AI at 50: From Programs to Solvers
Models and Techniques for General Intelligence

Héctor Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, Spain

Goals

- Address some changes that have taken place in **AI research** in last 20 years

  ▷ Changes that have to do with move from **programs** to **solvers**

  $$\textit{Problem} \Longrightarrow \boxed{\textit{Solver}} \Longrightarrow \textit{Solution}$$

  ▷ Solvers are **general programs** whose scope defined in terms of a **model**

  ▷ Challenge is computational: how to make the solvers **scale up**

- Articulate this research agenda that has emerged in last 10-20 years

- Explain its relevance to the old AI goals concerning **general intelligence** and **human cognition**

Motivation

- It is often assumed that no much has happened in AI Research since the 80's in relation to the grand old goals of AI

  - ▷ Marvin Minsky, a founding father, declares *AI to be brain-dead since the 70's*, Wired Magazine, Nov 8 2003

  - ▷ In *Complex Cognition*, by R. Sternberg and T. Ben-Zeev, Oxford Univ. 2001, AI chapter features 5 programs: Logic Theorist (1956), GPS (1958), Eliza (1963), Mycin (1975), Dendral (1976)

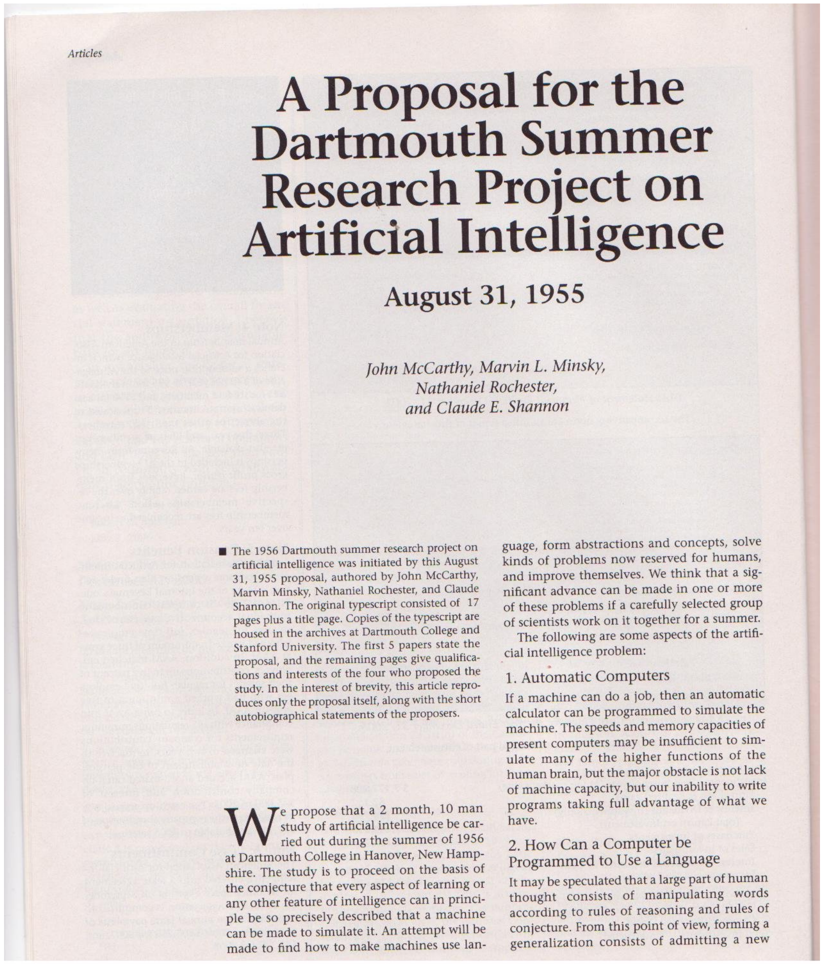
  - ▷ Many of the debates surrounding AI (GOFAI, Situated AI, Symbolic vs. Non-Symbolic) date back to the 80's and not revised since

- I aim to show that this impression is wrong

Outline

- Some AI history
- The problem of generality in AI
- Models and Solvers:
  - ▷ Graphical Models: SAT, CSPs, Bayesian Networks
  - ▷ Planning Models: Strips
  - ▷ Planning with Feedback: MDPs and POMDPs
- Lessons learned and why they matter
- Summary

Darmouth 1956



Articles

A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence

August 31, 1955

John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon

■ The 1956 Dartmouth summer research project on artificial intelligence was initiated by this August 31, 1955 proposal, authored by John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon. The original typescript consisted of 17 pages plus a title page. Copies of the typescript are housed in the archives at Dartmouth College and Stanford University. The first 5 papers state the proposal, and the remaining pages give qualifications and interests of the four who proposed the study. In the interest of brevity, this article reproduces only the proposal itself, along with the short autobiographical statements of the proposers.

