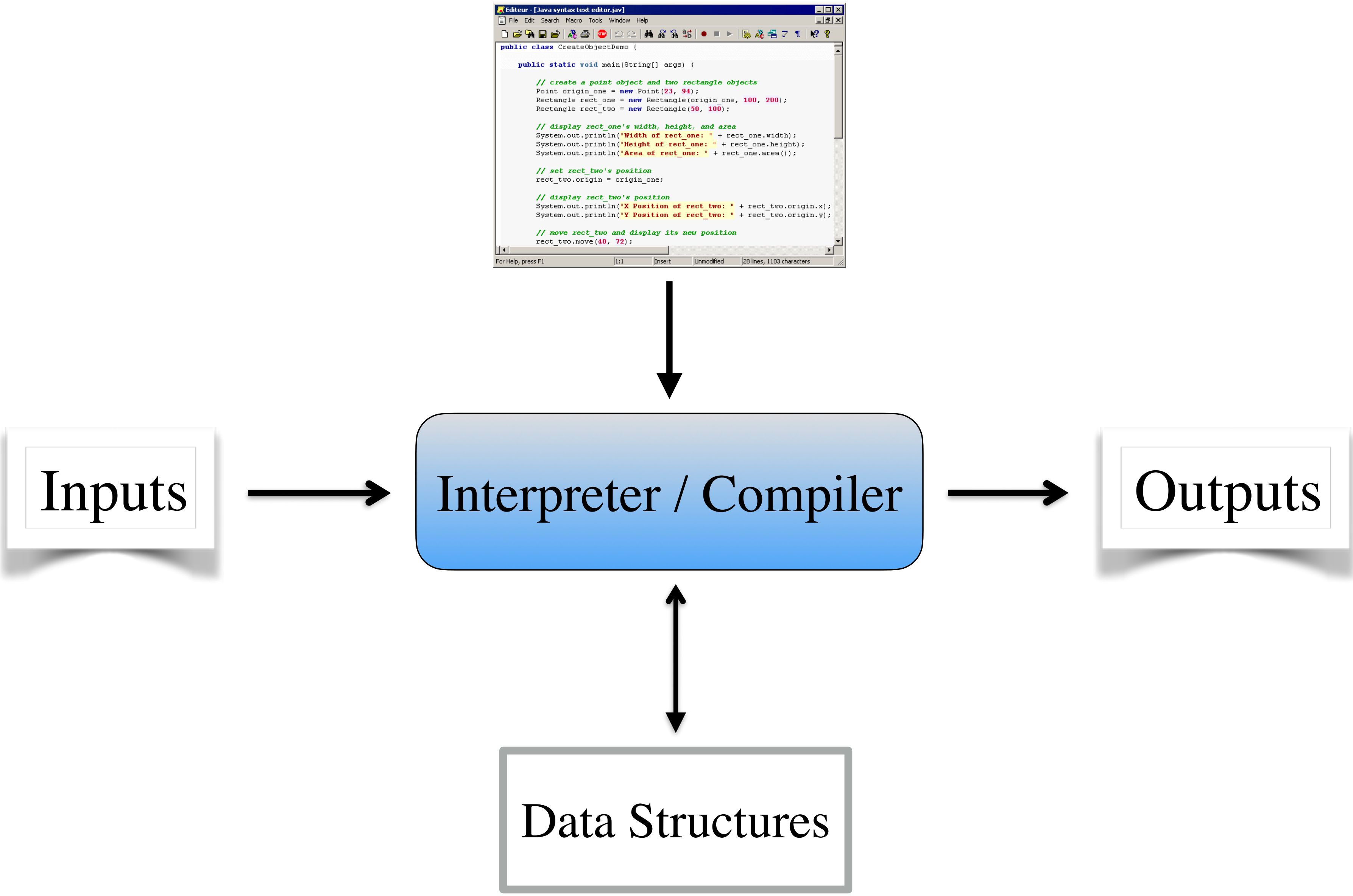
Logic Programming
Introduction

Michael Genesereth
Computer Science Department
Stanford University

Lecture will begin at ~1:35 PDT.

