

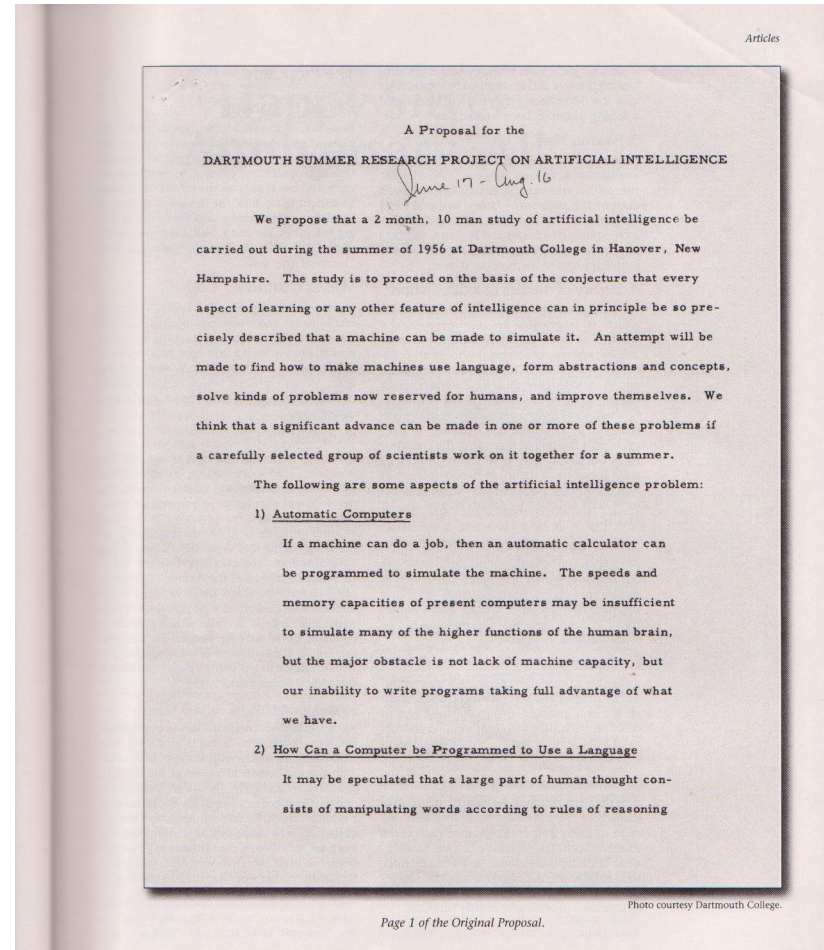
The Model-based Approach to Autonomous Behavior: Prospects and Challenges

