Artificial Intelligence: From Programs to Solvers

Turing Session, ECAI-2012

Hector Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, Spain
8/2012

My Story in Short

- AI is brain child of Alan Turing and his universable, programmable computer

- Programming played key role in the early years of AI: 60's, 70's, but things started to change in the 80's as focus increasingly shifted from

  - **programs** for **ill-defined problems**, to
  - **general solvers** for **well-defined but intractable mathematical models**

- Models include SAT, CSPs, Bayesian Networks, Strips, POMDPs, Answer Set Programming, General Game Playing, etc; some of which are very **expressive.**

- Main challenge is **scalability**, and methodology that combines **theory** and **experiments** has enabled **sustained progress**

- In spite of apparent technical nature, agenda connects well with original **AI goals**, and scalable computation critical for understanding **human mind**

Importance of Programs in Early AI Work

In preface of 1963 edition of seminal *Computers and Thought* book

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 of *Computer and Thought*

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
- . . .

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: Generality

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 2012

Recent issues of main journal and conferences show papers on:

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

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

AI Models and Solvers

Problem $\Longrightarrow$ [*Solver*] $\Longrightarrow$ *Solution*

- Some basic models and solvers currently considered in AI:
  - ▷ **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
  - ▷ **Answer Set Programming:** find answer set of logic program
  - ▷ **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)
- Solvers all have a clear and crisp scope; **they are not architectures**
- Challenge is mainly **computational**: **how to scale up**
- Methodology is **empirical**: benchmarks and competitions
- Significant **progress** . . .

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

Some of the best BN solvers built over these formulations . . .

Planning

- Planning is the **model-based approach** to autonomous behavior, which contrasts with two other approaches:
  - ▷ **programming-based:** where control is **hardwired** by programmer or nature
  - ▷ **learning-based:** where control **learned** from experience
- **Planning models** describe actions, states, goals, and observations, and come with names such as **State models**, MDPs, and POMDPs, depending on nature of **feedback** and **dynamics**
- Planning models described in **compact form** through languages such as STRIPS
- The solution to such models is **computationally intractable**

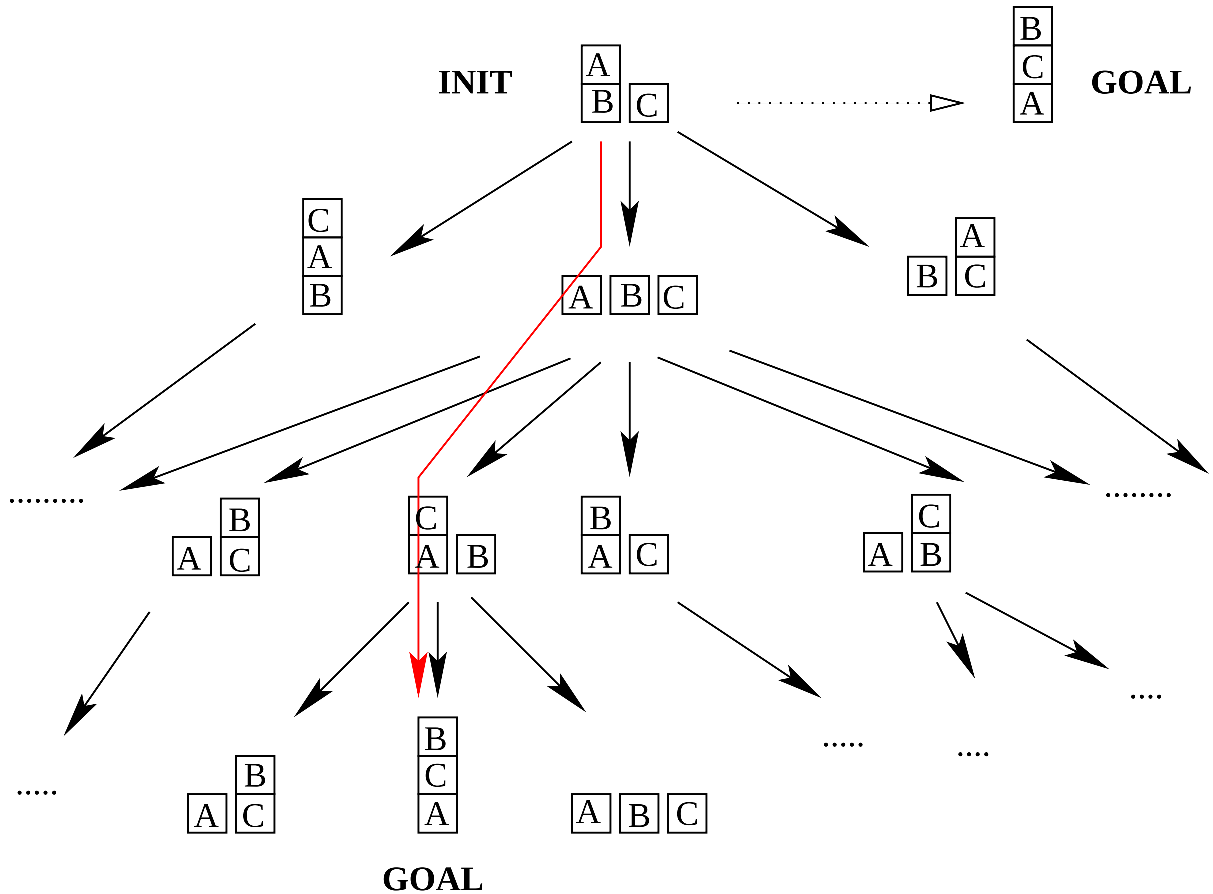
Basic (Classical) Planning Model and Task