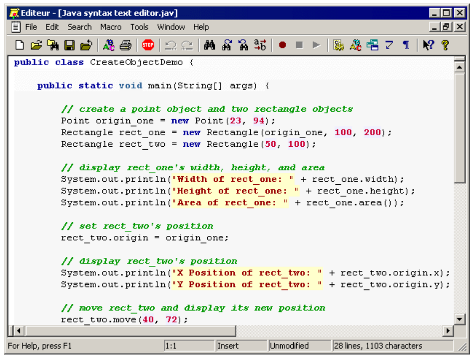
Programmed Computer System



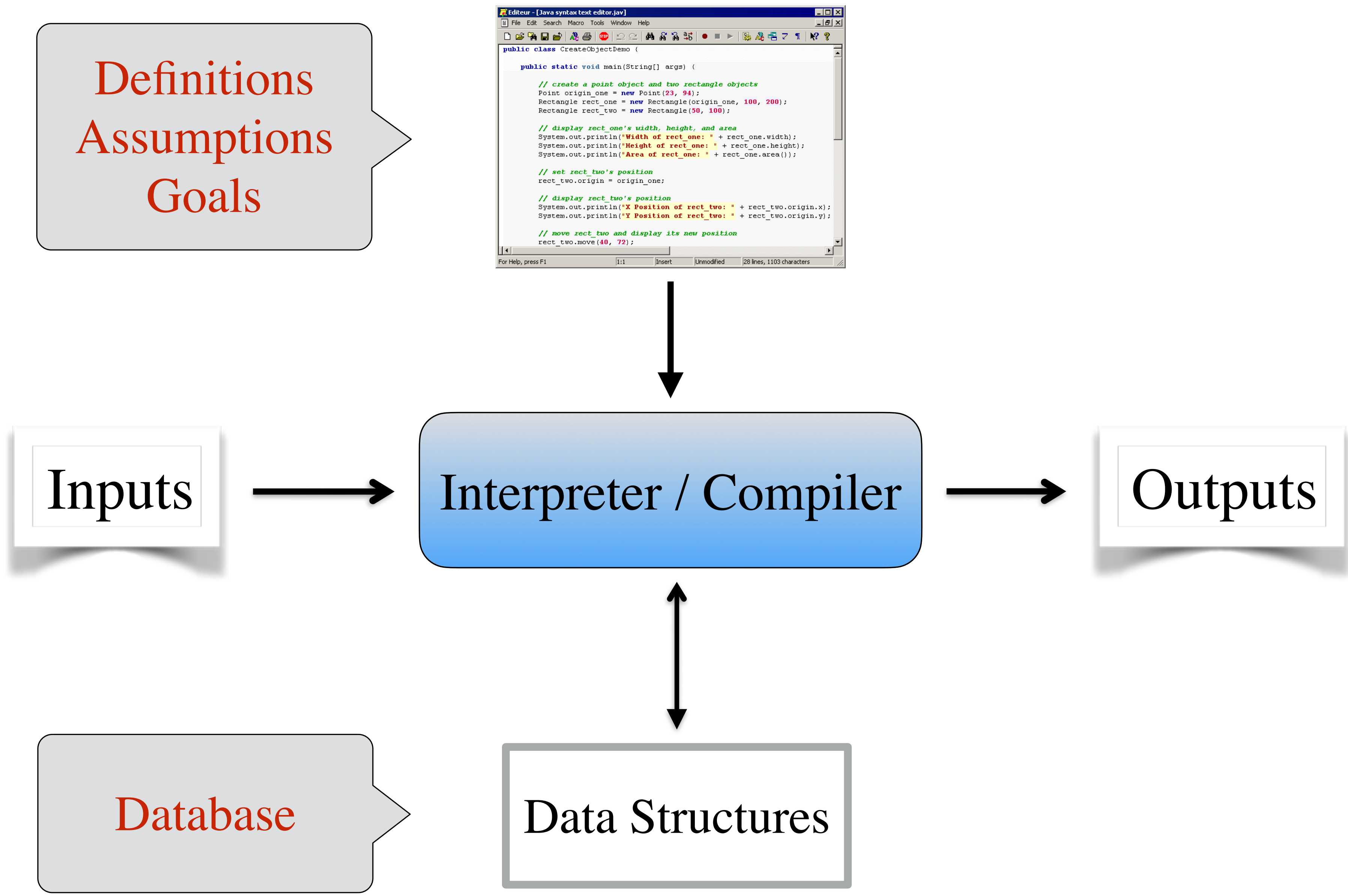
Specifications versus Programs

Definitions
Assumptions
Goals

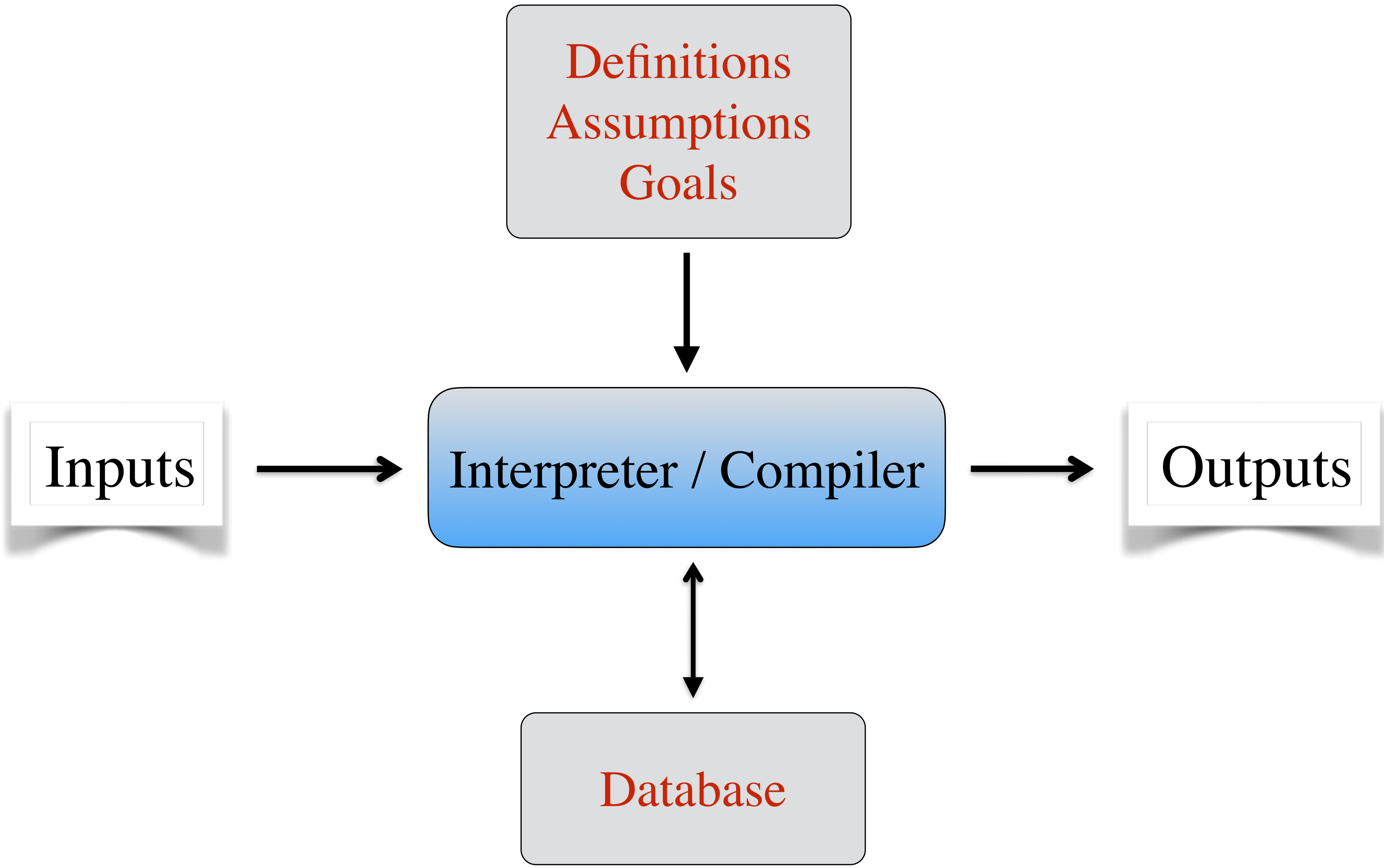
→



Traditional Program



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

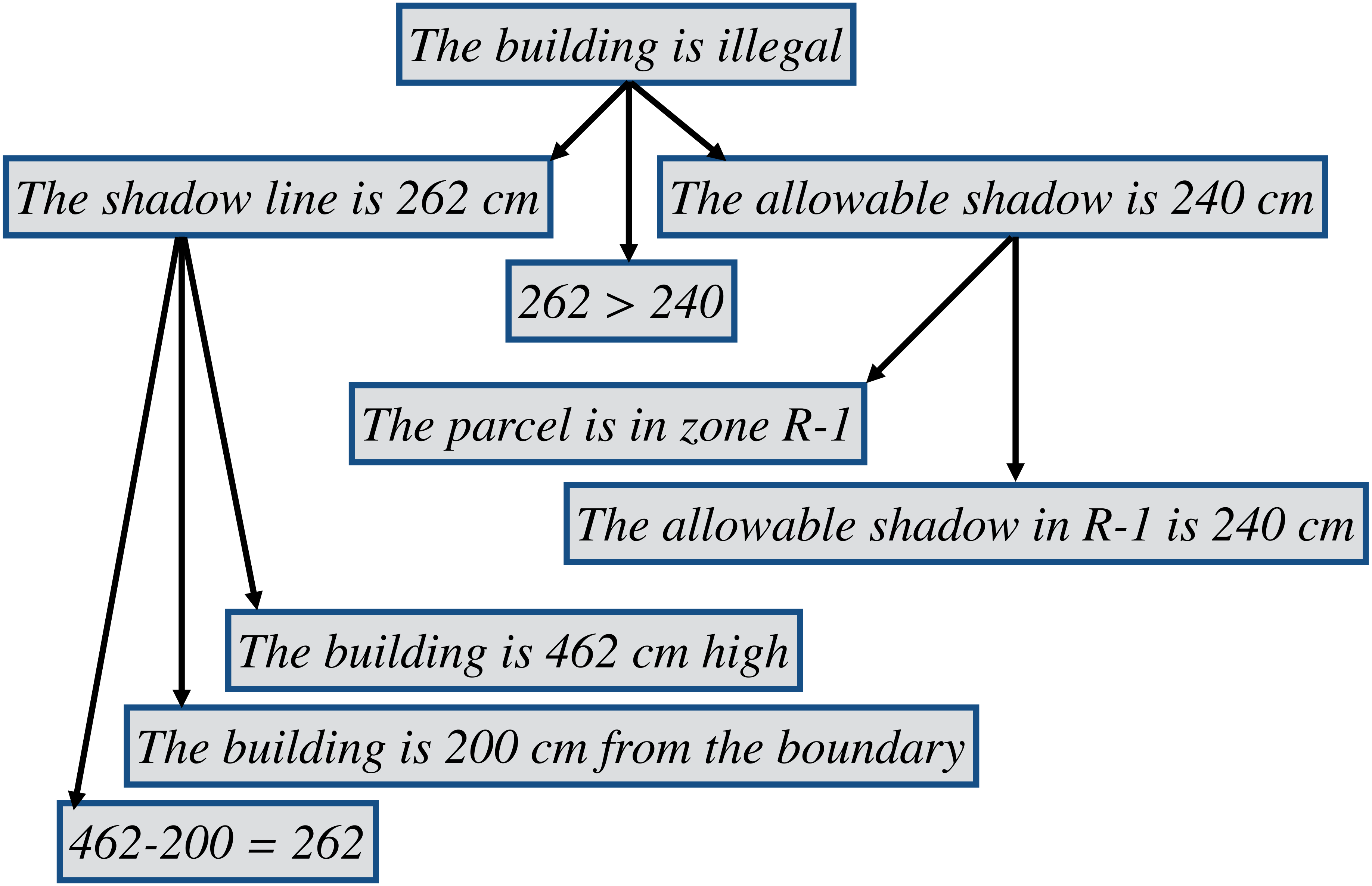
Determine whether Art is the grandparent of Cal.
Determine all of the grandchildren of Art.
Compute the grandparents of Cal.
Compute all grandparent-grandchildren pairs.

McCarthy's Example

McCarthy's Example

McCarthy’s Example

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.

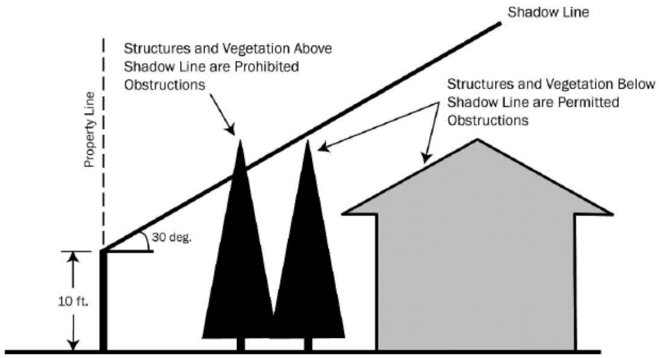


Figure 4. Permitted and Prohibited Obstructions

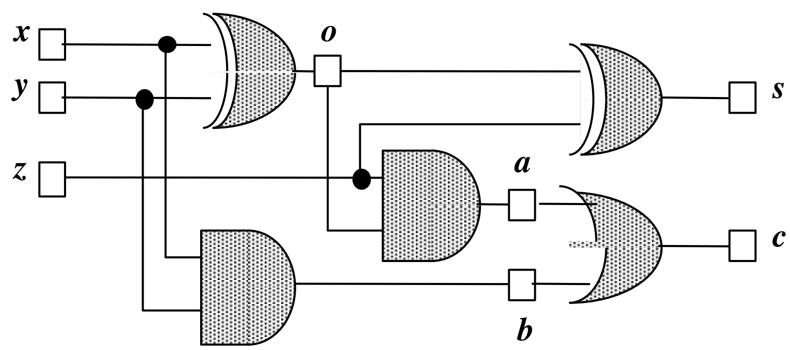
What is the allowable shadow line?