Hector Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, Spain
12/2010

Plan for the Talk

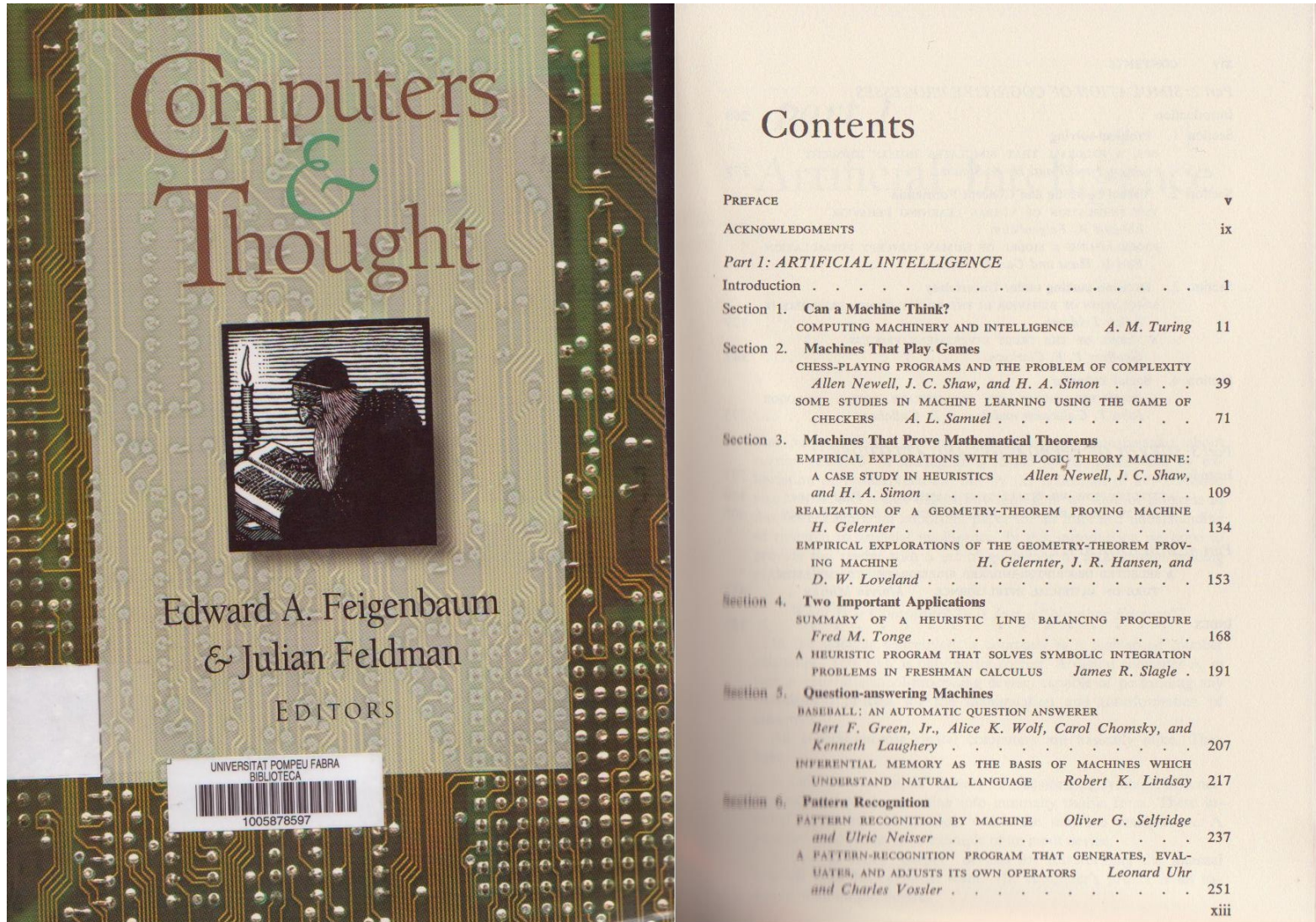
- Artificial Intelligence
 - ▷ *brief AI history*
 - ▷ *AI models and solvers*
- Planning
 - ▷ *what is planning?*
 - ▷ *what has been achieved?*
 - ▷ *heuristics and transformations*
- Wrap up
 - ▷ *challenges and opportunities*

Dartmouth 1956



“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



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

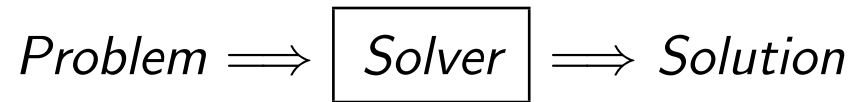
A critical selection criterion was that the paper had to describe . . . a running computer program . . . All else was talk, philosophy not science . . . (L)ittle has come out of the “talk”.