We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

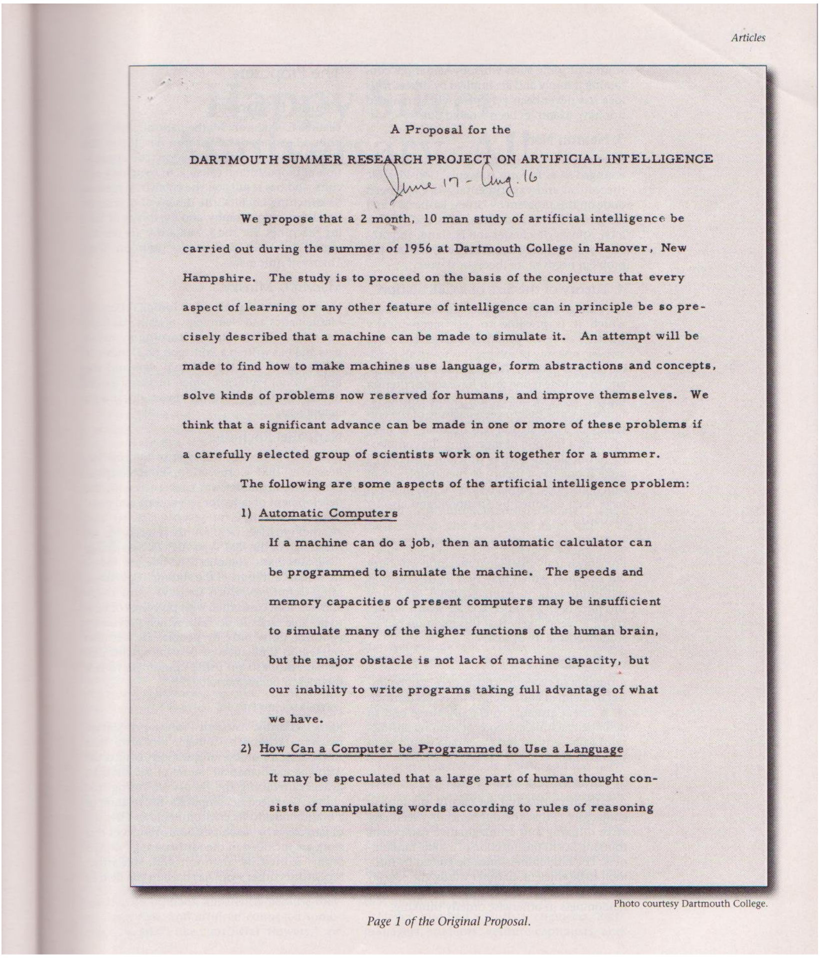
The following are some aspects of the artificial intelligence problem:

1. Automatic Computers

If a machine can do a job, then an automatic calculator can be programmed to simulate the machine. The speeds and memory capacities of present computers may be insufficient to simulate many of the higher functions of the human brain, but the major obstacle is not lack of machine capacity, but our inability to write programs taking full advantage of what we have.

2. How Can a Computer be Programmed to Use a Language

It may be speculated that a large part of human thought consists of manipulating words according to rules of reasoning and rules of conjecture. From this point of view, forming a generalization consists of admitting a new



Articles

A Proposal for the

DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE

June 17 - Aug. 16

We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

The following are some aspects of the artificial intelligence problem:

1) Automatic Computers

If a machine can do a job, then an automatic calculator can be programmed to simulate the machine. The speeds and memory capacities of present computers may be insufficient to simulate many of the higher functions of the human brain, but the major obstacle is not lack of machine capacity, but our inability to write programs taking full advantage of what we have.