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:

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

$$a \Leftrightarrow z \wedge o$$

$$b \Leftrightarrow x \wedge y$$

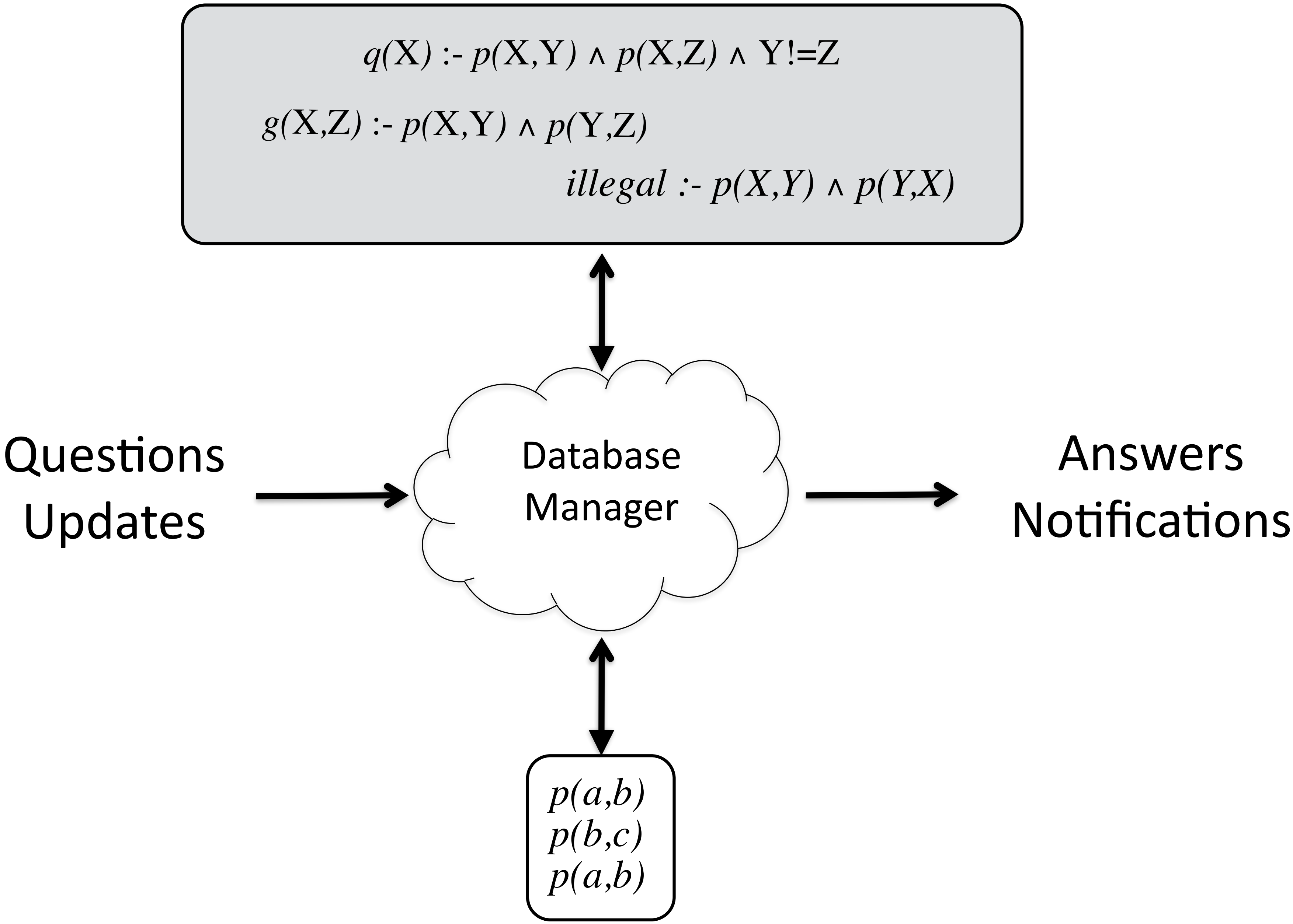
$$s \Leftrightarrow (o \wedge \neg z) \vee (\neg o \wedge z)$$

$$c \Leftrightarrow a \vee b$$

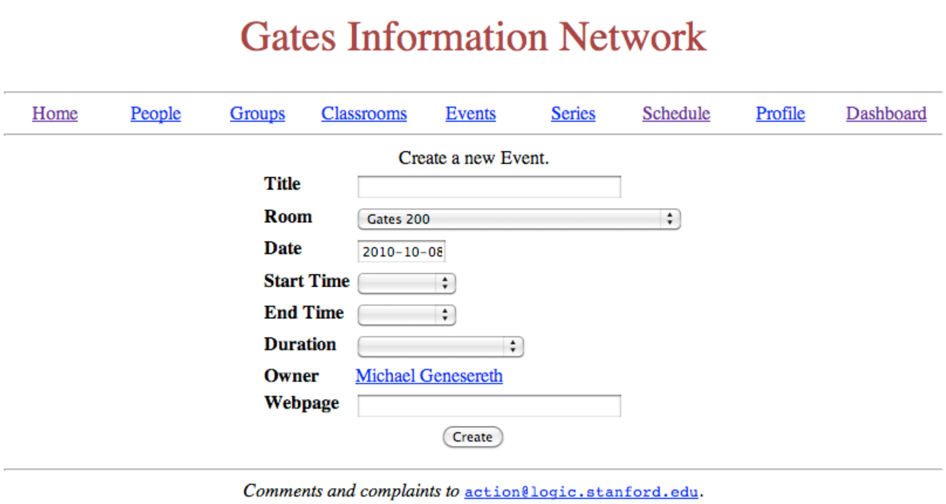
Applications:

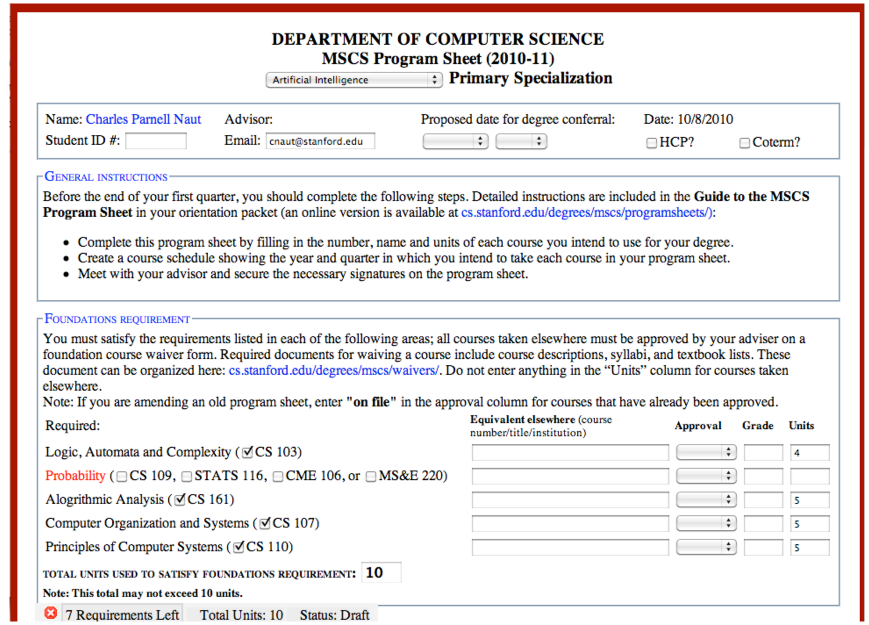
- Simulation
- Configuration
- Diagnosis
- Test Generation

