EECS 344

Design of Computer Problem Solvers

Winter 2008

Overview

- Introductions
- What the course is about
- Syllabus
- Grading
- Resources

Introduction

- Ken Forbus
  email: forbus@northwestern.edu
- But for class business, please use eecs344-staff@cs.northwestern.edu, to ensure proper handling

What this course is about

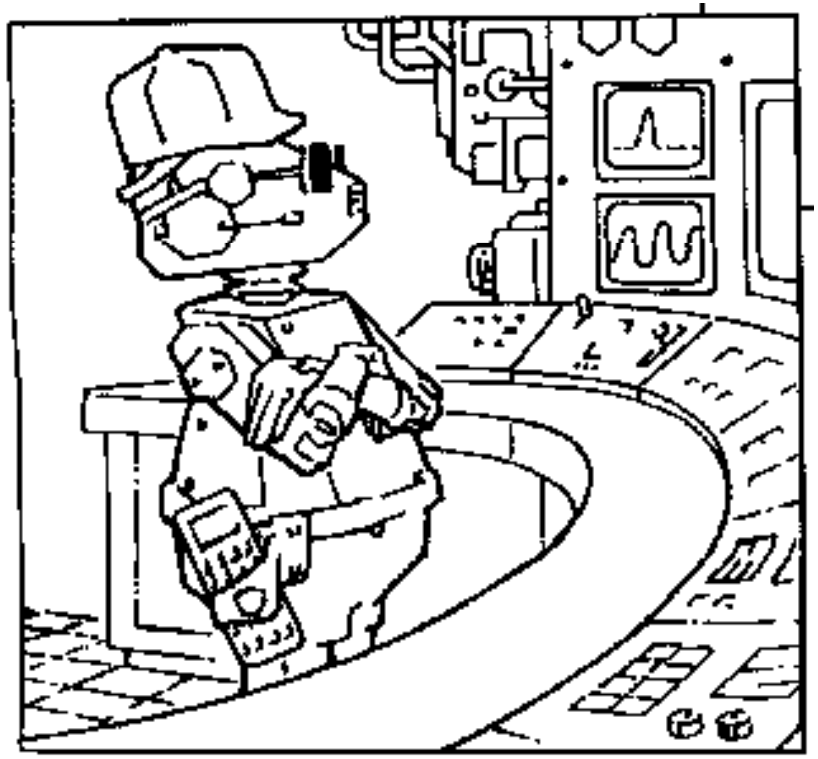
- How to build systems that reason
  - Solve problems, both simple and complex
  - Explain their results
- How to build well-engineered, efficient AI systems
  - Start simple, grow more complexity
  - Industrial-strength AI programming