2) How Can a Computer be Programmed to Use a Language

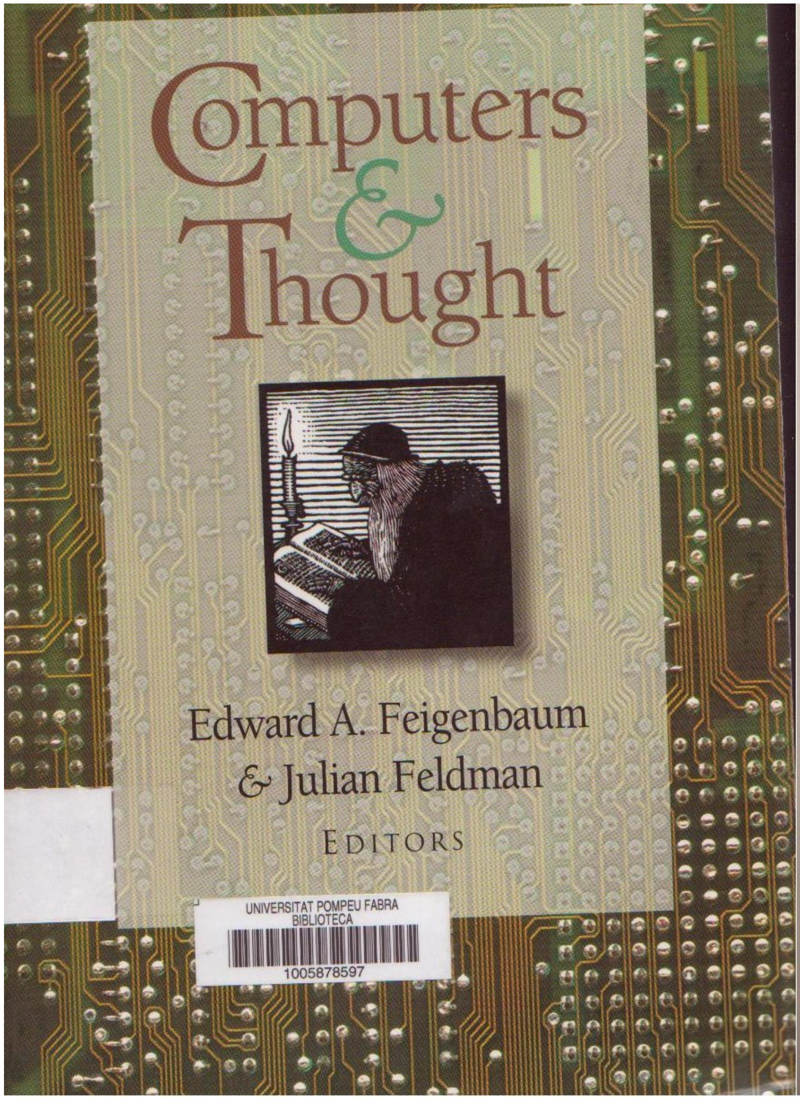
It may be speculated that a large part of human thought consists of manipulating words according to rules of reasoning

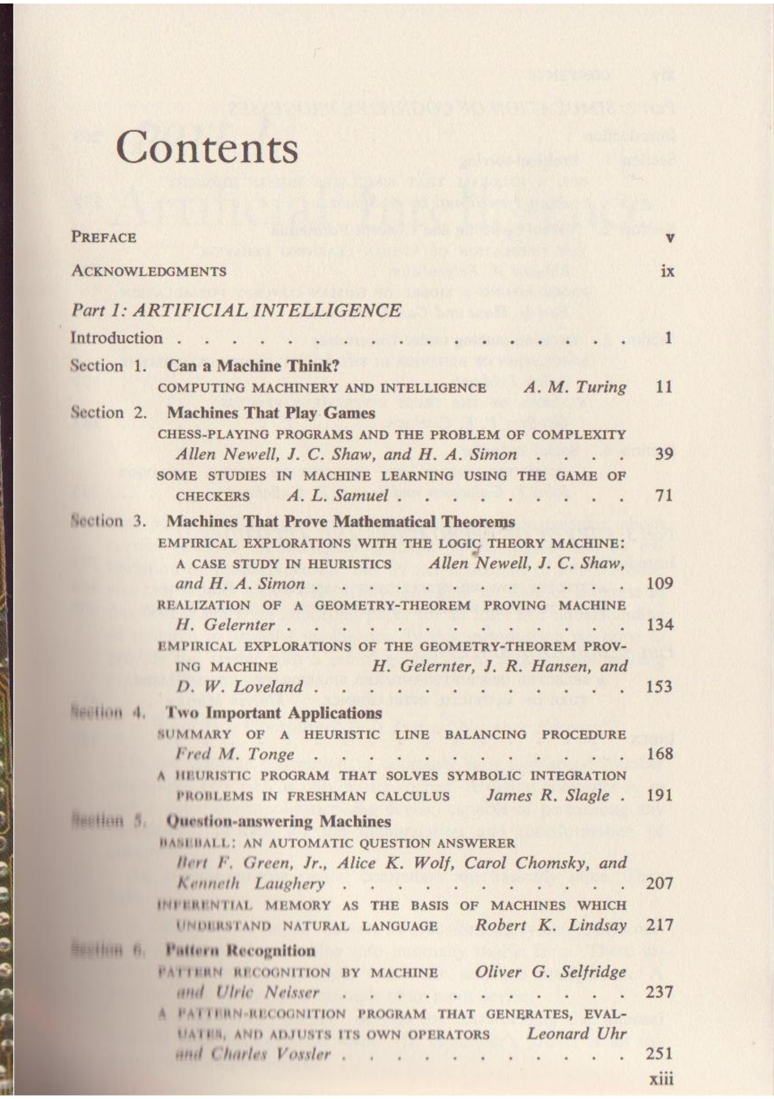
Photo courtesy Dartmouth College.

Page 1 of the Original Proposal.