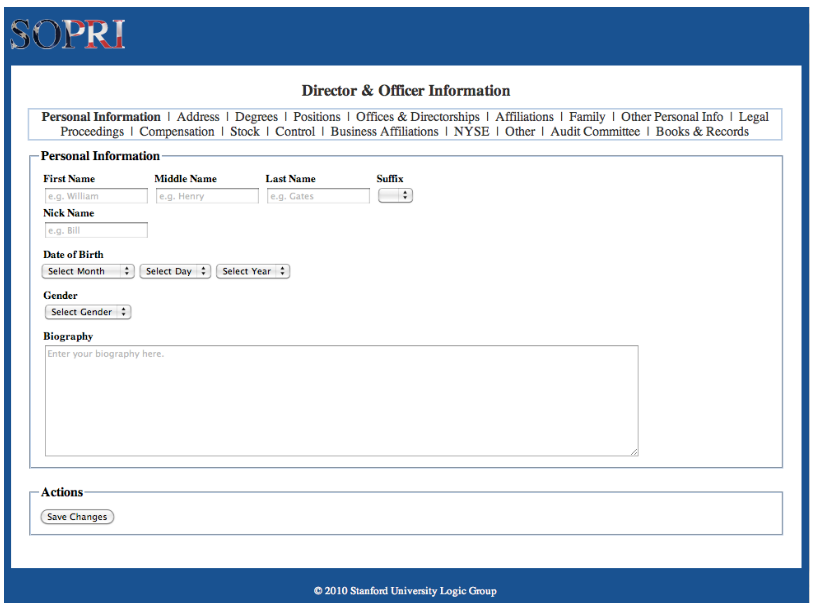
Deductive Databases

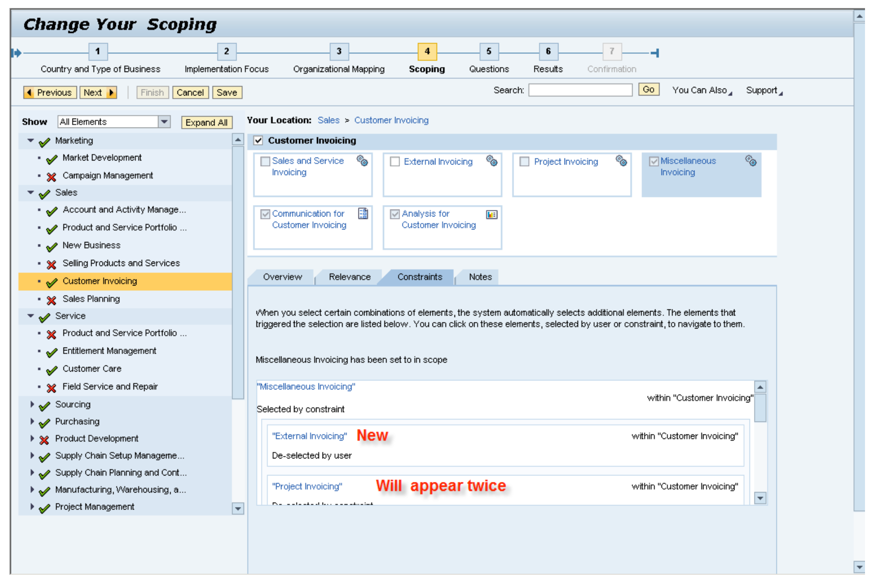


Interactive Web Pages (Worksheets)









<u>Program Sheet</u>

Business Rules and Workflow

SAP® ORACLE® IBM®

Computational Law

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

Automated Compliance Management

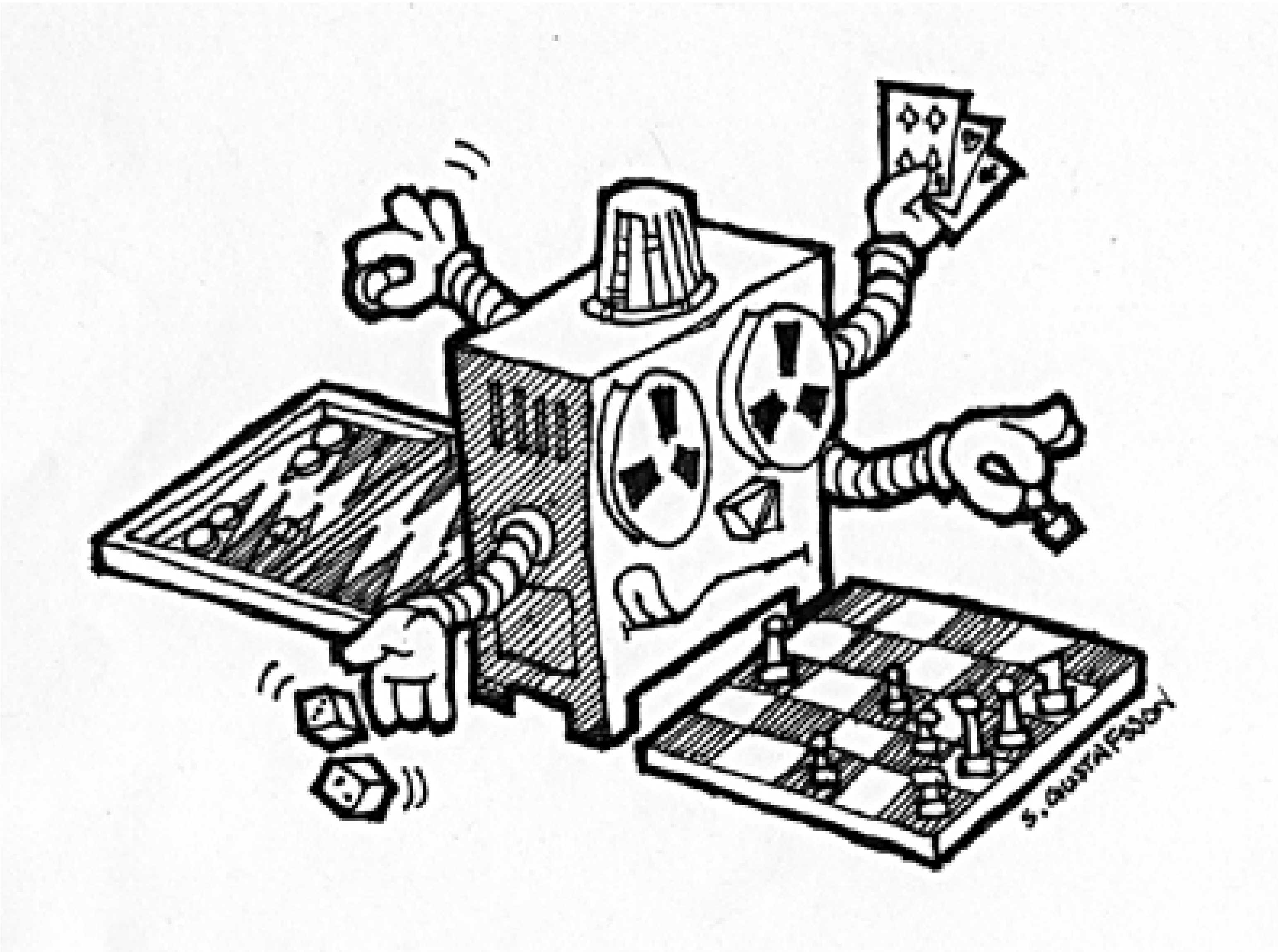
Legal analysis of specific cases

Planning for compliance in specific cases

Analysis of regulations for overlap, consistency, etc.