AI, Programming, and AI Programming

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

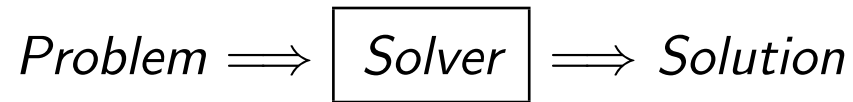
- Lisp (Functional Programming)
- Prolog (Logic Programming)
- Rule-based Programming
- Interactive Programming Environments and Lisp Machines
- Frame, Scripts, Semantic Networks
- 'Expert Systems' Shells and Architectures
-

From Programs to Solvers



- **Solvers** are programs that target a specific class of **models**
 - ▷ **Constraint Satisfaction/SAT**: find state that satisfies constraints
 - ▷ **Bayesian Networks**: find probability over variable given observations
 - ▷ **Planning**: find action sequence or policy that produces desired state
 - ▷ **General Game Playing**: find best strategy in presence of n -actors
 - ▷ . . .
- Solvers for these models are **general**; not tailored to specific instances
- Models are all **intractable**, and some extremely powerful (POMDPs)
- Challenge in all cases is **computational: how to scale up**
- Methodology is **empirical**: benchmarks and competitions

From Programs to Solvers



- **Solvers** are programs that target a specific class of **models**
 - ▷ **Constraint Satisfaction/SAT**: find state that satisfies constraints
 - ▷ **Bayesian Networks**: find probability over variable given observations
 - ▷ **Planning**: find action sequence or policy that produces desired state
 - ▷ **General Game Playing**: find best strategy in presence of n -actors
 - ▷ . . .
- Solvers for these models are **general**; not tailored to specific instances
- Models are all **intractable**, and some extremely powerful (POMDPs)
- Challenge in all cases is **computational: how to scale up**
- Methodology is **empirical**: benchmarks and competitions
- Crisp **validation**; significant **progress** in recent years

Planners

A **planner** is a **solver over a class of models**; it takes a model description, and automatically computes its solution, which is a **controller**



- Models encode **initial situation, actions, sensors, and goal**
- Many different planning **models**: uncertainty, feedback, costs, ...
- Many types of solutions forms (**controllers**) according to type of feedback

Basic State Model for (Classical) AI Planning

- finite and discrete state space S
- a **known initial state** $s_0 \in S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a **deterministic state transition function** $s' = f(a, s)$ for $a \in A(s)$
- action costs $c(a, s) > 0$

A **solution** is a sequence of applicable actions that maps s_0 into S_G

It is **optimal** if it minimizes sum of action costs (e.g., # of steps)

The resulting controller is **open-loop** (no feedback)

Uncertainty but No Feedback: Conformant Planning

- finite and discrete state space S
- a **set of possible initial state** $S_0 \in S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a **non-deterministic** transition function $F(a, s) \subseteq S$ for $a \in A(s)$
- action costs $c(a, s)$

Uncertainty but no sensing; hence controller still **open-loop**

A **solution** is an **action sequence** that achieves the goal in spite of the uncertainty; i.e. for **any possible initial state** and **any possible transition**

Planning with Sensing

- finite and discrete state space S
- a **set of possible initial state** $S_0 \in S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a **non-deterministic** transition function $F(a, s) \subseteq S$ for $a \in A(s)$
- action costs $c(a, s)$

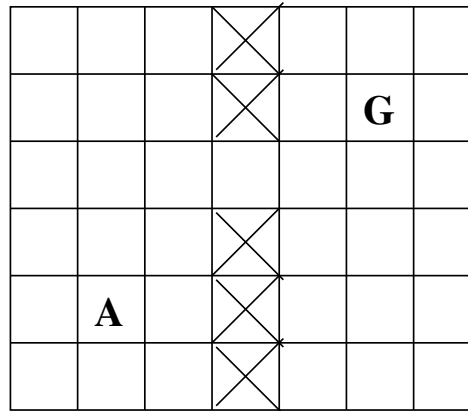
- a set O of **observation** tokens
- a **sensor model** $O(s)$ mapping states into observation tokens

Solutions can be expressed in many forms; e.g., **policies** mapping belief states into actions, contingent **trees**, finite-state **controllers**, etc.