"The proposal (for the meeting) is to proceed on the basis of the conjecture that every aspect of ... intelligence can in principle be so precisely described that a machine can be made to simulate it"

Computers and Thought 1963





Contents

xiii

An early collection of AI papers and programs for playing chess and checkers, proving theorems in logic and geometry, planning, etc.

Importance of Programs in Early AI Work

In preface of 1963 edition of *Computers and Thought*

We have tried to focus on papers that report results. *In this collection, the papers . . . describe actual working computer programs . . . Because of the limited space, we chose to avoid the more speculative . . . pieces.*

In preface of 1995 AAAI edition

A critical selection criterion was that the paper had to describe . . . a running computer program . . . All else was talk, philosophy not science . . . (L)ittle has come out of the "talk".

AI, Programming, and AI Programming

Many of the key AI contributions in 60's, 70's, and early 80's had to to with **programming** and the **representation of knowledge** in **programs**:

- Lisp (Functional Programming)
- Prolog (Logic Programming)
- Rule-based Programming
- Interactive Programming Environments and Lisp Machines
- Frame, Scripts, Semantic Networks
- 'Expert Systems' Shells and Architectures

AI methodology: Theories as Programs

- For writing an AI dissertation in the 60's, 70's and 80's, it was common to:
  - ▷ pick up a task and domain X
  - ▷ analyze/introspect/find out how task is solved
  - ▷ capture this reasoning in a program

- The dissertation was then
  - ▷ a **theory** about X (scientific discovery, circuit analysis, computational humor, story understanding, etc), and
  - ▷ a **program** implementing the theory, **tested** over a few examples.

Many great ideas came out of this work . . . but there was a problem . . .

Methodological problem:

Theories expressed as programs cannot be proved wrong: when a program fails, it can always be blamed on 'missing knowledge'

Three approaches to this problem

- narrow the domain (expert systems)
  - ▷ problem: lack of generality
- accept the program is just an illustration, a demo
  - ▷ problem: limited scientific value
- fill up the missing knowledge (intuition, commonsense)
  - ▷ problem: not successful so far

AI in the 80's

The knowledge-based approach reached an **impasse** in the 80's, a time also of debates and controversies:

- **Good Old Fashioned AI** is "rule application" but intelligence is not (Haugeland)
- **Situated AI:** representation not needed and gets in the way (Brooks)
- **Neural Networks:** inference needed is not logical but probabilistic (PDP Group)

Many of these criticisms of mainstream AI at least partially valid then.

How valid are they now?

AI Research in 2007

Recent issues of AIJ, JAIR, AAAI or IJCAI shows papers on:

1. **SAT and Constraints**
2. **Search and Planning**
3. **Probabilistic Reasoning**
4. **Probabilistic Planning**
5. Inference in First-Order Logic
6. Machine Learning
7. Natural Language
8. Vision and Robotics
9. Multi-Agent Systems

I'll focus on 1–4: these areas often deemed about **techniques**, but more accurate to regard them as **models** and **solvers**.

Example: Solver for Linear Equations

$$\textit{Problem} \Longrightarrow \boxed{\textit{Solver}} \Longrightarrow \textit{Solution}$$

- **Problem:** The age of John is 3 times the age of Peter. In 10 years, it will be only 2 times. How old are John and Peter?
- **Expressed as:** $J = 3P \;\; ; \;\; J + 10 = 2(P + 10)$
- **Solver:** Gauss-Jordan (Variable Elimination)
- **Solution:** $P = 10 \;\; ; \;\; J = 30$

Solver is **general** as deals with any problem expressed as an instance of **model**

Linear Equations Model, however, is **tractable**, AI models are not . . .

AI Solvers

$$\textit{Problem} \Longrightarrow \boxed{\textit{Solver}} \Longrightarrow \textit{Solution}$$

- The basic models and task we will consider are
  - ▷ **Constraint Satisfaction/SAT**: find state that satisfies constraints
  - ▷ **Bayesian Networks:** find probability over variable given observations
  - ▷ **Planning Problems:** find actions that map given state into a final state
  - ▷ **Planning with Feedback:** find strategy for mapping state into final state

- All of these models are **intractable**, and some extremely powerful (POMDPs)
- The challenge is computational: **how to scale up**
- For this, solvers must **recognize and exploit structure** of the problems
- Methodology is **empirical**: benchmarks and competitions
- Significant progress in recent years

SAT and CSPs

- **SAT** is the problem of determining whether there is a **truth assignment** that satisfies a set of clauses

$$x \vee \neg y \vee z \vee \neg w \vee \cdots$$

- Problem is NP-Complete, which in practice means worst-case behavior of SAT algorithms is **exponential** in number of variables ($2^{100} = 10^{30}$)
- Yet current SAT solvers manage to solve problems with **thousands of variables and clauses**, and used widely (circuit design, verification, planning, etc)
- **Constraint Satisfaction Problems (CSPs)** generalize SAT by accommodating non-boolean variables as well, and constraints that are not clauses

How SAT solvers manage to do it?