How AI systems are different

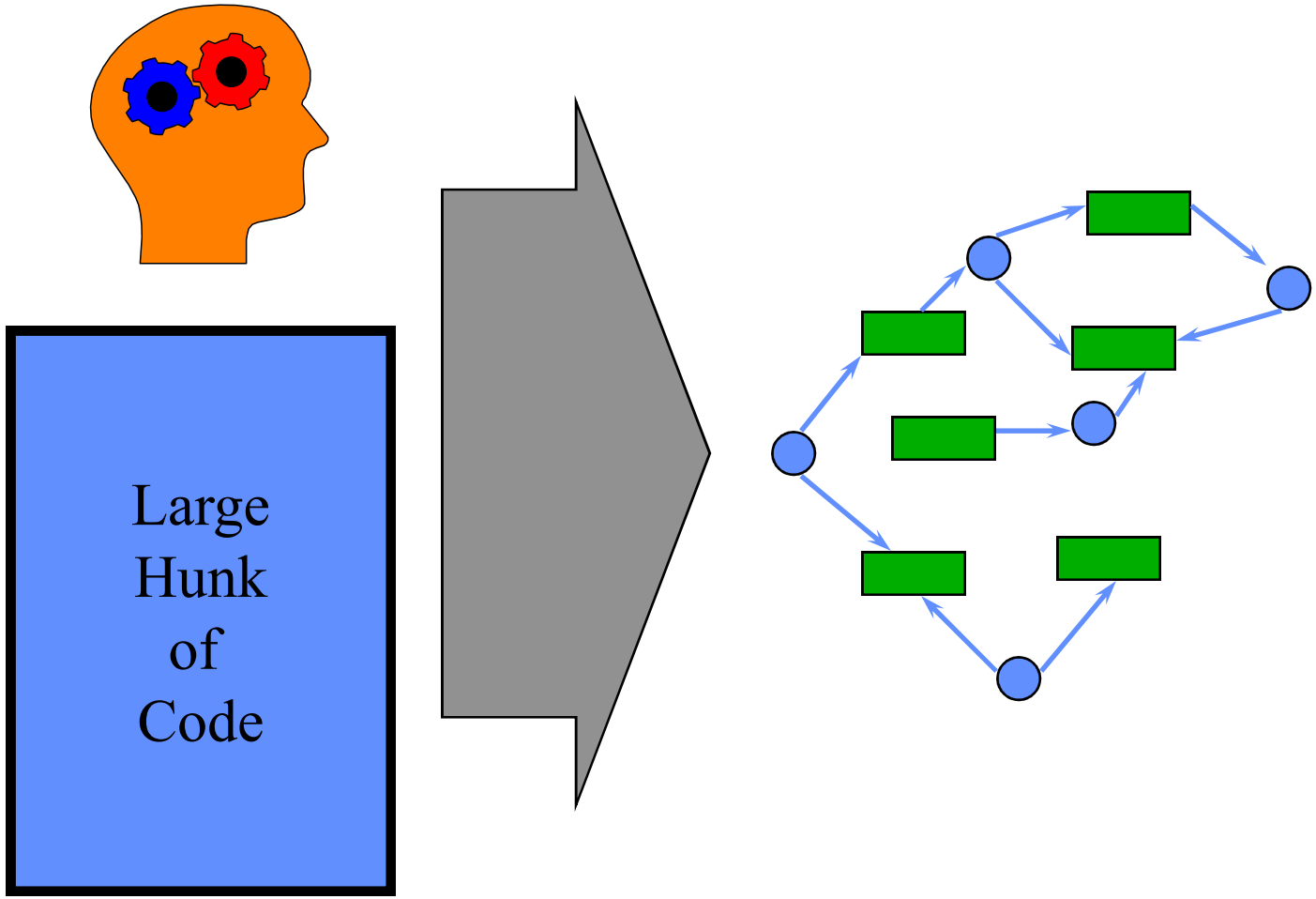
- Constraints all programs must satisfy:
  - Efficiency
  - Coherence
  - Flexibility
  - Additivity
  - Explicitness
- Which is more important?

Example: Traffic control programs

- Very efficient
- Handles routine situations well
- Doesn’t work well in unusual circumstances
- Hard to extend or adapt to new circumstances
- Hard to figure out why it does everything it does



The essence of creating reasoning systems



Tradeoffs

- Efficiency & Coherence improve together
  - Store only the minimum necessary to get job done (reduces explicitness)
  - Figure out optimal order of actions in advance (reduces flexibility)
  - Both of these reduce additivity because knowledge resides only in the programmer's head.

Tradeoffs

- Flexibility, Additivity, and Explicitness improve together
  - Explicit representation of knowledge allows program to figure out what to do (but with reduced coherence and efficiency).
  - Program can be extended by adding new knowledge rather than "mind surgery" (but with reduced coherence and efficiency).

Best solutions
to tradeoffs
depend on
what you are doing

Phases of AI programming

- *Conceptualization*
  - Figuring out what the problem really is
  - No programming here
- *Initial Exploration*
  - Trying out the idea to see if you are on track
  - Need rapid, throwaway prototype(s) to avoid wasting time on dead-ends
  - Can be large and complex, if built on pre-existing modules

Phases of AI Programming (2)

- *Experimentation*
  - Test the idea out on dozens to hundreds of examples
  - Need robust, efficient code, good interfaces
- *Production*
  - Building a fieldable module or system for non-experts to use
  - Need very robust, efficient code, excellent interfaces.

Programming Sins

- Optimizing the wrong program
- Not optimizing the right program
- Wasting time on fancy interfaces
- Not building time-saving interfaces

- Avoiding these problems is a matter of experience (and sometimes taste).