Probabilistic version of this model known as **POMDP**: Partially Observable Markov Decision Process

Example

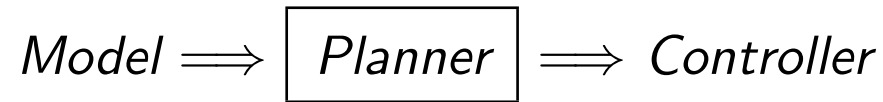
Agent **A** must reach **G**, moving deterministically, one cell at a time, in **known** map



- If **A** knows its location, planning problem is **classical**
- If **A** doesn't know its location, problem is **conformant**
- If **A** doesn't know door location but can sense it, it's **planning with sensing**

Different combinations of uncertainty and feedback: three problems, three models

Model Representation and Planning Languages



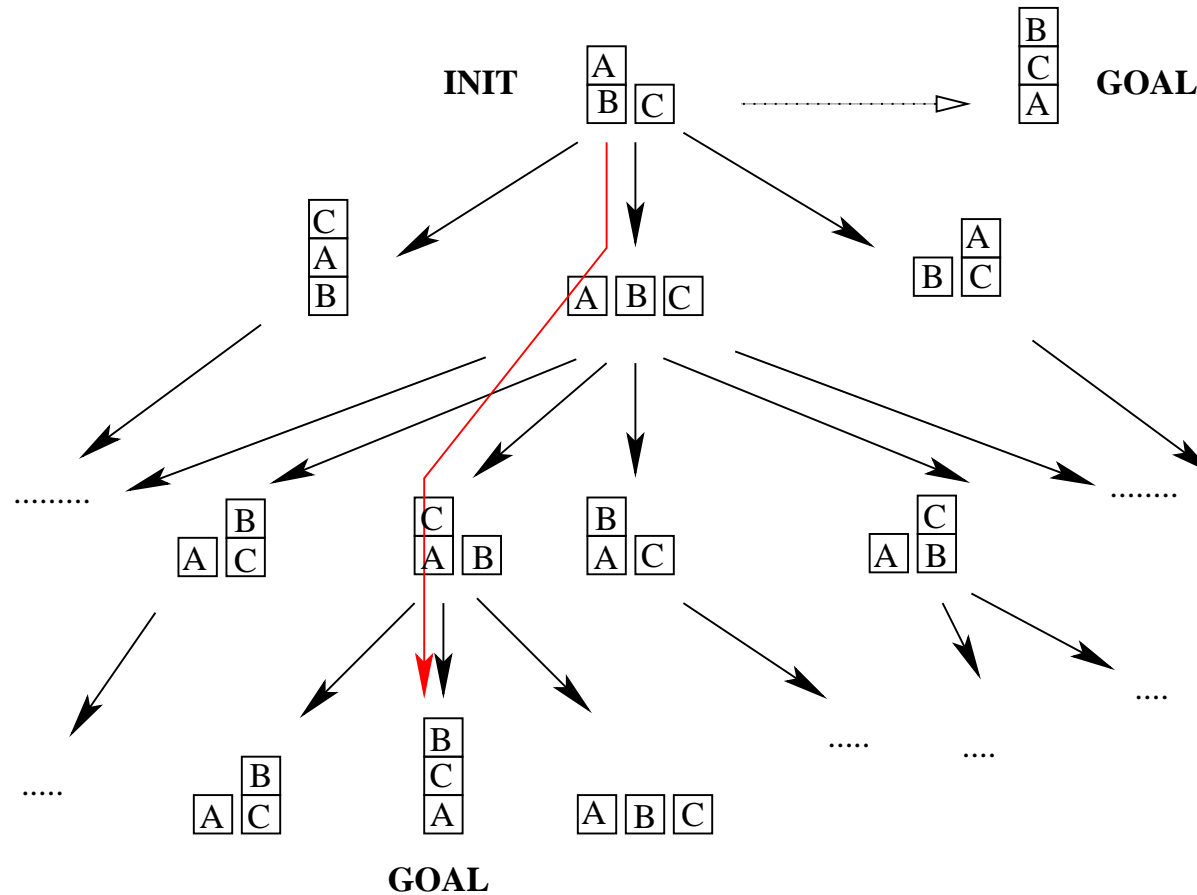
- Planning **models** described in compact form in terms of **variables**
- The **states** are the assignments of **values** to the **variables**
- The **action** effects change the states **locally**:
 - ▷ **adding** values that become true, and
 - ▷ **deleting** values that become false
- **Languages** like **PDDL** and **PPDDL** standard in planning competitions