Portico

General Game Playing

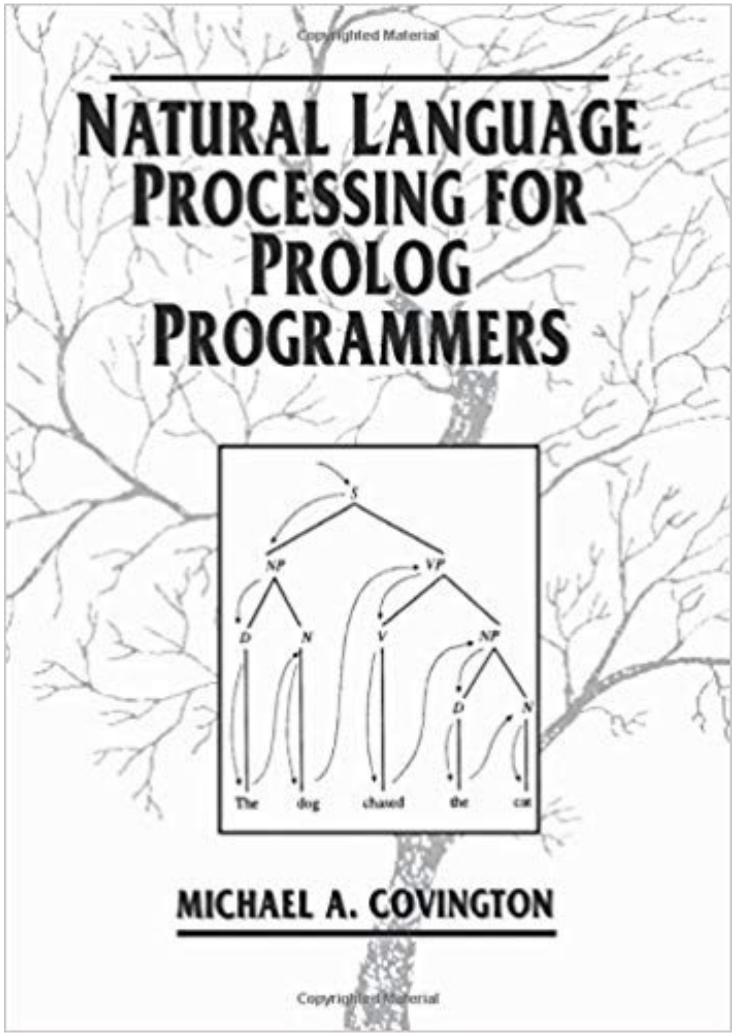


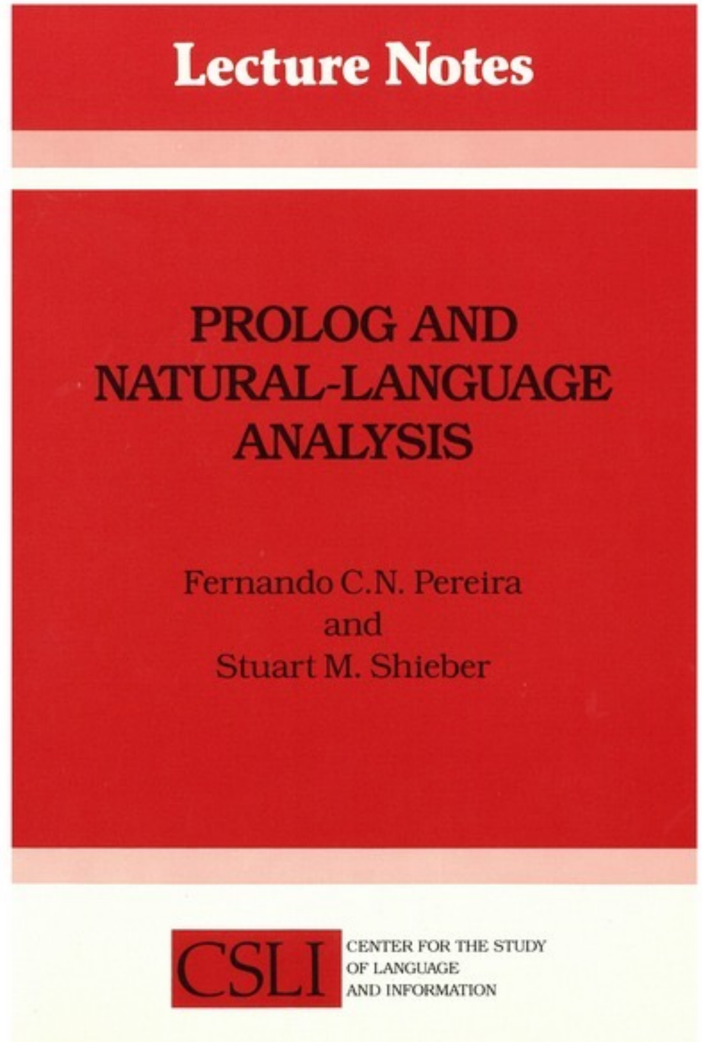
General Game Playing

Pelican Hunters

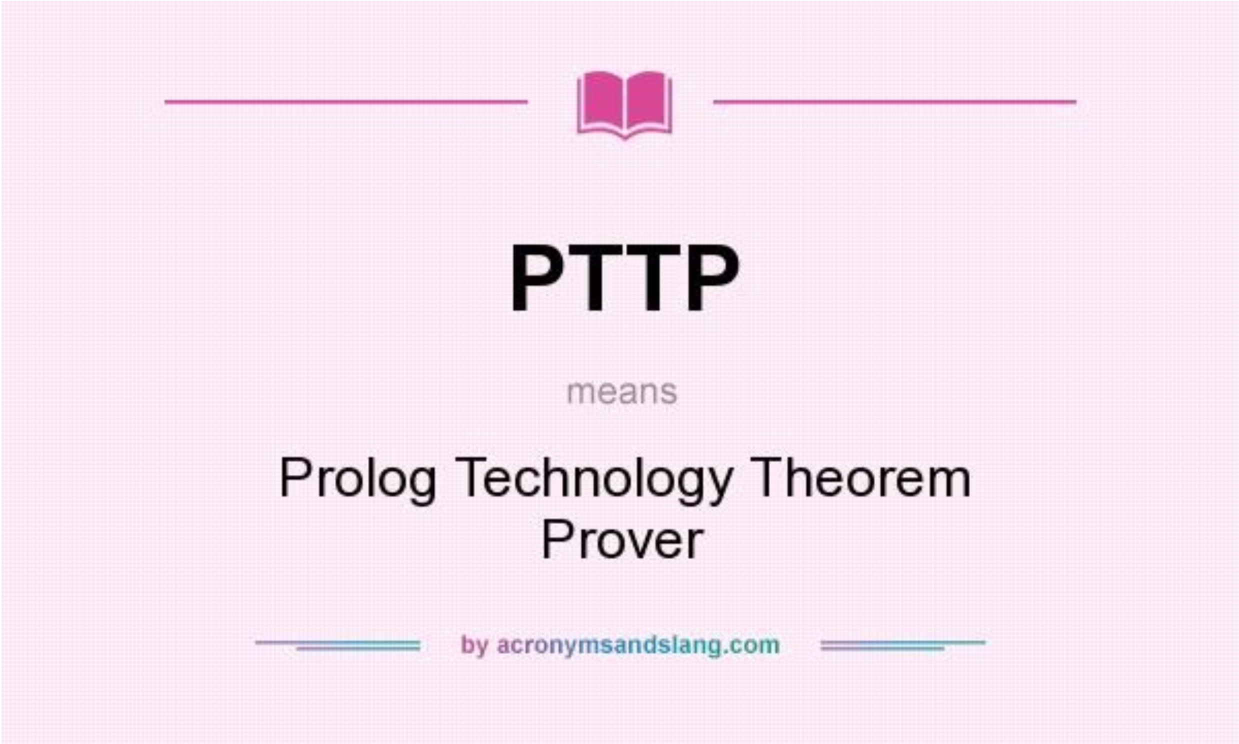
Non-Successes

Natural Language Processing

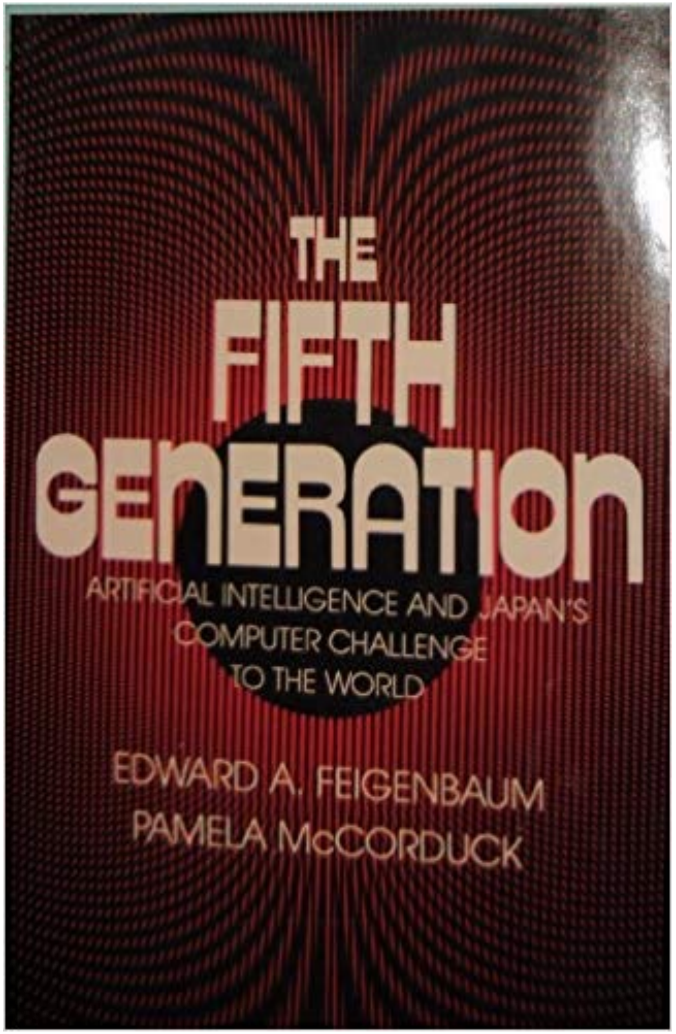




Theorem Proving

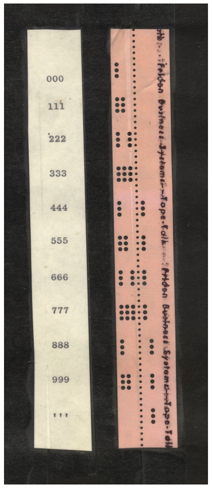


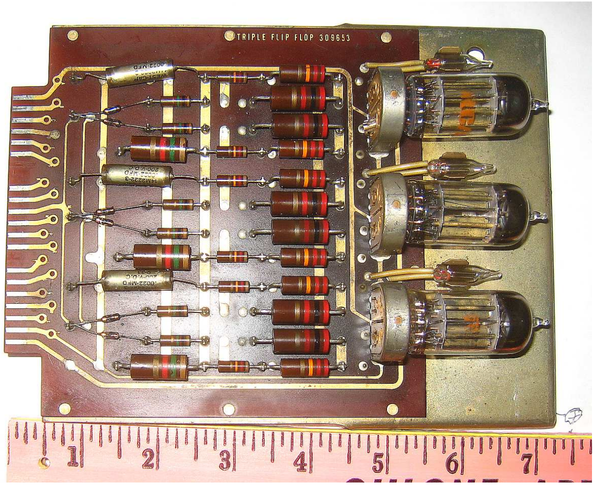
Japan's Fifth Generation Project

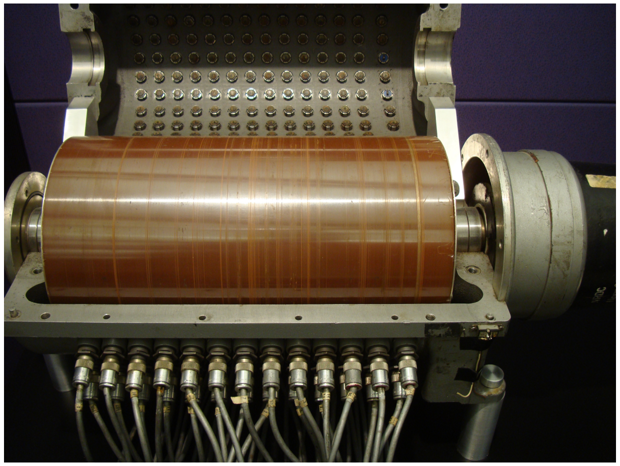


History

LGP-30 (1GL)