Building Problem Solvers

Kenneth D. Forbus and Johan de Kleer

Classical Problem Solving

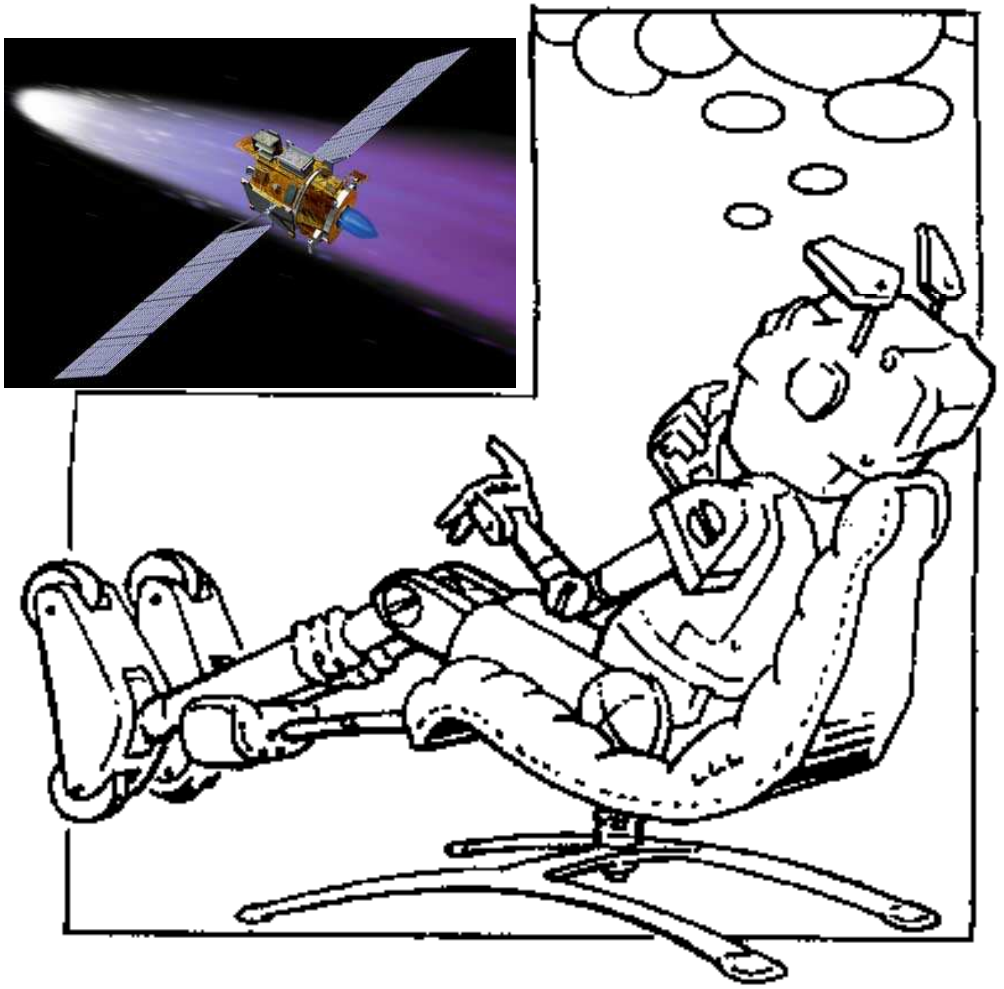
- Problem space model
- CPS – a simple but general search engine
- Examples: Subway pathfinding, solving algebraic equations

Pattern-Directed Rule Systems

- Antecedent reasoning
- Organizing programs as sets of rules
- Increasing power via manipulation of assumptions
- Increasing efficiency via open-coding of pattern matching and aligning program models
- Example: Natural Deduction

Truth-Maintenance Systems

- Major part of the course
- Great tool for efficient inference, generating explanations, and efficient search
- Will focus on major variations, ignoring offshoots



Truth Maintenance: Threat or Menace?

