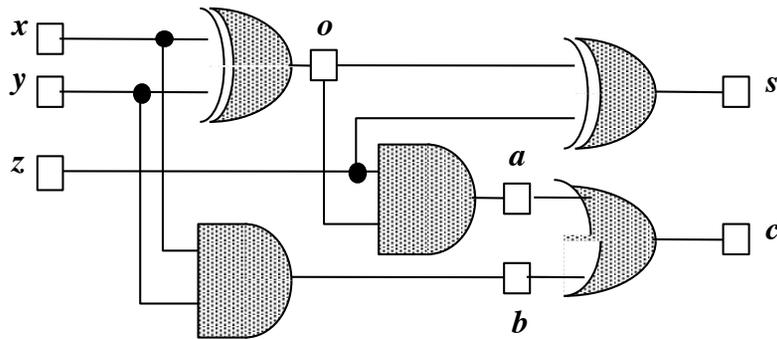
What is the allowable shadow line?

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

Engineering

Circuit:



Description:

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

$$a \Leftrightarrow z \wedge o$$

$$b \Leftrightarrow x \wedge y$$

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

$$c \Leftrightarrow a \vee b$$

Applications:

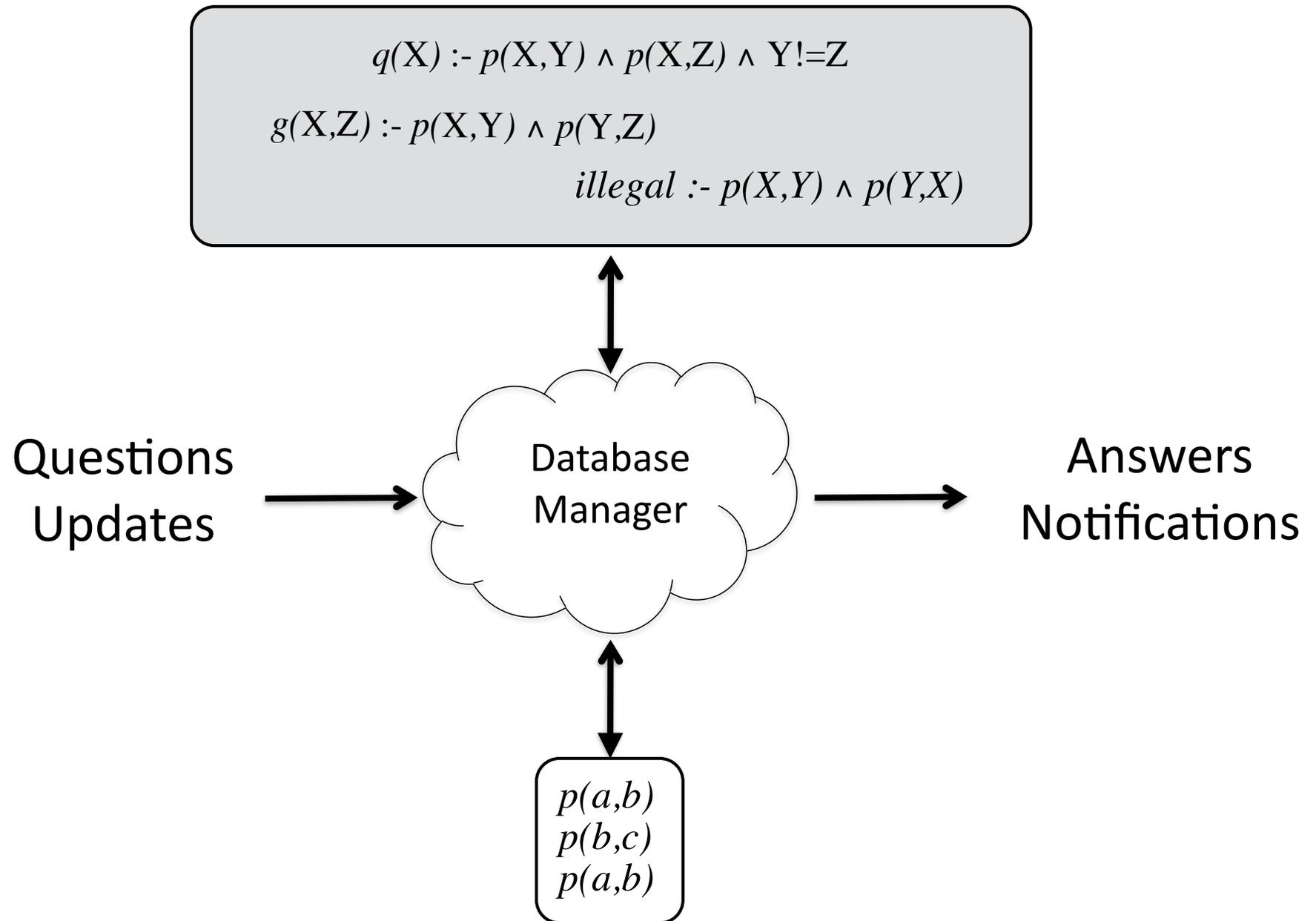
Simulation

Configuration

Diagnosis

Test Generation

Deductive Databases



Interactive Web Pages (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 Geneserth](#)
Webpage

Comments and complaints to 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

First Name **Middle Name** **Last Name** **Suffix**
Nick Name
Date of Birth
Gender
Biography

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:** HCP? Coterm?

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/):

- 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 adviser 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/. 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

Search: 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

Sales and Service Invoicing External Invoicing Project Invoicing Miscellaneous Invoicing

Communication for Customer Invoicing 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

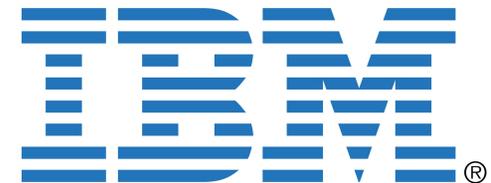
"External Invoicing" **New** within "Customer Invoicing"

De-selected by user

"Project Invoicing" **Will appear twice** within "Customer Invoicing"

Program Sheet

Business Rules and Workflow



Computational Law

Computational Law is that branch of legal informatics concerned with the mechanization of legal reasoning.