Two types of **efficient (poly-time) inference** in every node of the search tree:

- **Unit Resolution:**
  - ▷ *Derive clause C from $C \vee L$ and* **unit clause** $\sim L$
- **Conflict-based Learning and Backtracking:**
  - ▷ *When empty clause $\Box$ derived, find 'causes' S of $\Box$, add $\neg S$ to theory, and backtrack til S disabled*

Other ideas are **logically possible** but **do not work** (do not scale up):

- Generate and test each one of the possible assignments **(pure search)**
- Apply resolution without the unit restriction **(pure inference)**

Related tasks: Enumeration and Optimization SAT Problems

- **Weighted MAX-SAT**: find assignment σ that minimizes total cost $w(C)$ of violated clauses

$$\sum_{C:\sigma \not\models C} w(C)$$

- **Weighted Model Counting**: Adds up 'weights' of satisfying assignments:

$$\sum_{\sigma:\sigma \models T} \prod_{L \in \sigma} w(L)$$

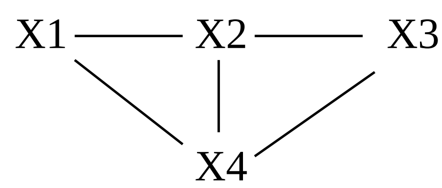
SAT methods extended to these other tasks, closely connected to **probabilistic** reasoning tasks over **Bayesian Networks**:

- **Most Probable Explanation (MPE)** easily cast as Weighted MAX-SAT
- **Probability Assessment** $P(X|Obs)$ easily cast as Weighted Model Counting

Current best BN solvers built over this formulation (ACE, Weighted Cachet)

SAT, CSPs, and BNs: Complexity and Treewidth

- The underlying structure of SAT, CSPs, and BNets can be expressed by graph G



- A parameter called the **(induced) treewidth** $w(G)$ measures then how 'close' is G to a Tree, $w(G) = 2$ for G above, and $w(G) = 1$ if G is a tree.

- All SAT, CSP, and BN tasks are **exponential** in $w(G)$, and thus solvable in **polynomial time** for **bounded** $w(G)$ (e.g., trees)

- These models often referred to as **graphical models** (Dechter 03)

From Graphical Models to Planning Models

- **Planning** concerned with finding a **sequence of actions** that transforms an **initial state** into a **goal state**. This is called a **plan**

- **States** are truth assignments as before, represented by the atoms that are true

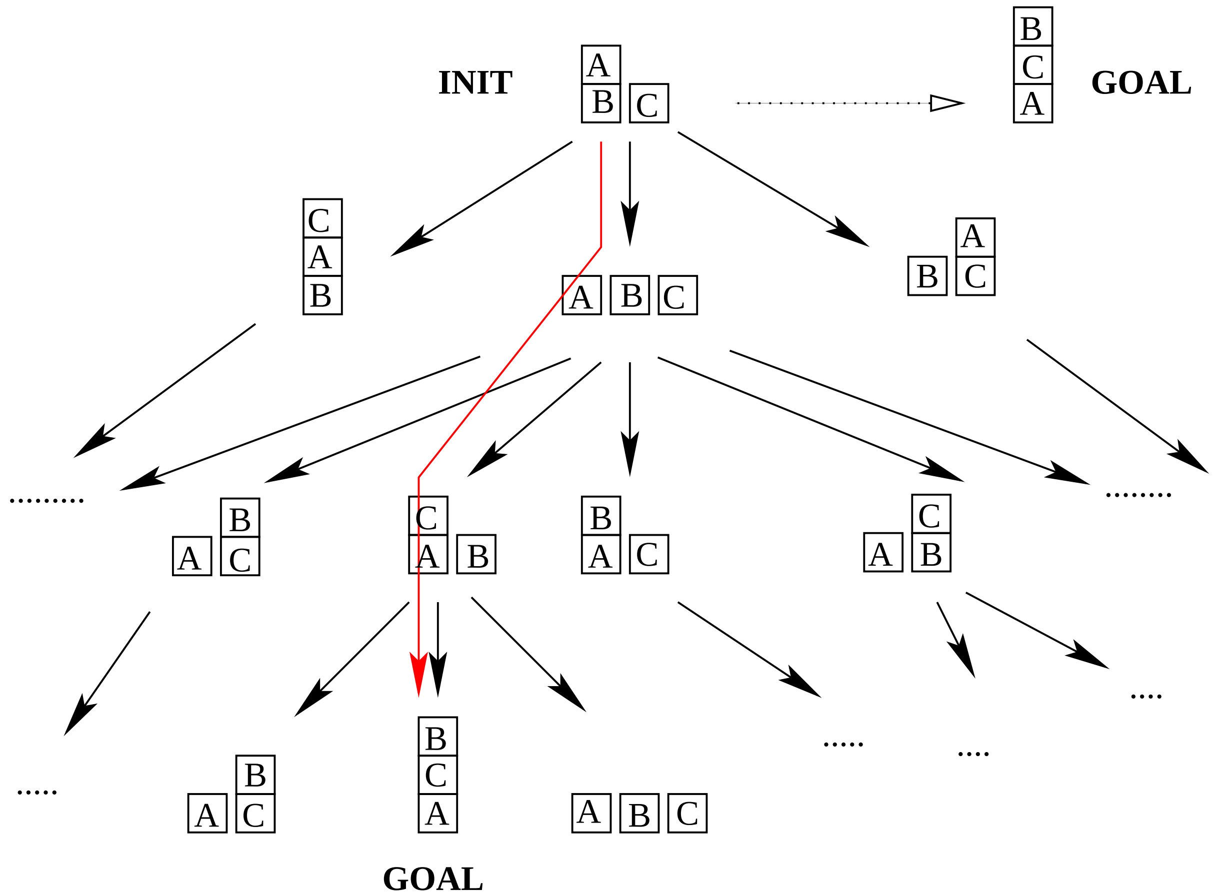
- Actions **add** certain atoms and **delete** others, provided their **preconditions** hold