- "Total Information Awareness is one of those few ideas that justify the cliché "Orwellian". Just start with the name, go on to the technologies it touts ("biologically inspired algorithms for agent control", "truth maintenance")…"
  NYT Magazine, December 22, 2002
- Echos earlier New Yorker article (12/9/02) which mentioned "Orwellian" idea of "truth maintenance"
- Probably started with Safire op-ed piece (11/14/02), http://www.nytimes.com/2002/11/14/opinion/14SAFI.html

Total Information Awareness

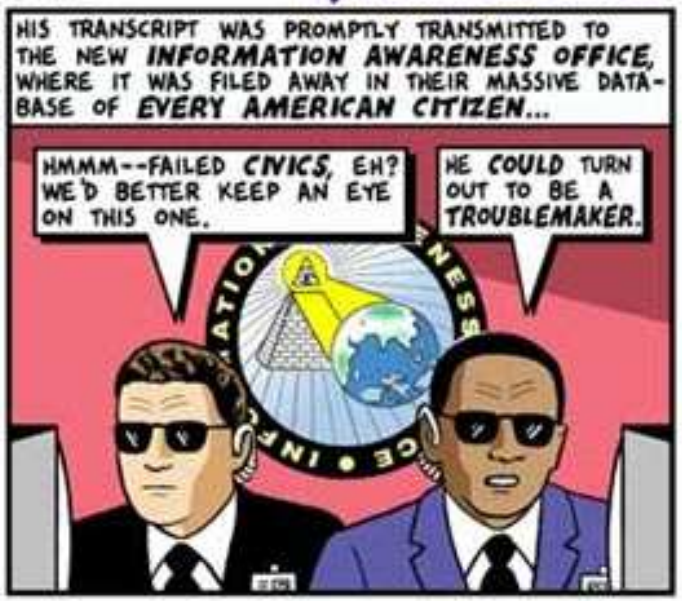
IAO Vision (http://www.darpa.mil/iao/)

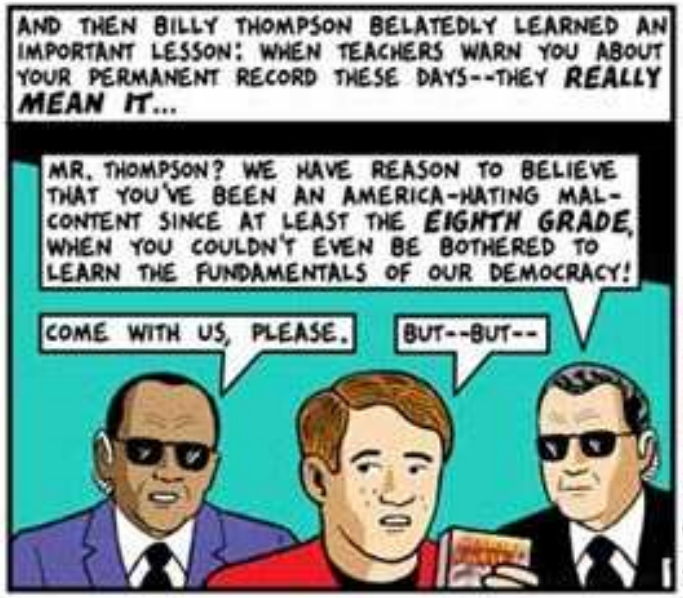
The most serious asymmetric threat facing the United States is terrorism, a threat characterized by collections of people loosely organized in shadowy networks that are difficult to identify and define. IAO plans to develop technology that will allow understanding of the intent of these networks, their plans, and potentially define opportunities for disrupting or eliminating the threats. To effectively and efficiently carry this out, we must promote sharing, collaborating and reasoning to convert nebulous data to knowledge and actionable options. IAO will accomplish this by pursuing the development of technologies, components, and applications to produce a proto-type system. Example technologies include:

- Collaboration and sharing over TCP/IP networks across agency boundaries
- Large, distributed repositories with dynamic schemas that can be changed interactively by users
- Foreign language machine translation and speech recognition
- Biometric signatures of humans
- Real time learning, pattern matching and anomalous pattern detection
- Entity extraction from natural language text
- Human network analysis and behavior model building engines
- Event prediction and capability development model building engines
- Structured argumentation and evidential reasoning
- Story telling, change detection, and truth maintenance
- Business rules sub-systems for access control and process management
- Biologically inspired algorithms for agent control
- Other aids for human cognition and human reasoning