IBM 360

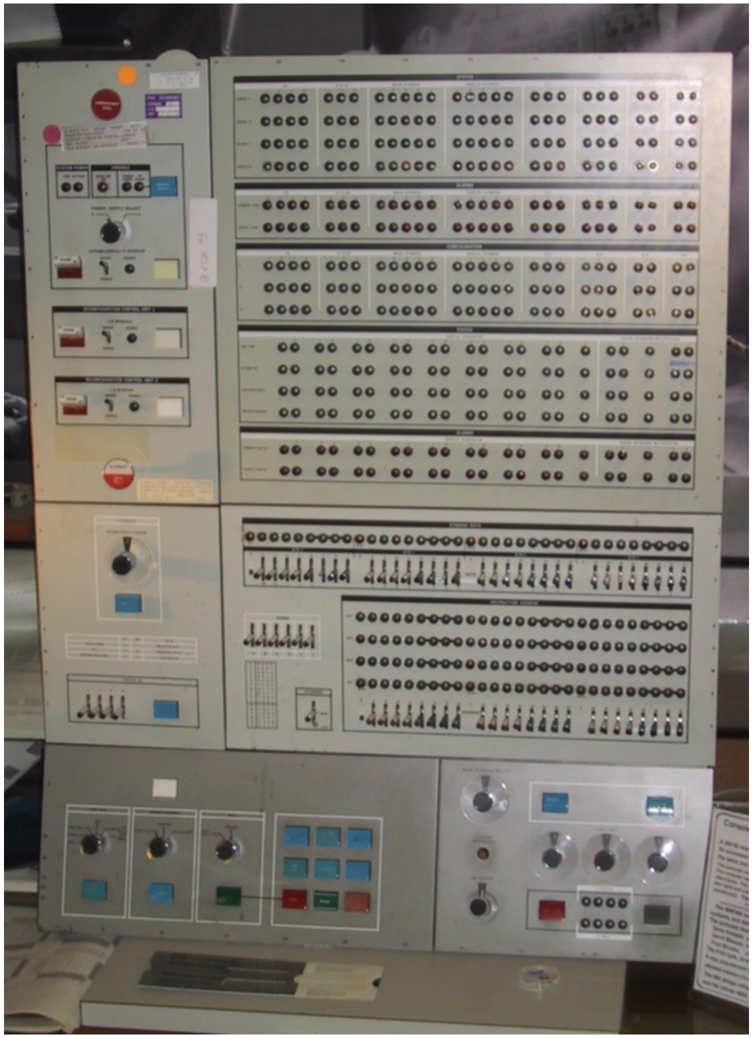
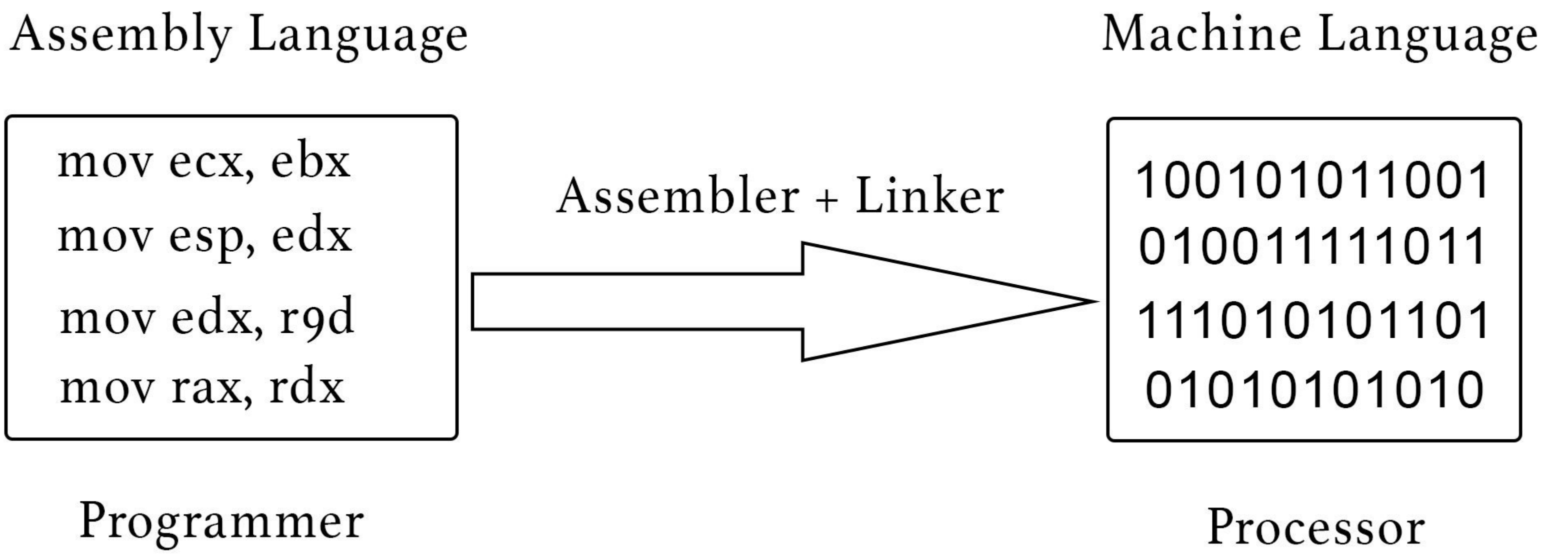
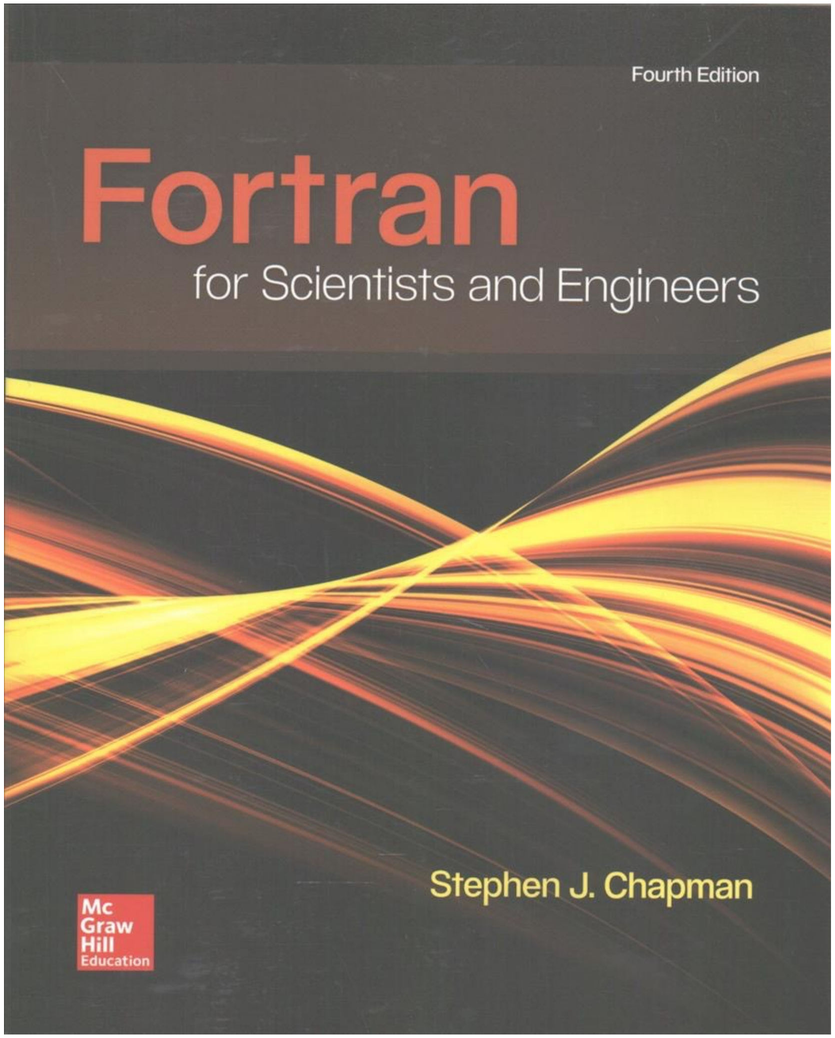