Status AI Planning

- **Classical Planning works pretty well**
 - ▷ *Large problems solved very fast*
 - ▷ *New idea: automatic derivation and use of **heuristics***
- **Model simple but useful**
 - ▷ *Operators not primitive; can be policies themselves*
 - ▷ *Fast closed-loop replanning able to cope with uncertainty sometimes*
- **Beyond Classical Planning:** incomplete information, uncertainty, . . .
 - ▷ *Top-down approach: general **native solvers** for MDPs, POMDPs, etc.*
 - ▷ *Bottom-up approach: **transformations** and use of classical planners*

I'll focus on two techniques: **heuristics for classical planning**, and **transformations for soft goals, plan recognition, and finite-state controllers**

Classical Planning Example

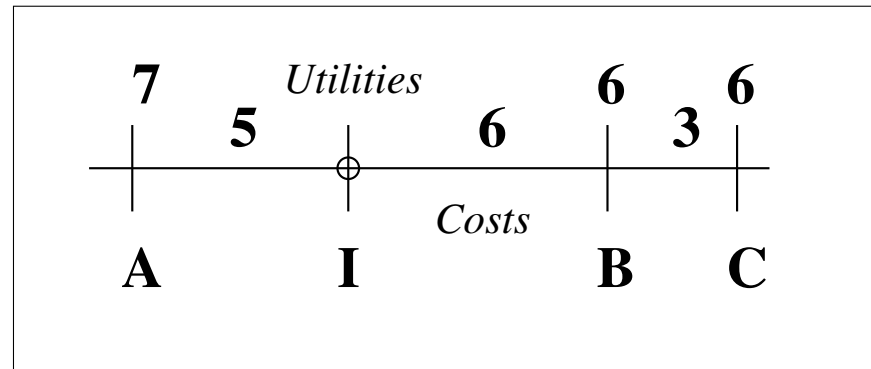


- Classical problem: move blocks to transform **Init** into **Goal**
- Problem becomes **finding a path** in a **directed graph**
- Difficulty is that graph is **exponential** in number of blocks . . .

How are heuristics defined and derived?

- Heuristics $h(s)$ defined in terms of solution cost of **simplified problem**
- Most common simplification in planning is **delete relaxation**
- In this relaxation, **new value** of variable **doesn't delete old value**
- These relaxed problems are **easy to solve; low poly-time**
- Use of **relaxations** for informing search is old idea in AI (Minsky 63; Pearl 83)
- Yet made it into **domain-independent planning** in last decade (Bonet & G. 99, Hoffmann 01, Helmert 04)
- Other approaches: SAT (Kautz & Selman), LPG (Gerevini, Serina, Saetti), . . .

Transformations: Soft Goals (Rewards)



- **Soft goals** as opposed to **hard goals** are to be achieved if worth the costs
- **Utility of plan** is **utility of soft goals achieved** minus **plan cost**:

$$u(\pi) = \sum_{\pi \models p} u(p) \quad - \quad \sum_{a \in \pi} c(a)$$

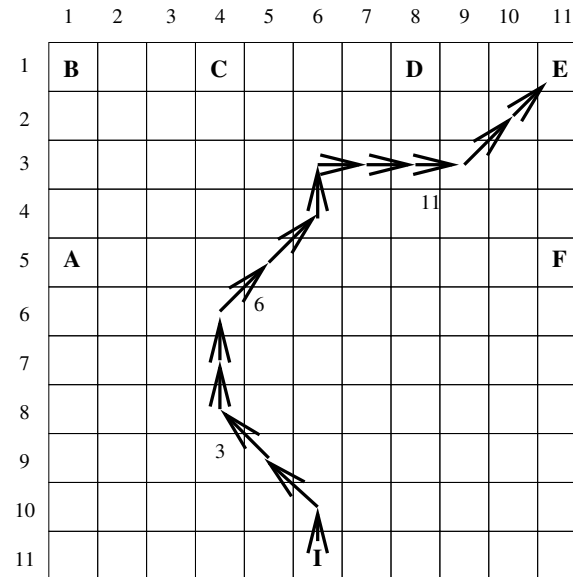
- **Best plan** above is to go right to *B* and *C* (from *I*)
- 2008 Int. Planning Competition featured **soft goal planning** track (net-benefit)