THIS MODERN WORLD

by TOM TOMORROW



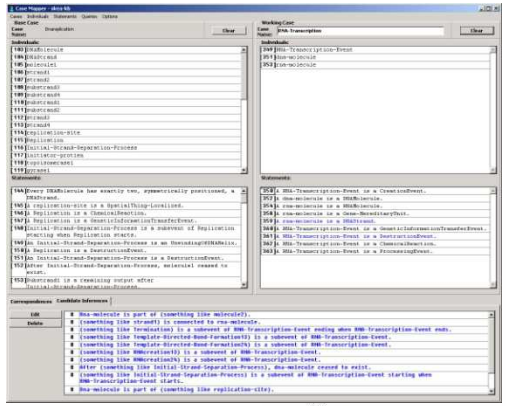


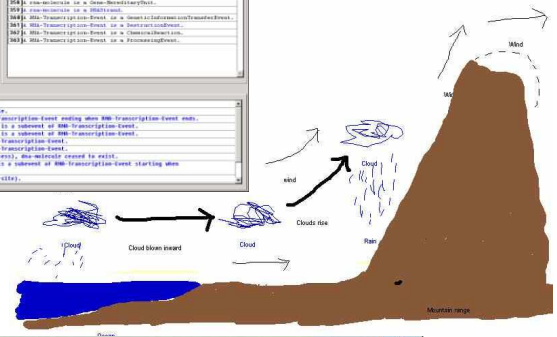
What's the reality?

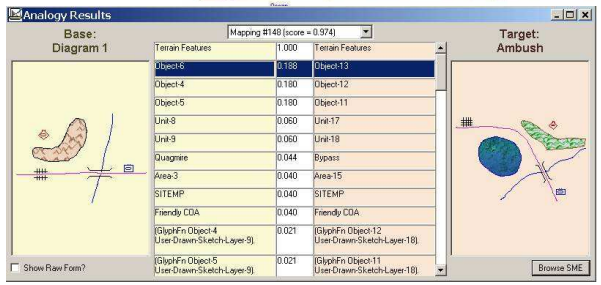
- Privacy would be enhanced if truth maintenance systems were used more widely
  - Example: Credit bureaus often don't track who put what into your file, or how reliable that source is
- Finding ways to catch terrorists while maintaining civil liberties is a political, legal, but also technical challenge
  - Some schemes under consideration would enable searches for patterns, without identifying individuals. Individuals would only be identified by getting a court order.
  - Technology, used wisely, could provide more privacy, yet with more ability to detect terrorists, than we have today

Large knowledge-based systems

- FIRE reasoning engine
  - KB = subset of Cyc
  - Logic-based TMS
  - Backchainer
  - Agenda-based solver
  - Federated architecture
- You'll use FIRE to build a problem solver
- You'll become familiar with its internals