Figure 4. Card Codes and Graphics for 64-Character Set

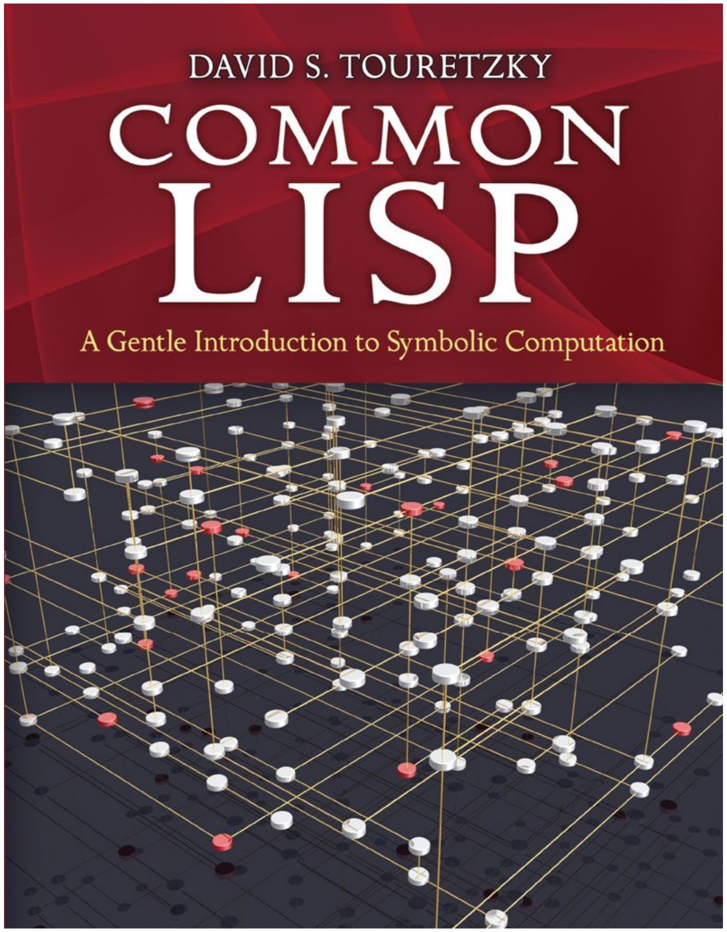
Assembly Language (2GL)

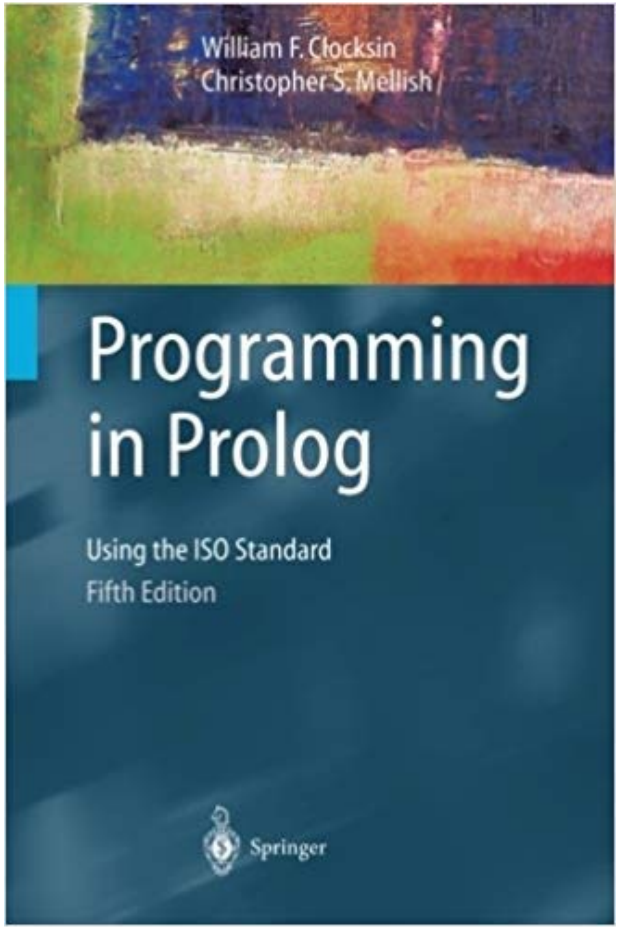


Higher Level Languages (3GL)



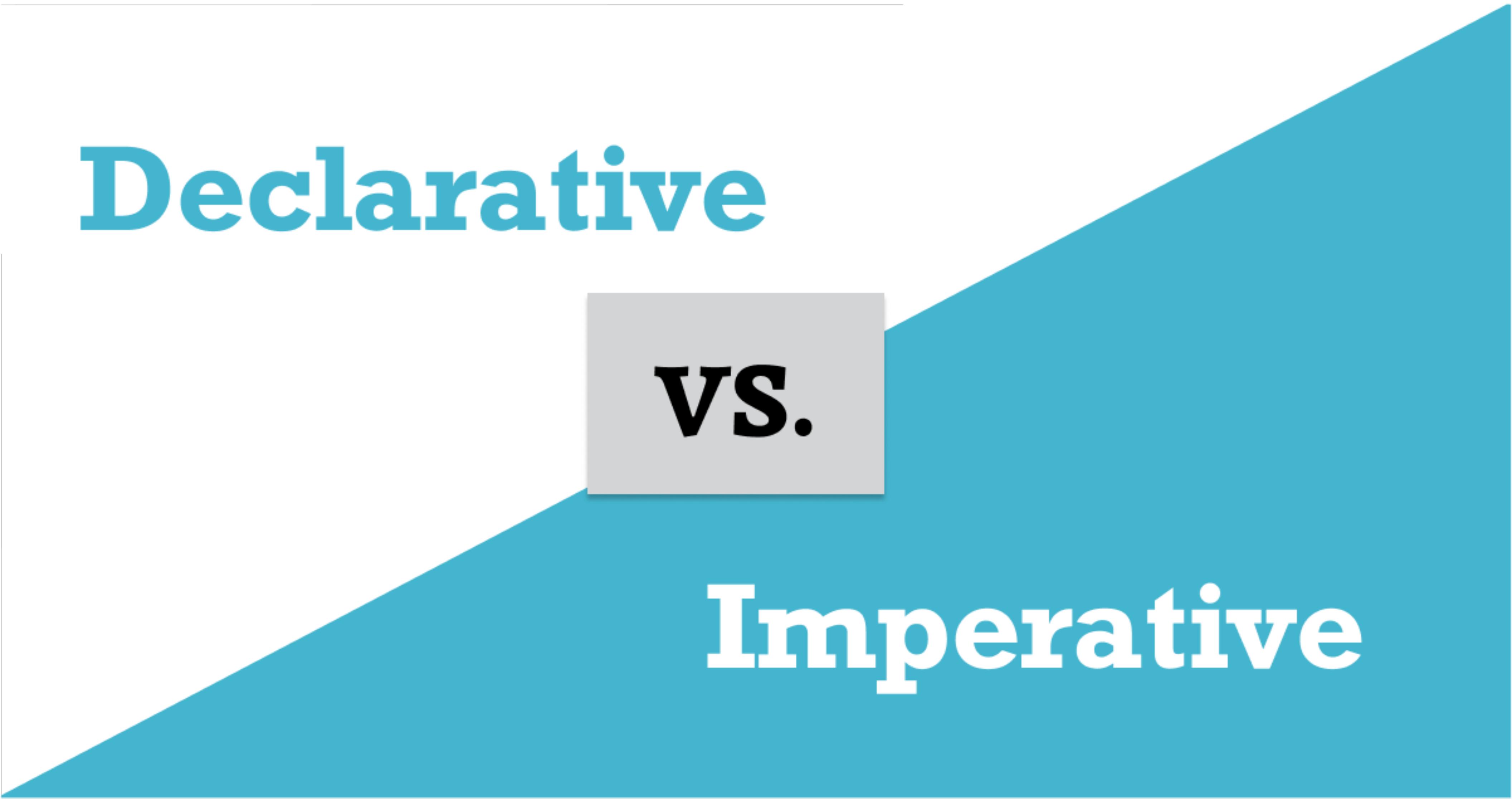
Symbolic Processing Languages (3GL)