Automated Compliance Management

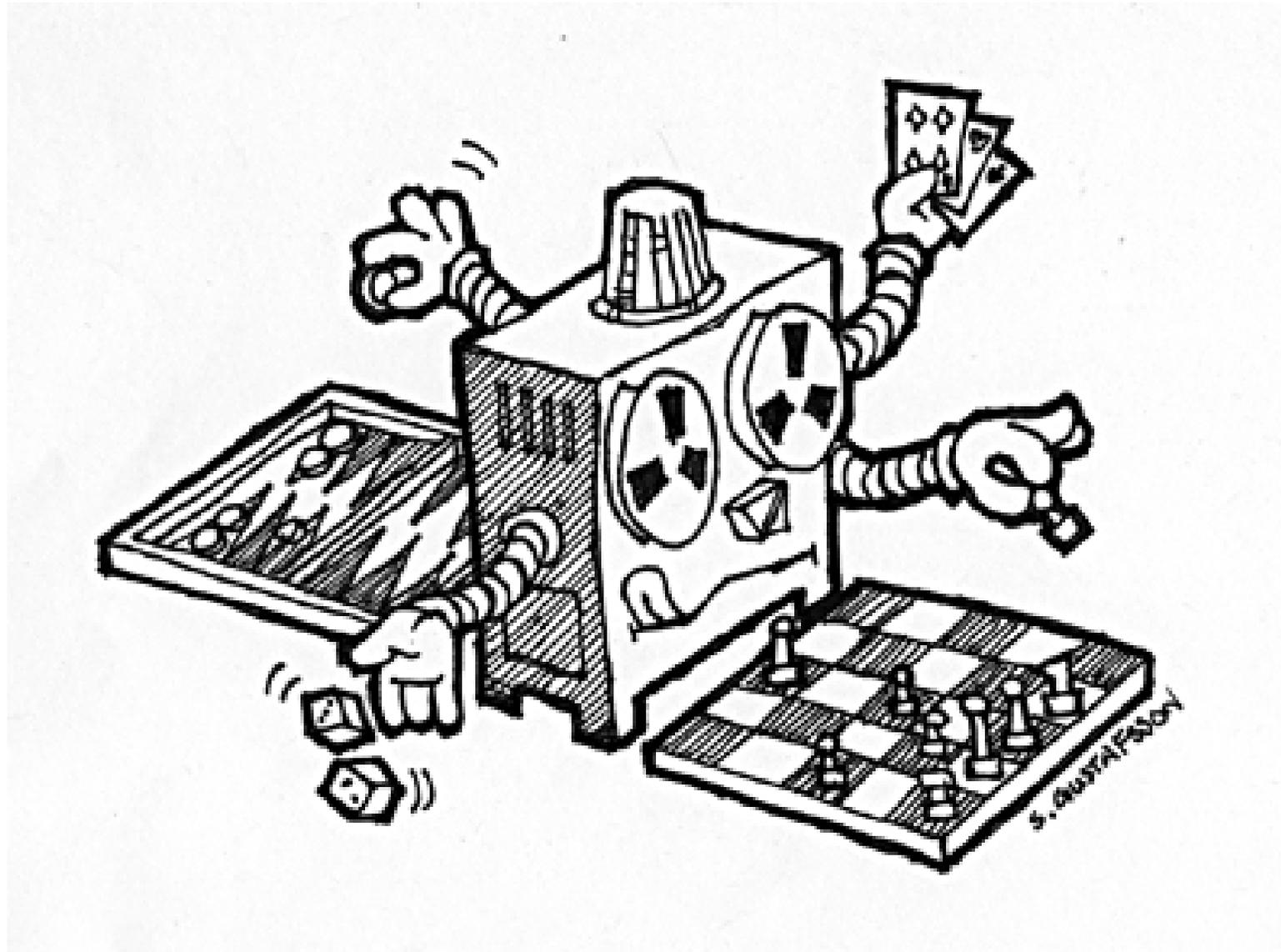
Legal analysis of specific cases

Planning for compliance in specific cases

Analysis of regulations for overlap, consistency, etc.