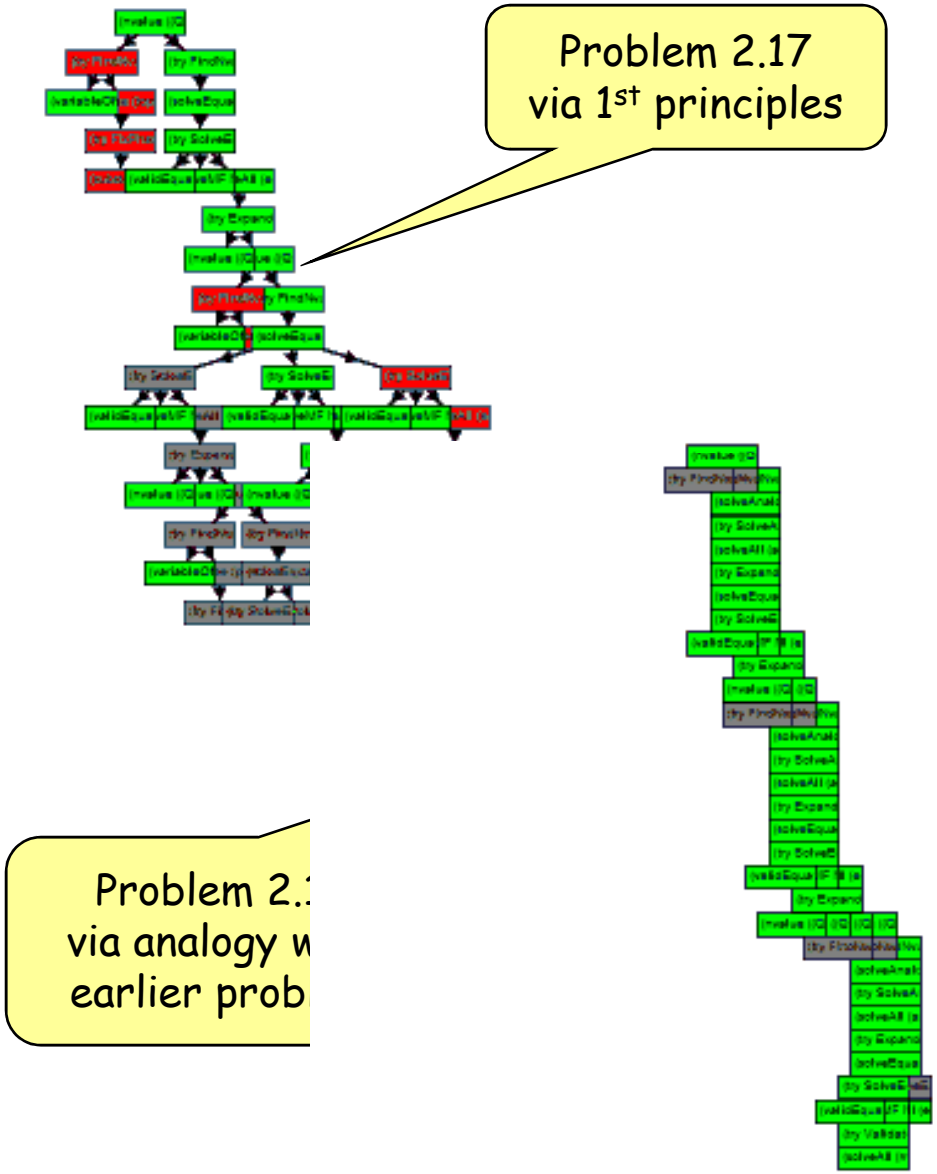






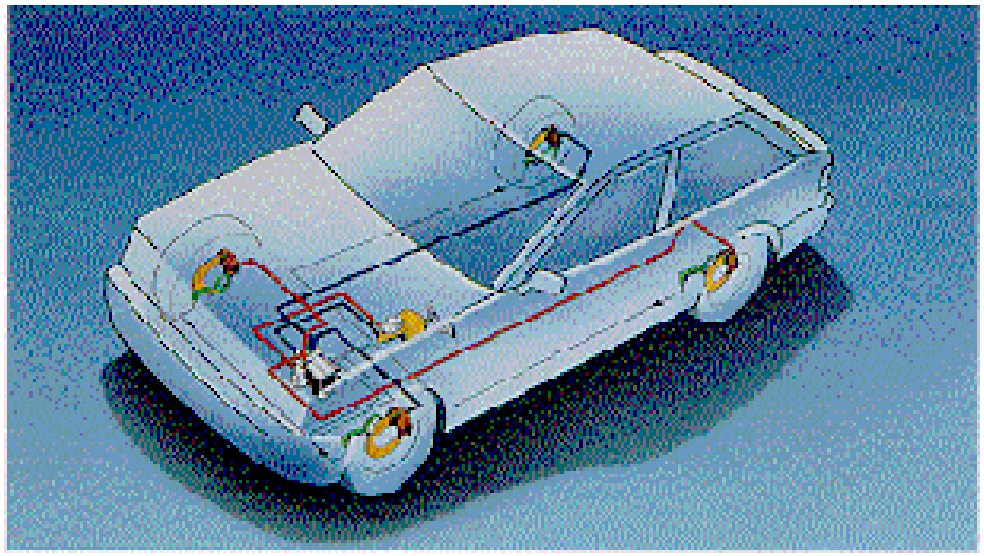
Analogical Reasoning

- Using models of human analogical processing
  - Matching, retrieval, generalization
- Integrating analogy smoothly with other forms of reasoning