- A **planner** is a solver that takes a **planning problem** (initial and goal states, and actions) and outputs a **plan**

- The **cost** of a plan given by the **number of actions**

$$\textit{Init, Actions, Goals} \Longrightarrow \boxed{\textit{Planner}} \Longrightarrow \textit{Plan}$$

Example

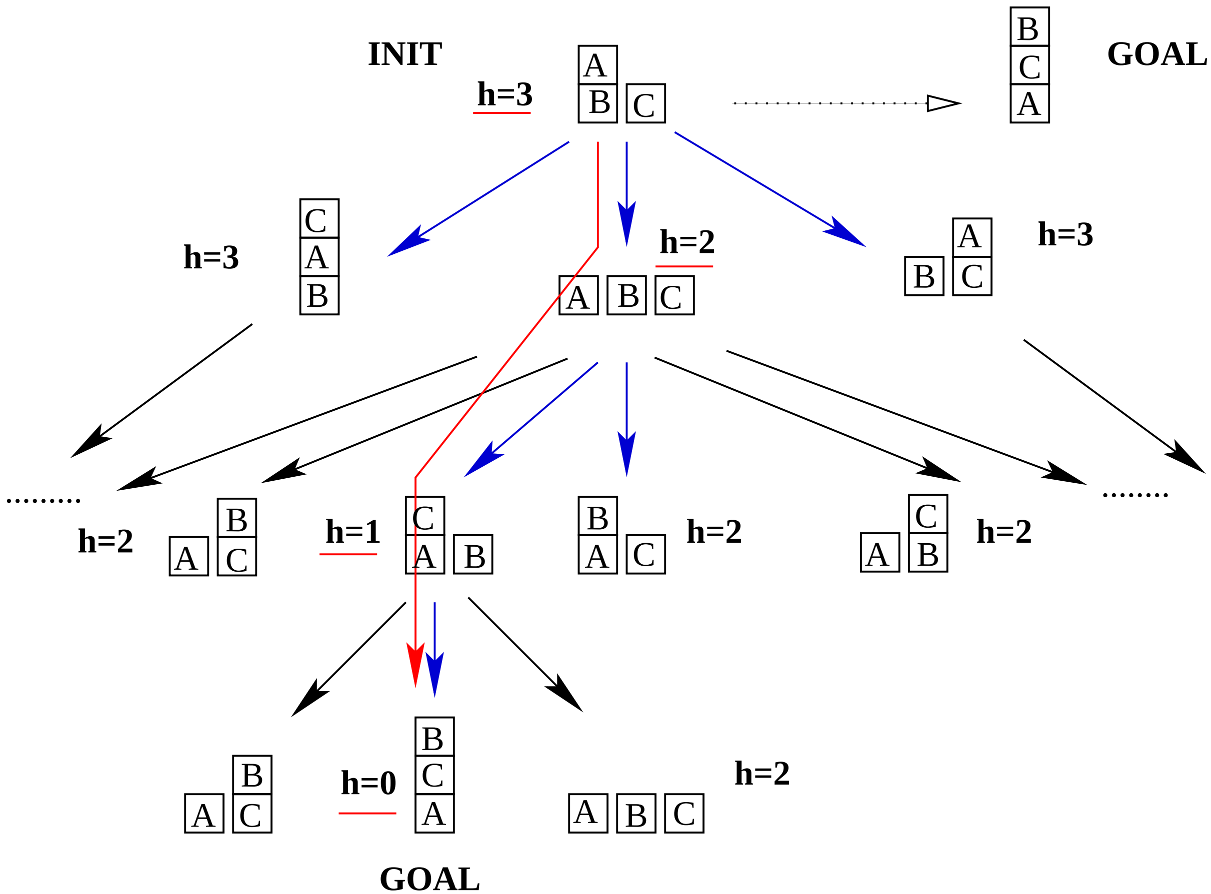


- Given the **actions** that move a 'clear' block to the table or onto a another 'clear' block, **find a plan** to achieve the goal
- Problem becomes **finding a path** in a **directed graph**