Soft Goal Planning with a Classical Planner

Yet soft goals can be easily **compiled away** (Keyder & G. 07,09)

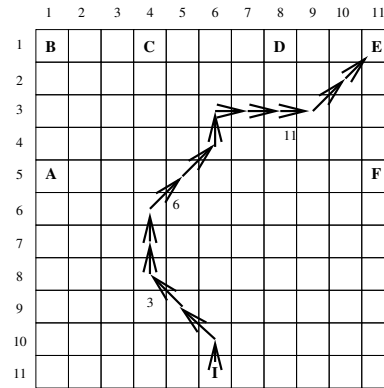
- For each soft goal p , create **new hard goal** p' initially false, and **two new actions**:
 - ▷ $collect(p)$ with precondition p , effect p' and **cost** 0, and
 - ▷ $forgo(p)$ with an empty precondition, effect p' and **cost** $u(p)$
- Plans π **maximize** $u(\pi)$ iff **minimize** $c(\pi) = \sum_{a \in \pi} c(a)$ **in translation**
- Classical planners over translation beat native net-benefit planners in competition

Transformations: Plan Recognition



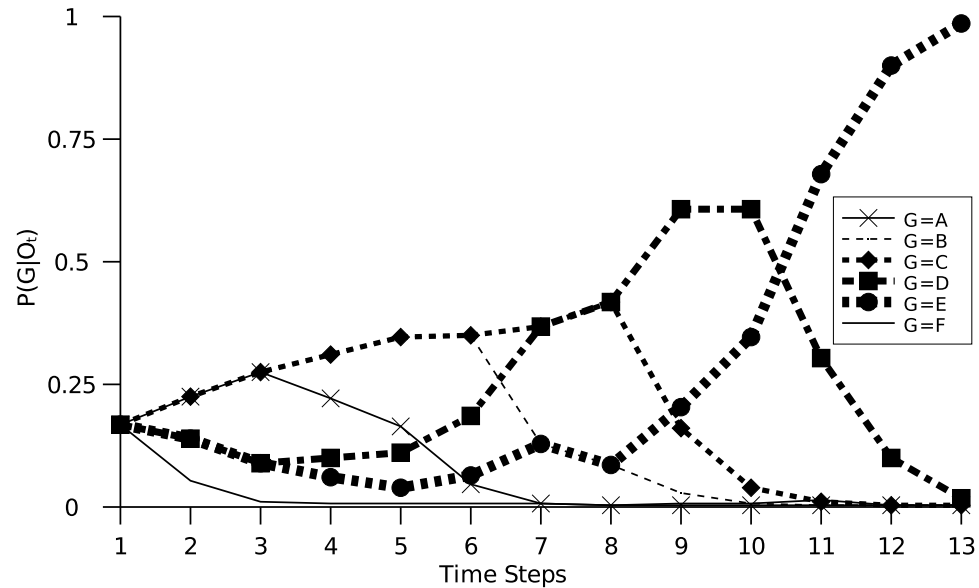
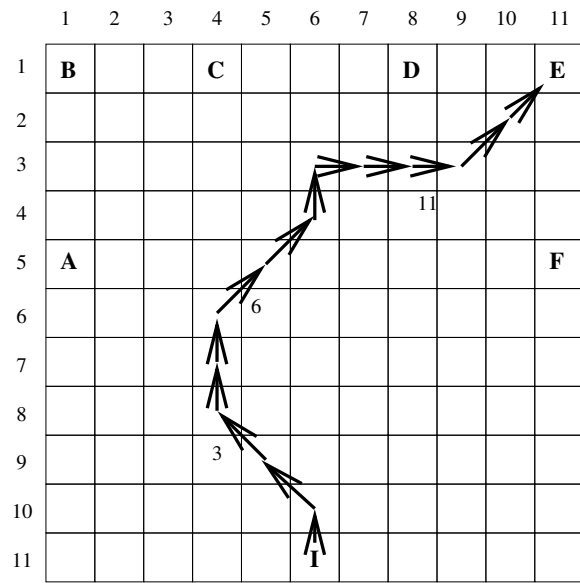
- Plan recognition is planning 'in reverse'
- Task is to find goal G from partially observed action sequence O
- Agent is moving from I to one of the goals A, B, C, D, \dots
- How to predict where is he going?