- A system that can be in one of many **states**
- States assign **values** to a set of **variables**
- **Actions** change the value of certain variables
- **Task:** find **action sequence** to drive **initial state** into **target state**

$$\textit{Model} \Longrightarrow \boxed{\textit{Box}} \Longrightarrow \textit{Action sequence}$$

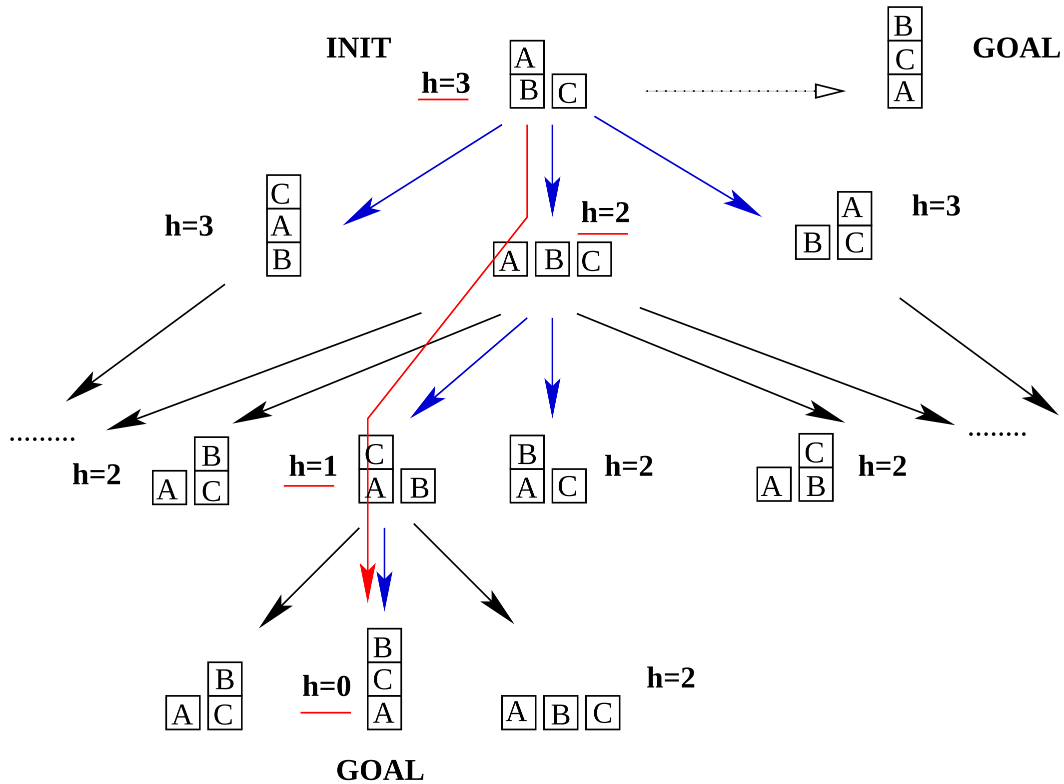
- **Complexity:** NP-hard; i.e., exponential in number of vars **in worst case**
- **Box** is generic; it should work on any domain no matter what variables are about

Concrete Example



- Given the **actions** that move a 'clear' block to the table or onto another 'clear' block, **find a plan** to achieve the goal
- How to find path in the graph whose size is **exponential** in number of blocks?

Problem Solved with Heuristics Derived Automatically



- **Heuristic evaluations** $h(s)$ provide 'sense-of-direction'
- Derived **efficiently** in a **domain-independent** fashion from **simplification** where effects made **monotonic** (delete relaxation).

Old Debates, New Insights?

- **Logic vs. Probabilistic Inference:** don't look that different now
- **Intelligence can't be rules all the way down:** not in Planning
- **Symbolic vs. Non-Symbolic:** are (learned) BNets and MDPs 'symbolic'?
- **GOFAI vs. Mainstream AI:** is GOFAI just 'old' AI, no longer current?
- **Solvers vs. Architectures:** architectures don't "solve" anything; solvers do.
- **Mind as Architecture or Solver?** Adaptive, heuristic, multiagent POMDP solver?
- **Can AI shed light on Unconscious Inference and the Emotions?** . . . . . .

Unconscious Inference

- We have learned a lot about **effective inference mechanisms** in last 20–30 years from work on **domain-independent** solvers
- The problem of AI in the 80's (the 'knowledge-based' approach), was probably lack of **mechanisms**, 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)*
  - ▷ *. . .*
  - ▷ *Thinking, Fast and Slow. D. Kahneman (2011)*

The appraisals/heuristics $h(s)$ 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

Summary

- Shift in AI since 80's from **programs** over ill-defined problems to **solvers** over well-defined but intractable mathematical models

- **Solvers** unlike other programs are **general** as they do not target individual problems but infinite sets of problems (**models**)

- The challenge is **computational**, methodology is **empirical**; and consistent **progress** achieved

- While agenda is technical, resulting ideas likely to be relevant for understanding **general intelligence** and **human cognition**

- Many **challenges** and **open problems**; e.g., in planning:
  - ▷ planning in presence of other intentional agents (**multiagent planning**)
  - ▷ computation of general and compact policies (e.g. car racing for **any track**)
  - ▷ learning the models and in particular **learning the state space**
  - ▷ . . .