Portico

General Game Playing



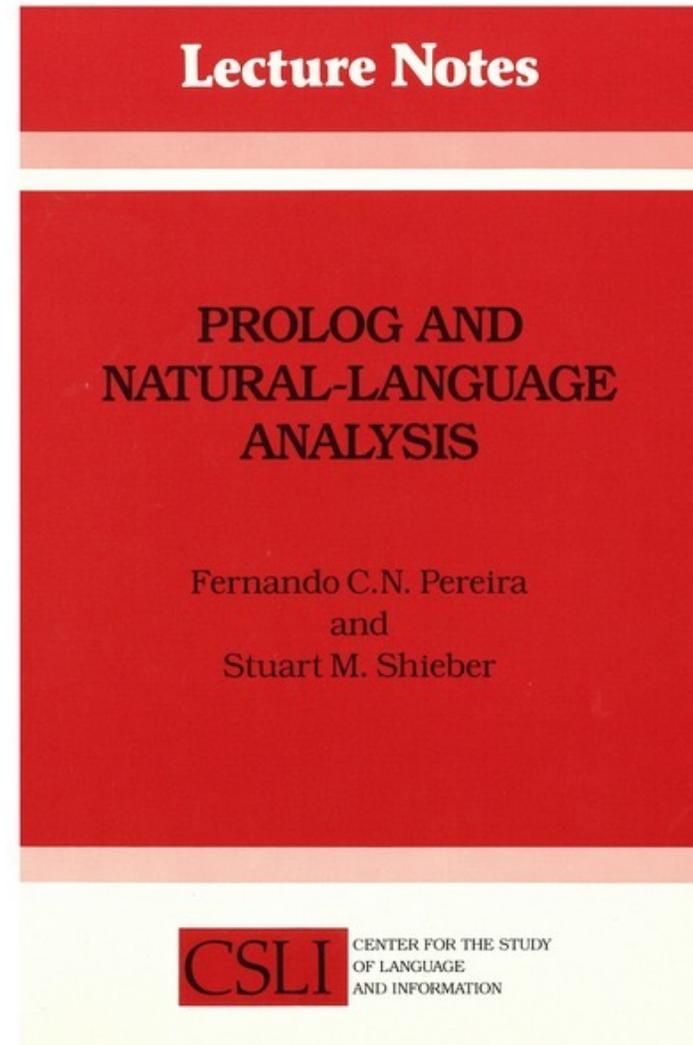
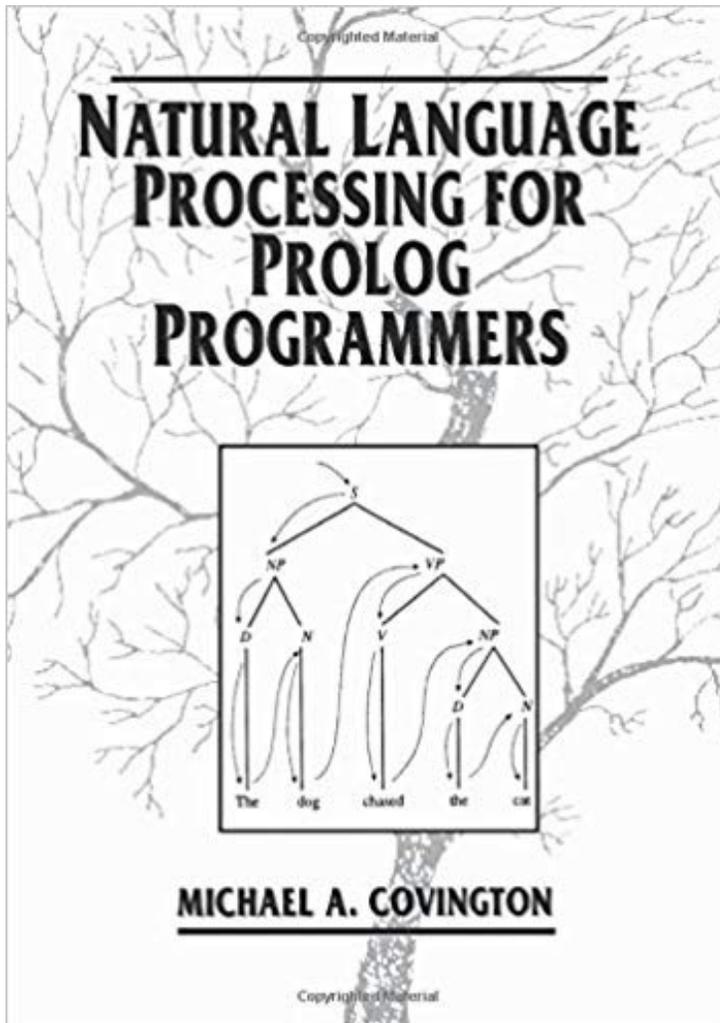
General Game Playing