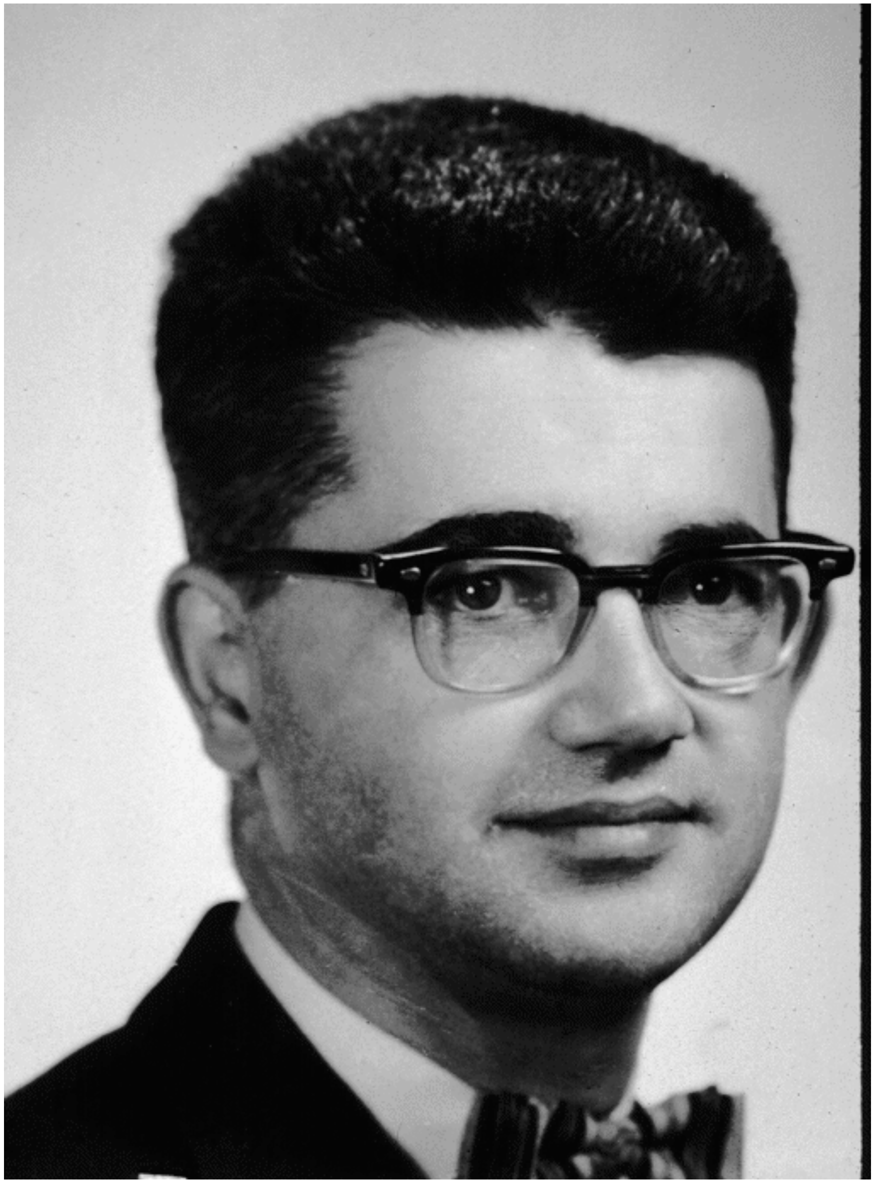


Imperative Programming Languages

Declarative Programming 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.*

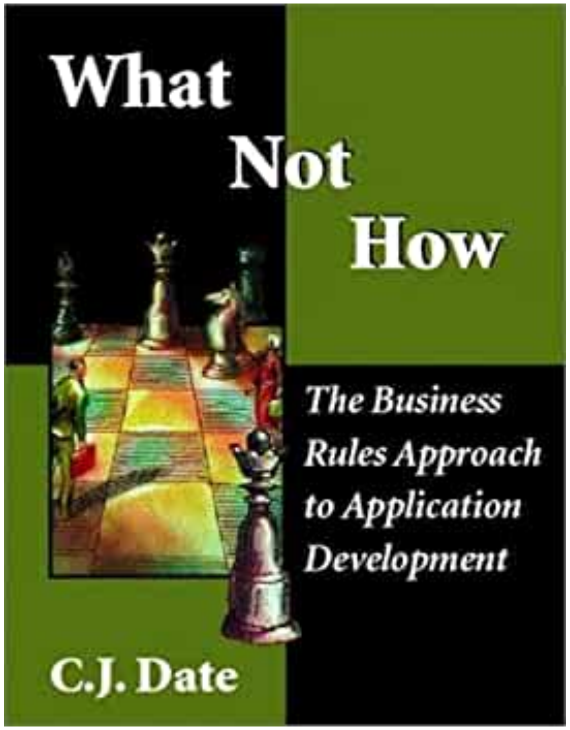
- John McCarthy1958

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

Apr	2	Introduction
	4	Datasets
	9	Queries
	11	Query Examples
	16	Query Evaluation
	18	Query Optimization
	23	View Definitions
	25	View Evaluation
	30	Simple Examples
May	2	Lists, Sets, Trees
May	7	Operation Definitions
	9	Model Management
	14	Reactive Worksheets
	16	Semantic Worksheets
	21	Constraint Satisfcation
	23	Program Synthesis
	28	Extensions and Advances
	30	Past Projects
Jun	4	Project Reports
	6	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 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

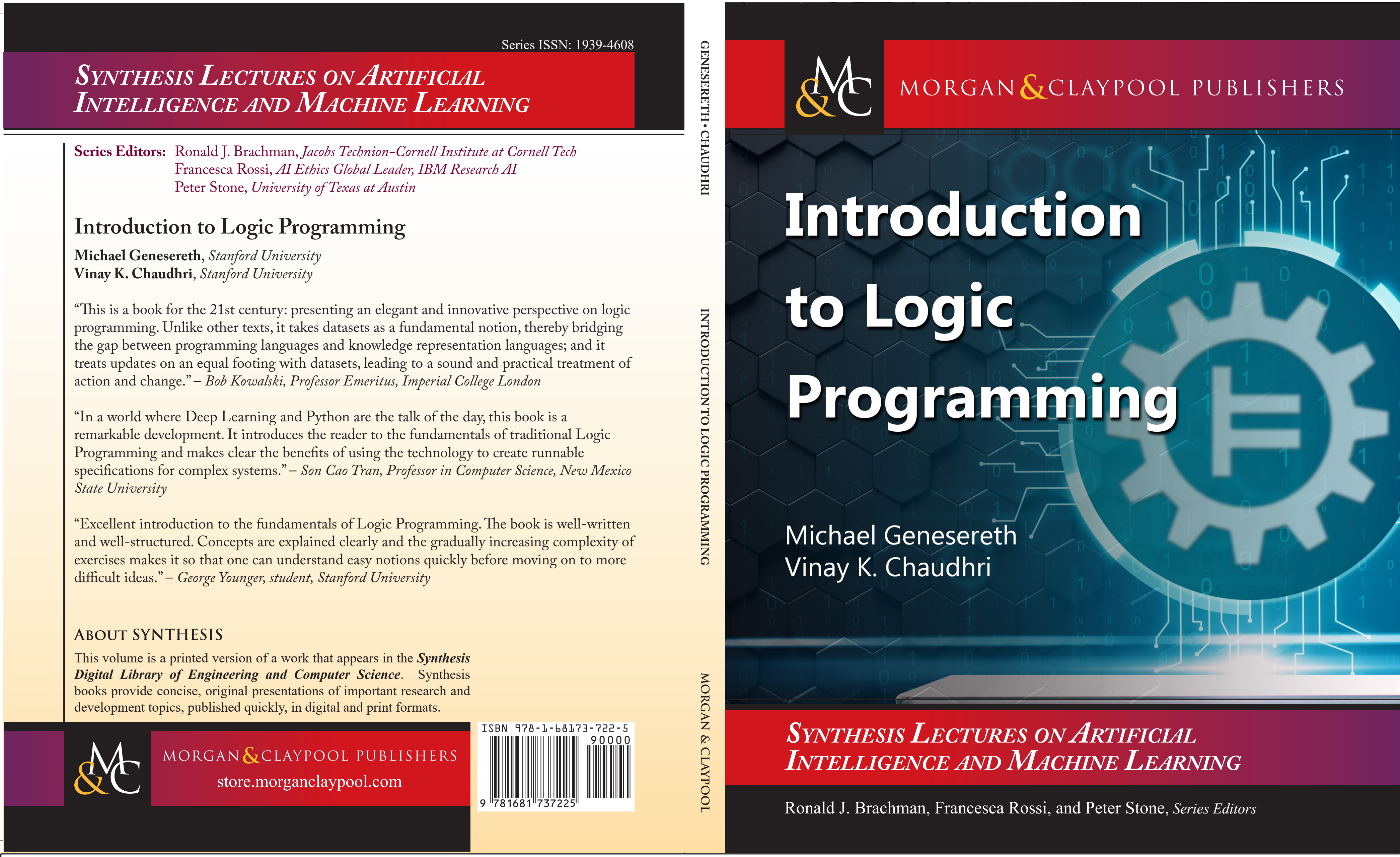
Satisfactory = 70% and above

Extra Credit

Added to score before determining Reported Grade

Discretionary

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
ISBN 978-1-68173-722-5
90000
9 781681 737225
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