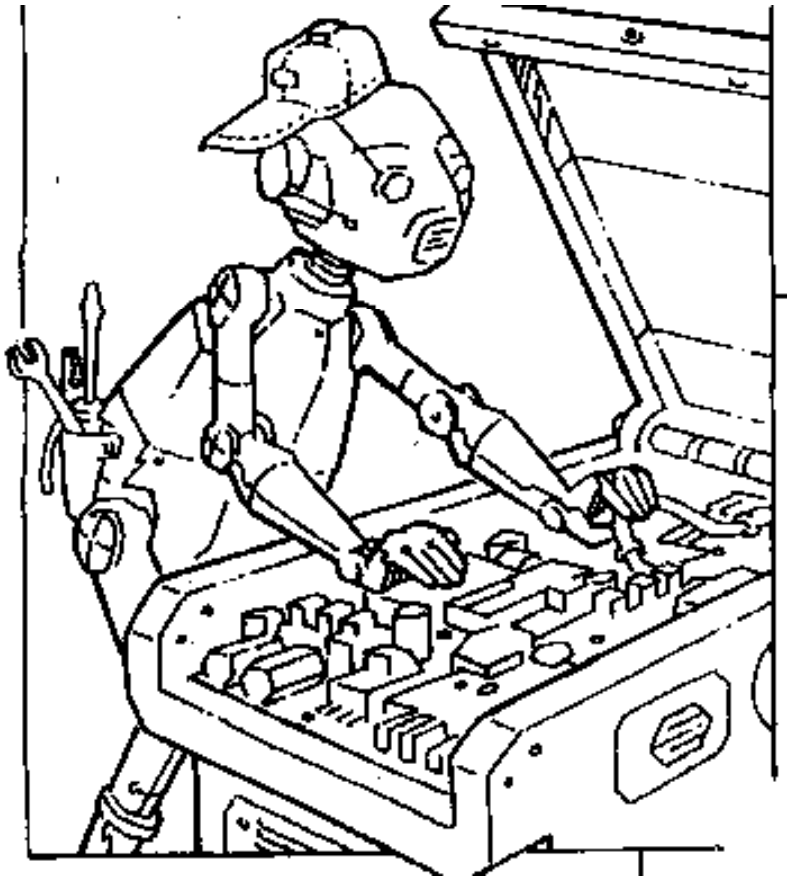
Assumption-based TMS

- Useful for diagnosis, planning
- Provides elegant conceptual model for many tasks
- Example: Planning



Constraint systems

- Symbolic relaxation
- Propagation methods
- Examples: Scene analysis, diagnosis



Grading

- Homework (60%)
  - Homeworks will involve programming.
  - Homeworks will involve writing short but subtle programs, or modifying existing programs
- Term project (30%)
  - Term project must involve using techniques from class on something you're interested in.
- Participation (10%)
  - Both in class and on-line

Homework procedures

- Turn in via email to
  eecs344-staff@cs.northwestern.edu
  - Softcopy only, no hardcopies will be accepted
  - Code as attachments.
  - Body of the email should describe the attachments and show results of running the code on test cases.
- Your code must work, i.e., execute correctly on reasonable test cases. Please don't turn in non-working code.
- Criteria includes style and quality of both your code and your explanations.

Collaboration

- Homework assignments and term projects must be done by individuals working alone, not in collaboration and not by groups.
  - You can discuss assignments in a general way with your peers, but you must do your own programming.
  - Turning in work that is not your own, or other violations of academic honesty, will be treated severely.

Resources

- Mailing list
  - eecs344-staff@cs.northwestern.edu for private questions
- Blackboard for class discussions
  - You are responsible for monitoring the newsgroup for class-related announcements and updates
- Web site: http://www.cs.northwestern.edu/~forbus/c44
  - All lectures notes and homeworks will be posted there
  - Code will be posted there
  - You are responsible for announcements, material on the web site

Course software

- BPS code available from class web site
  - Please use this version, not others
- All programming will be in Common Lisp
  - The workhorse language of artificial intelligence
  - Used today in a surprising number of enterprises
  - Good working knowledge of common lisp essential for AI wizardry

Access to Common Lisp

- Allegro Common Lisp for Windows or Linux, 8.1
  - Preferred version for class
  - Available in the T-Lab
  - Site license for on-campus use
    - See Blackboard
- Most of the code will run on any Common Lisp
  - I won't be able to support you in using them, however
  - Exception: FIRE uses AllegroCache.

Studio Instruction

- Ideally, we interleave lecture, discussion, and doing
- Who's got notebooks/tablets?

Brushing up on Common Lisp

- Highly recommended books
  - Steele, Guy. *Common Lisp: The Language*. 2nd Ed.
  - Norvig, Peter. *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp.*
- Various on-line tutorials available
  - http://www.alu.org/table/learn.htm
  - http://www.cs.berkeley.edu/~russell/ai.html
  - http://www.norvig.com/luv-slides.ps

Assignment

- Read BPS Chapters 1-3