Pelican Hunters

Non-Successes

Natural Language Processing



Theorem Proving



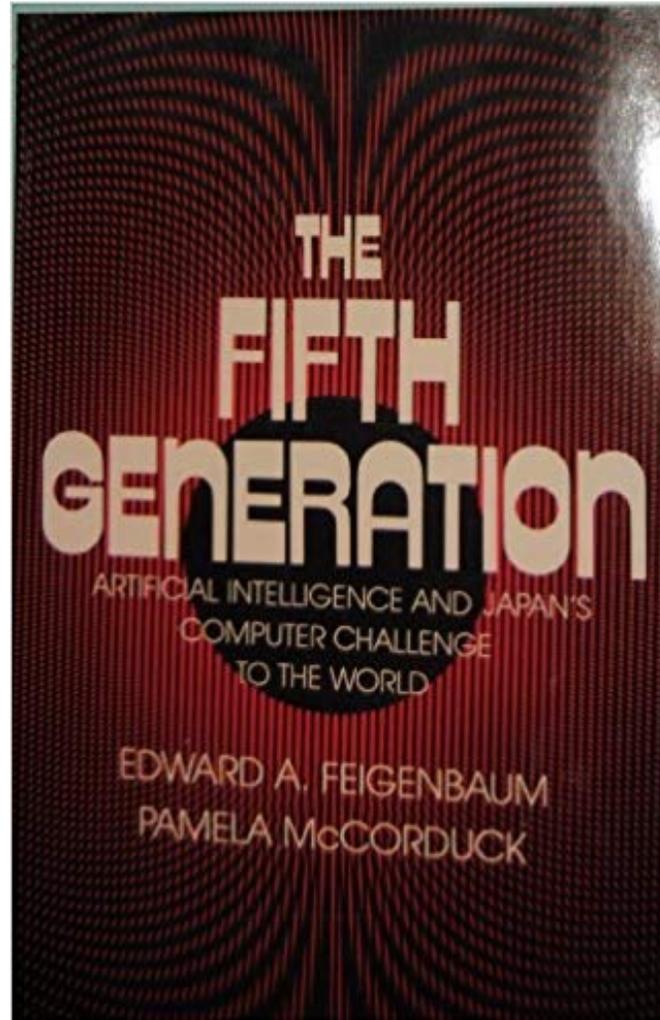
PTTP

means

Prolog Technology Theorem
Prover

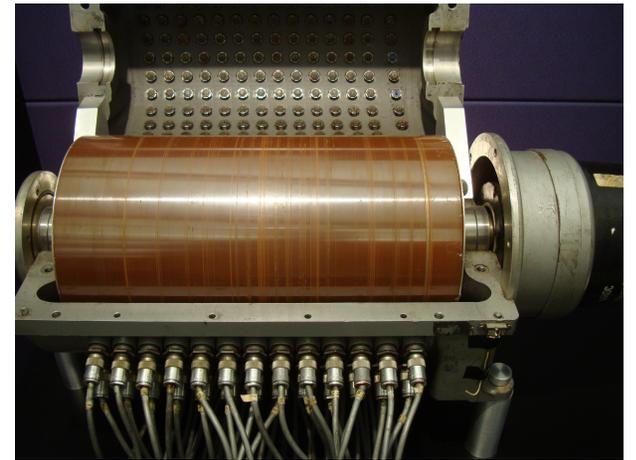
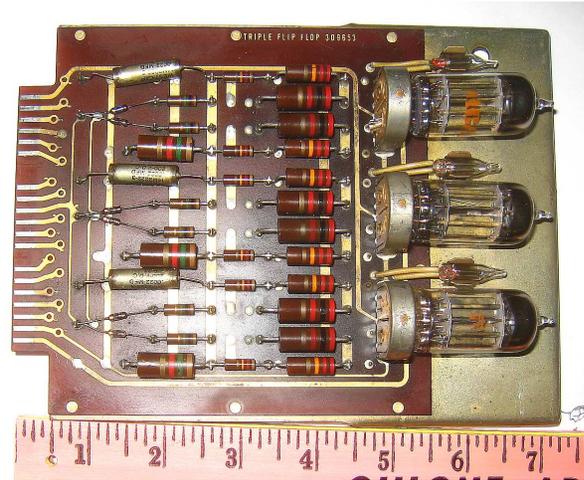
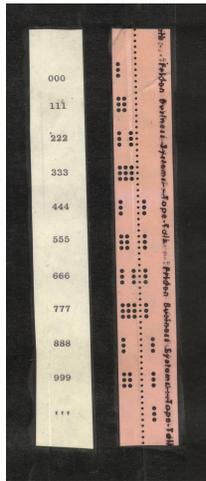
by [acronymsandslang.com](https://www.acronymsandslang.com)