How planning problems are solved?

- How do we find a route in a map from Barcelona to Madrid?
- Need **sense of direction:** whether an action takes us towards the goal or not
- In AI, this is captured by **heuristic functions**: functions $h(s)$ that provide an **estimate of the cost** (number of actions) from any state s to the goal
- Key new idea in planning is that useful heuristics $h(s)$ can be obtained **automatically** from the problem encoding (Bonet and Geffner 01)
- **How?** Solving a **relaxed problem** where **deletes** are dropped
- Heuristic $h(s)$ is cost of solution found for **relaxed problem** in **poly-time**

How is our problem is then solved?



- Provided with the **heuristic** h, plan found without search by **hill-climbing**
- Actually, only the states reached by the actions in blue evaluated (planner FF; Hoffmann and Nebel 2001)

The appraisals $h(s)$ from a cognitive point of view

- they are **opaque** and thus cannot be **conscious**

  meaning of symbols in the relaxation is not the normal meaning; e.g., objects can be at many places at the same time as old locations not deleted

- they are **fast and frugal** (linear-time), but unlike the 'fast and frugal heuristics' of Gigerenzer et al. are **general**

  they apply to all problems fitting the model (planning problems)

- they play the role of **'gut feelings'** or **'emotions'** according to De Sousa 87, Damasio 94, Evans 2002, Gigerenzer 2007

  providing a guide to action while avoiding infinite regresses in the decision process

Planning with Feedback and Situated AI

- In the presence of **uncertainty** and **feedback**, a solution to a planning problem is not a **fixed action sequence** but an **action strategy**

- We consider briefly two models that extend the 'classical planning' model above

  ▷ one in which the actions have **uncertain effects** and the state of the system is **full observable** (MDPs)
  ▷ one in which the actions have **uncertain effects** and the state is only **partially observable** (POMDPs)

- The motivation is twofold:

  ▷ show that some models are very expressive
  ▷ show relation to **Situated AI** (Brooks 1990)

MDPs and POMDPs

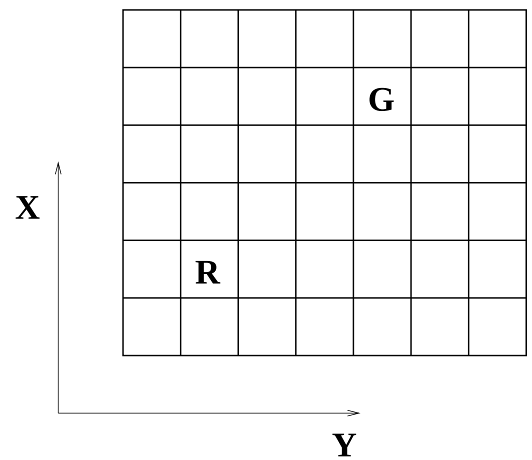
- Fully and Partially Observable MDPs are like 'classical planning' models except that:
  - ▷ **uncertain effects:** after doing a in s, prob. of moving to s' is $P_a(s'|s)$
  - ▷ **sensor model:** after a in s, observe o with prob $P_a(o|s)$ (in MDPs $o = s$)
- Solutions are **closed-loop policies** mapping **(belief) states** into actions
- Optimal solutions minimize **expected cost to goal**
- MDPs and POMDPs solvers compute such solutions from model.

Actions
Sensors $\longrightarrow$ $\boxed{\textit{MDP / POMDP Solver}}$ $\longrightarrow$ *Plan** $\underset{observations}{\overset{actions}{\rightleftarrows}}$ *World*
Goals

Representation of Policies

- MDP solutions are **functions** $\pi : S \mapsto A$ mapping states into actions
- No commitment about **representation** of this function needed; e.g., if goal is to reach (x_G, y_G) in **empty greedy**, and robot loc. is (X, Y), policy π can be

$$\begin{array}{lcl} X < X_G & \rightarrow & MoveLeft \\ X > X_G & \rightarrow & MoveRight \\ Y < Y_G & \rightarrow & MoveUp \\ Y > Y_G & \rightarrow & MoveDown \end{array}$$



- Moreover, **stimulus-action** rules can represent solutions to **sets of problems** as well (e.g., all Blocks)
  - ▷ **if** $on(X, Y)$ *in Goal,* Y *well-placed and clear, and* X *is clear*
    then *MOVE* X *onto* Y
  - ▷ . . .