Inferring Goal Probabilities from Plan Costs



- Why initially A more likely than F ; i.e. $P(A|O) > P(F|O)$?
- Bayes rule says $P(G|O) = \alpha P(O|G)P(G)$ for possible goals G
- **Claim:**
 - ▷ $P(O|G)$ decreases monotonically with **difference** $c(G, \bar{O}) - c(G, O)$, where $c(G, O)/c(G, \bar{O})$ are **costs** of achieving G while complying/not complying with O
 - ▷ These costs can be computed with a **classical planner** from suitable **translation**

Probabilistic Plan Recognition using a Classical Planner



- **Goal posterior probabilities** $P(G|O)$ obtained from Bayes Rule and **likelihoods**

$$P(O|G) = \text{sigmoid}\{\beta [c(G, \bar{O}) - c(G, O)]\}$$

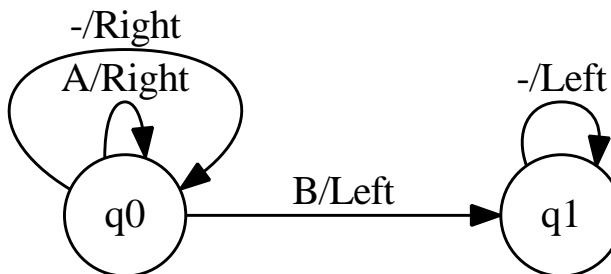
- Costs $c(G, O)$ and $c(G, \bar{O})$ computed with a **classical planner** over **translation**
- **Sigmoid function** follows from assuming probability of plan decays exponentially with cost (soft rationality assumption; Ramirez and G. 2009, 2010)

Transformations: Finite State Controllers

- Starting in A , move to B and back to A ; marks A and B **observable**.



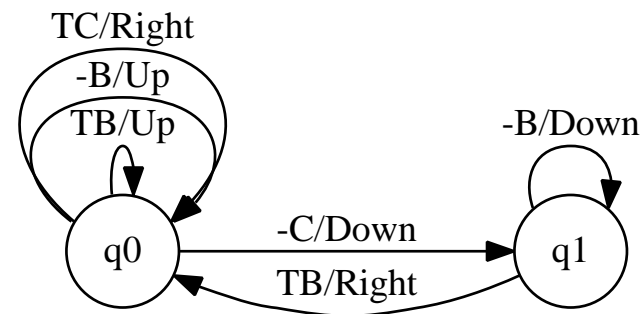
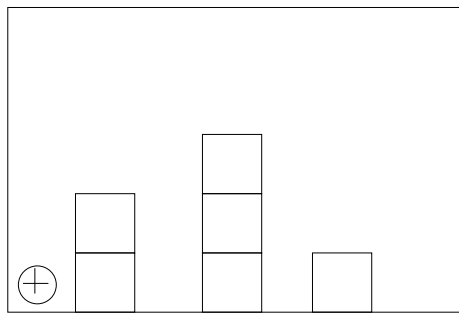
- This **finite-state controller** solves the problem