Japan's Fifth Generation Project



History

LGP-30 (1GL)



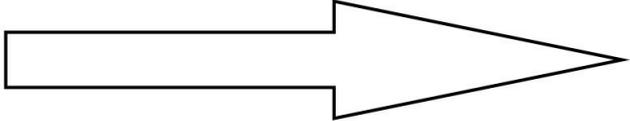
Assembly Language (2GL)

Assembly Language

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

Programmer

Assembler + Linker

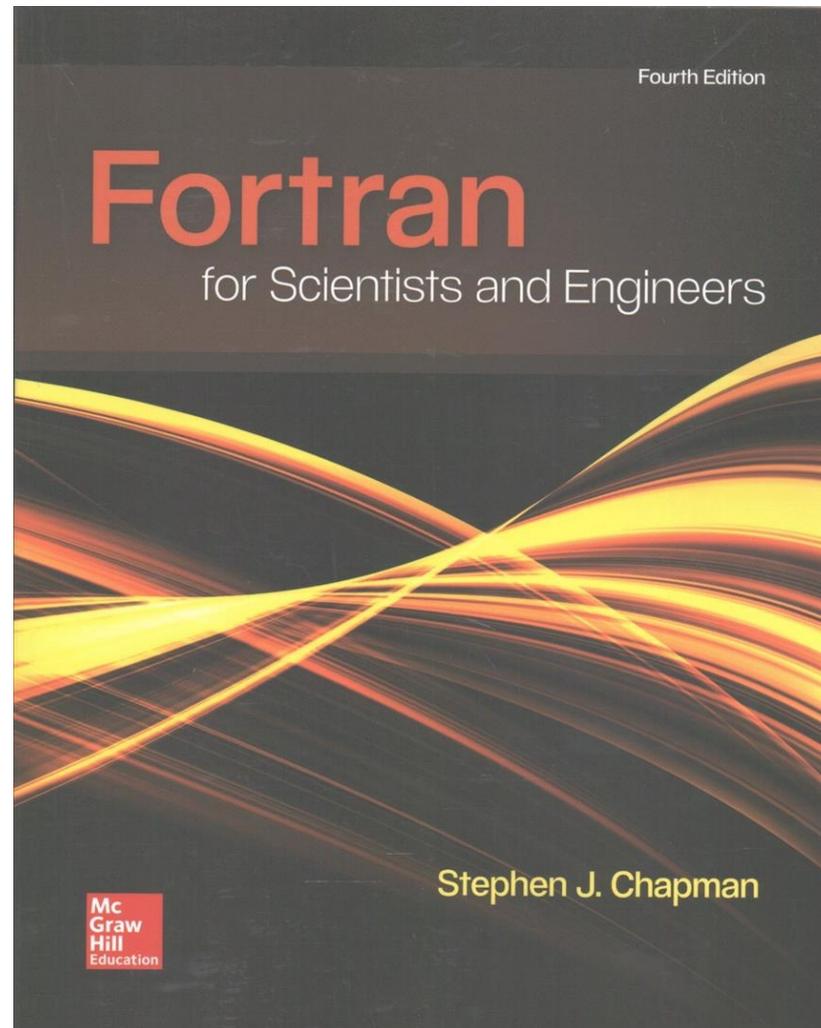


Machine Language

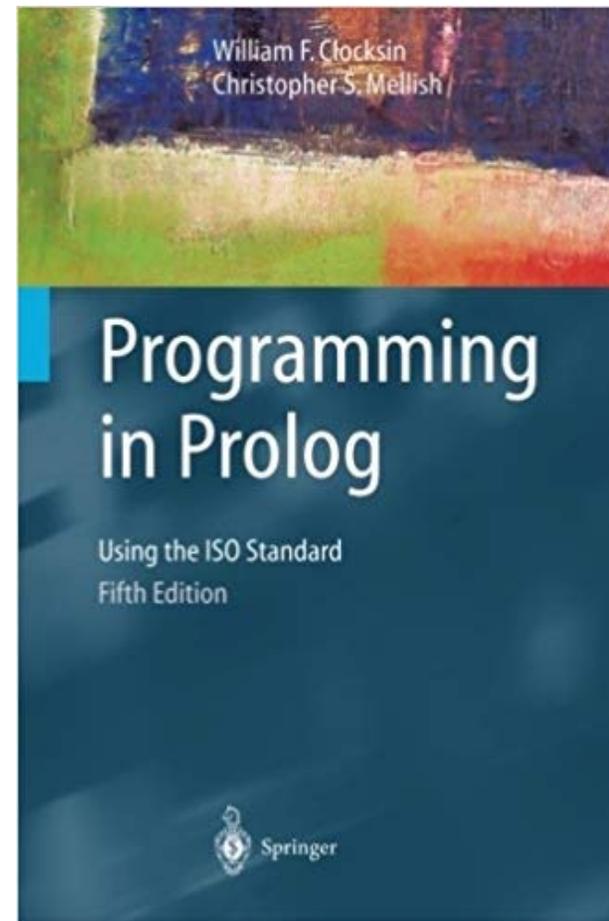