Executing Policies vs Finding the Policies

- **Solutions** to planning problems (Strips, MDPs, POMDPs,. . .) can be **expressed** as **stimulus-action rules** (and many other ways)

- Yet, **expressing solutions** (to Strips, MDPs, . . . , ..) is different than **finding solutions**

- In **Situated-AI** (Brooks 90), it is assumed that the **programmer solves the problem** in his/her head, and the robot **just executes this solution**

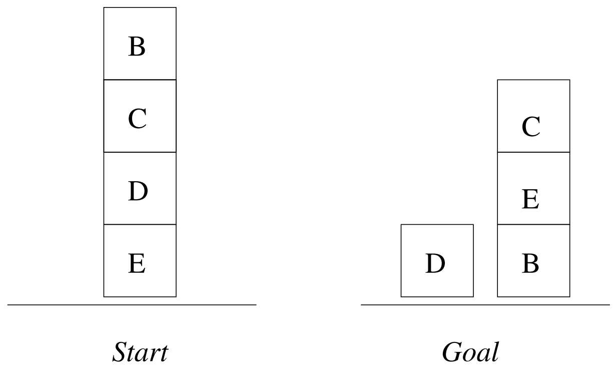
- In **Model-based AI**, solutions computed by solver. Is this necessary?
  - ▷ in lower animals, solutions **hardwired** by evolution
  - ▷ in humans, evolution and culture have produced **solution methods** as well

- **Learning** is the third approach to get solutions, different than **Model-based Solver** or **Programming/Evolution**

Summary

- A **research agenda** that has emerged in last 20 years: **solvers** for a range of **intractable models**
- **Solvers** unlike other programs are **general** as they do not target individual problems but families of problems (**models**)
- The challenge is **computational** and the methodology **empirical**
- Consistent **progress:**
  - ▷ efficient but effective inference methods (derivation of h, conflict-learning)
  - ▷ islands of tractability (treewidth methods and relaxations)
  - ▷ transformations (compiling away incomplete info, extended goals, . . .)
- While the agenda is technical, resulting ideas likely to be **relevant** for understanding general intelligence and human cognition

General Solvers and People



How people solve this problem?

- Blocks looks too easy; psychologists preferred puzzles like Tower of Hanoi. Yet
  - ▷ Blocks is **not trivial for a general solver**, and indeed
  - ▷ Language and Perception easy for people but not **computationally**
- Are these problems solved using 'domain-knowledge'?
  - ▷ **not clear:** easy to introduce variations where knowledge does not apply
  - ▷ **not necessary:** can be solved provided 'right inferences' are captured

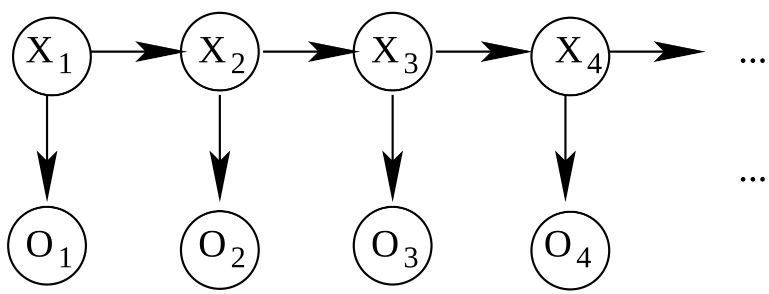
Bayesian Networks

A Bayesian Network (Pearl 1988) is a compact representation of a joint probability distribution over a set of variables $X_1, \ldots, X_n$ made up of:

- a DAG where the nodes are the variables $X_1, \ldots, X_n$
- Conditional probability tables $prob(X_i|pa(X_i))$, $i = 1, \ldots, n$, where $pa(X_i)$ refers to the parents of X_i in the DAG

The DAG implicitly defines a set of **independences** that result in joint distrib.

$$P(X_1, \ldots, X_n) = \prod_{i=1,n} P(X_i|pa(X_i))$$



A **Hidden Markov Model** is a **Bayesian Tree** solvable in **linear-time.**

Lessons

- Solvers must know how to search by performing **efficient but effective inference** during the search
- If they do it well enough, they may do **little or no search:** planner CPT (Vidal and Geffner 2005) solves hundreds of benchmarks **backtrack-free**
- Powerful inference methods **developed** and **tested** in last 20 years for range of models include:

  boolean-constraint propagation, conflict-learning, treewidth methods, heuristic estimations based on relaxations, . . .

- Quest for **general solvers able to scale up** can be a useful source of models for **cognition**