- FSC is **compact** and **general**: can add noise, vary distance, etc.
- Heavily **used in practice**, e.g. video-games and robotics, but **written by hand**

Derivation of Finite State Controllers Using Planners

- Finite state controllers (FSCs) derived using two **transformations**:
 - ▷ Planning problem \times Possible FSCs \longrightarrow conformant problem
 - ▷ Conformant problem \longrightarrow classical problem



- **Example:** move 'eye' (circle) one cell at a time til **green block** found
- **Observables:** Whether cell 'seen' contains a green block (G), non-green block (B), or neither (C); and whether on table (T) or not (-)
- Derived controller is **compact** and **general**, and applies to **any number of blocks** and **any configuration** (Bonet, Palacios, G. 2009)

Wrap up . . .

Planning and Autonomous Behavior

- In **planning**, autonomous behavior **derived** automatically from **model**
- Two other approaches to **autonomous behavior** are:
 - ▷ **programming-based**: behavior is **hardwired** by programmer
 - ▷ **learning-based**: behavior is **learned** from experience
- The **three approaches** compatible and even complementary (e.g., model-based reinforcement learning)
- Elements like **heuristic** or **value functions** often appear in them all, e.g.
 - ▷ Evaluation functions **hardwired** in Chess
 - ▷ Valuations functions **learned** from experience in reinforcement learning
 - ▷ Heuristic functions **derived** from relaxed models in planning

Challenges and Opportunities

- **Technical Challenges**

- ▷ **Scaling up** further: classical, MDP, POMDP, multi-agent planning
- ▷ **Hierarchies**: what to abstract away and when

- **Applications**

- ▷ in settings where **autonomy** required; e.g. aerospace
- ▷ **video-games** as future 'killer application' of planning?

- **Lessons for Cognitive Science**

- ▷ simple tasks are hard for **general solver** if structure not exploited
- ▷ automatic derivation of **heuristics** can provide model for generation of quick but global **appraisals**

Summary

- Shift in AI since 80's from programs to **solvers**
- Scope of solvers defined by **mathematical models**
- All these models **intractable**; the challenge is to **scale up**
- **Planning models** come in many forms: uncertainty, feedback, costs, . . .
- Key technique in **classical planning** is automatic derivation and use of **heuristics**
- Power of classical planners used for other tasks via **transformations**
- **Promise** of planning research is a solid **model-based** methodology for **autonomous agent design and analysis**

Unconscious Inference

- We have learned a lot about **effective inference mechanisms** in last 20 years from work on **domain-independent** solvers (SAT, Planning, BNets, etc)
- The problem of AI in the 80's (the 'knowledge-based' approach), was probably lack of **mechanisms** and not only **knowledge**.
- Commonsense based not only on massive amounts of knowledge, but also **massive amounts of fast and effective but unconscious inference**
- This is clearly true for **Vision** and **NLP**, but likely for **Everyday Reasoning**
- The **unconscious**, not necessarily Freudian, getting renewed attention:
 - ▷ *Strangers to Ourselves: the Adaptive Unconscious* by T. Wilson (2004)
 - ▷ *The New Unconscious*, by Ran R. Hassin et al. (Editors) (2004)
 - ▷ *Blink: The Power Of Thinking Without Thinking* by M. Gladwell (2005)
 - ▷ *Gut Feelings: The Intelligence of the Unconscious* by Gerd Gigerenzer (2007)
 - ▷ *Better Than Conscious?: Decision Making, the Human Mind, and Implications For Institutions* by C. Engel and W. Singer (2008)
 - ▷

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

Technical Challenges in Planning

- **Classical Planning**

- ▷ heuristics $h(s)$ are not **black boxes**; how to exploit **structure** further?

- **Probabilistic MDP & POMDP Planning**

- ▷ inference can't be at level of **states** or **belief states** but at level of **variables**

- **Multi-agent Planning**

- ▷ should go long way with **single-agent planning**; game theory seldom needed

- **Hierarchical Planning**

- ▷ how to **infer** and **use** hierarchies; what can be **abstracted away** and when?