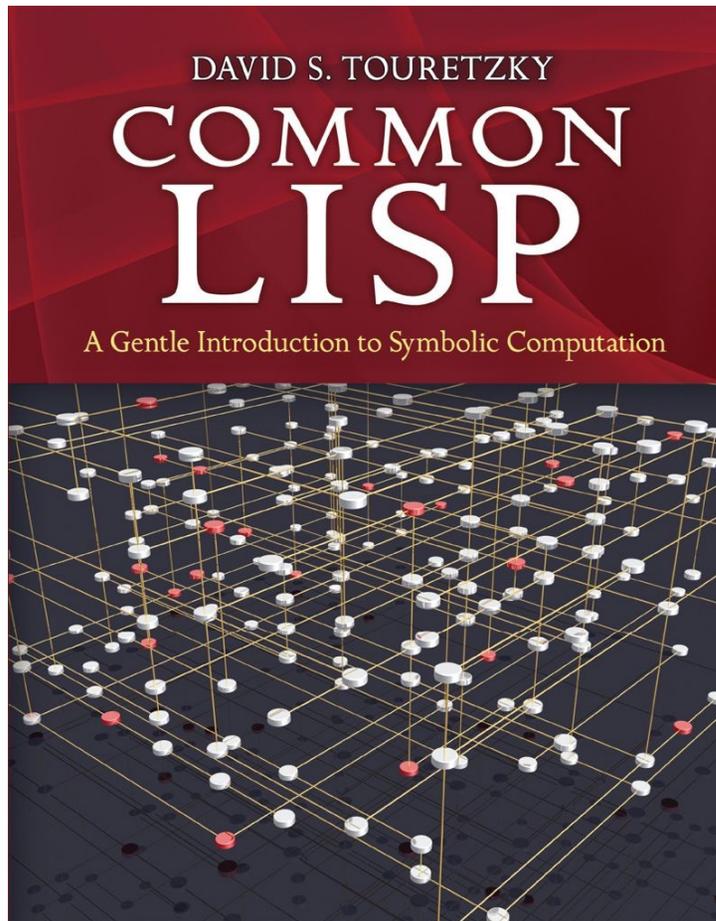
```
100101011001  
010011111011  
111010101101  
01010101010
```

Processor

Higher Level Languages (3GL)



Symbolic Processing Languages (3GL)



Imperative Programming Languages



JavaScript

C++



Ruby



Scala



C#

Declarative Programming Languages

Declarative

vs.

Imperative

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

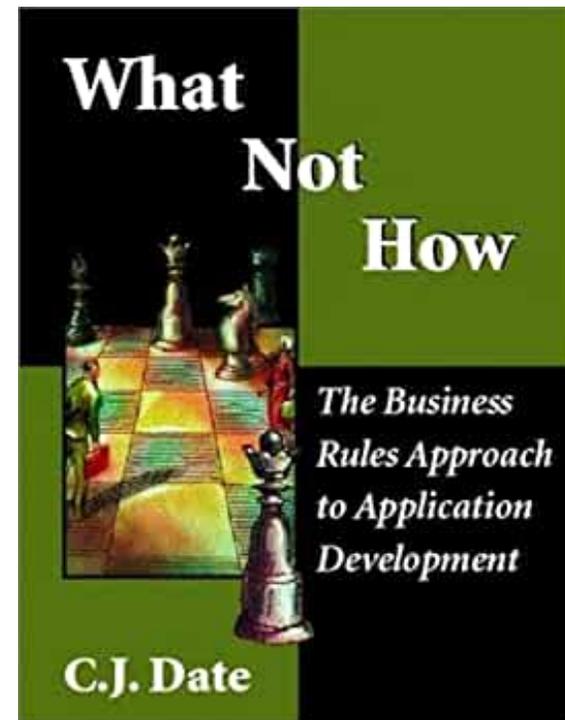
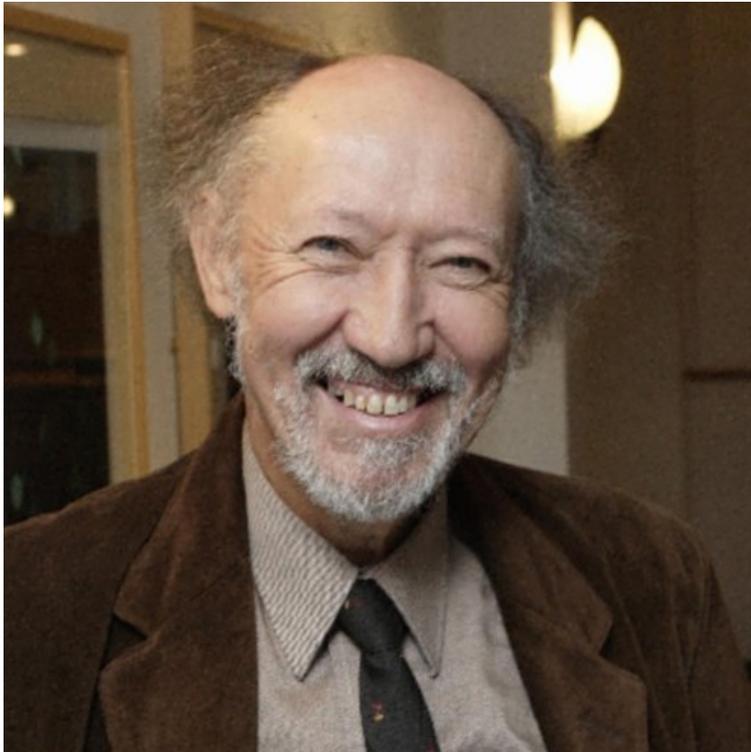
Ed Feigenbaum



*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

Chris Date (Mr. SQL)



Val Huber



Years of experience have taught us ... it takes far too long to turn a relatively simple set of requirements into a system that meets the user needs.

*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

Schedule

Mar 31 Introduction
Apr 2 Datasets
7 Relational Queries
9 Query Evaluation
14 Functional Queries
16 Transitions
21 View Definitions
23 View Evaluation
28 Simple Examples
30 Lists, Sets, Trees

May 5 Function Definitions
7 Operation Definitions
12 Applications
14 Worksheets
19 Incomplete Data
21 Constraint Satisfaction
26 Beyond LP
28 Past Projects
Jun 2 Project Reports
4 Project Reports

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

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

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

Grades

Numerical Score

15% for Assignment 1

20% for each of Assignments 2, 3, 4

25% 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>



