Planning and Plan Recognition

Hector Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, Spain
Dagstuhl Seminar 4/2011

Planning and Autonomous Behavior

Three approaches to the problem of **selecting the action to do next**:

1. **Programming: specify** control by hand
2. **Learning: learn** control from experience
3. **Planning: derive** control from model

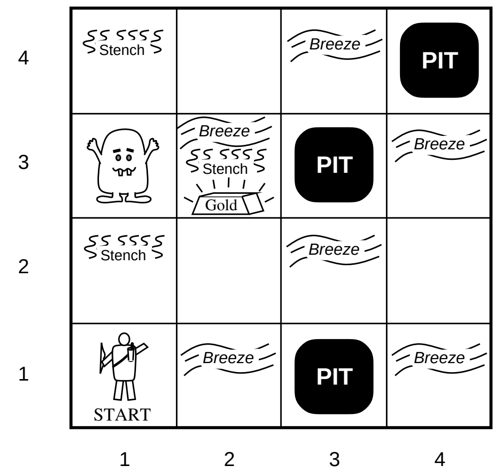
Planning is the **model-based** approach to action selection: behavior obtained from **model** of the **actions**, **sensors**, **preferences**, and **goals**

$$\mathit{Model} \Longrightarrow \boxed{\mathit{Planner}} \Longrightarrow \mathit{Controller}$$

Wumpus World PEAS description

Performance measure
- gold +1000, death -1000
- -1 per step, -10 for using the arrow

Environment
- Squares adjacent to wumpus are smelly
- Squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Shooting kills wumpus if you are facing it
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
- Releasing drops the gold in same square



Actuators Left turn, Right turn,
Forward, Grab, Release, Shoot

Sensors Breeze, Glitter, Smell

Chapter 7 5

Outline of the Talk

- **Planning Models**
  - ▷ *Many dimensions: uncertainty, feedback, costs, . . .*
- **Planning Algorithms**
  - ▷ *Key issue is scalability*
- **Plan Recognition as Planning**
  - ▷ *Behavior generation algorithms can be used for recognition*
- **Variations: HTN Planning**
  - ▷ *Between programming and planning*

Basic State Model: Classical 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 transition function** $s' = f(a, s)$ for $a \in A(s)$
- positive **action costs** $c(a, s)$

A **solution** is a sequence of applicable actions that maps s_0 into S_G, and it is **optimal** if it minimizes sum of action costs ($\#$ of steps)

Different **models** obtained by relaxing assumptions in **bold** . . .

Uncertainty and Full Feedback: Markov Decision Processes

MDPs are **fully observable, probabilistic** state models:

- a state space S
- initial state $s_0 \in S$
- a set $G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each state $s \in S$
- **transition probabilities** $P_a(s'|s)$ for $s \in S$ and $a \in A(s)$
- action costs $c(a, s) > 0$

– **Solutions** are **functions (policies)** mapping states into actions

– **Optimal** solutions minimize **expected cost** to goal

Uncertainty and Partial Feedback: Partially Observable MDPs (POMDPs)

POMDPs are **partially observable, probabilistic** state models:

- states $s \in S$
- actions $A(s) \subseteq A$
- transition probabilities $P_a(s'|s)$ for $s \in S$ and $a \in A(s)$
- observable goal states $S_G \subseteq S$

- initial **belief state** b_0
- **sensor model** given by probabilities $P_a(o|s)$, $o \in O$, $s \in S$

– **Belief states** are probability distributions over S

– **Solutions** are policies that map belief states into actions

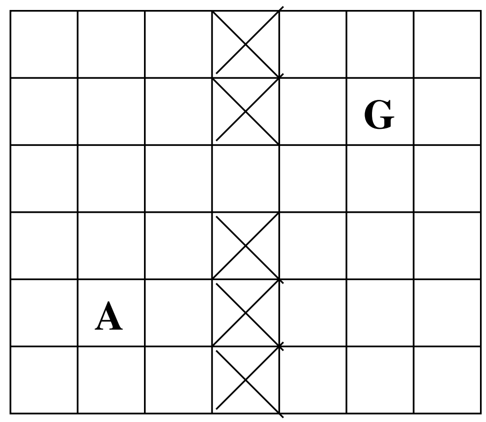
– **Optimal** policies minimize **expected** cost to go from b_0 to b_F

Further Variations: Discounted Reward and Qualitative Models

- **Rewards** used often instead of **costs**, along with a **discount factor** γ, $0 < \gamma < 1$
- Rewards can be positive, negative, or zero, and **goals** not needed then
- Best policies then not the ones that **minimize expected cost to goal**, but that **maximize discounted accumulated reward**
- Still goal-based formulation strictly **more general**, even if rewards, unlike costs, can be **positive** or **negative** (!)
- **Qualitative** version of MDPs and POMDPs where**uncertainty** represented by **sets of states** rather than **probability distributions** also used
- Planners for qualitative POMDPs, referred to as **contingent planners** or **planners with sensing**

Example

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



- If actions deterministic and initial location known, planning problem is **classical**
- If actions stochastic and location observable, problem is an **MDP**
- If actions stochastic and location partially observable, problem is a **POMDP**

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

Compact Model Representations and Planning Languages

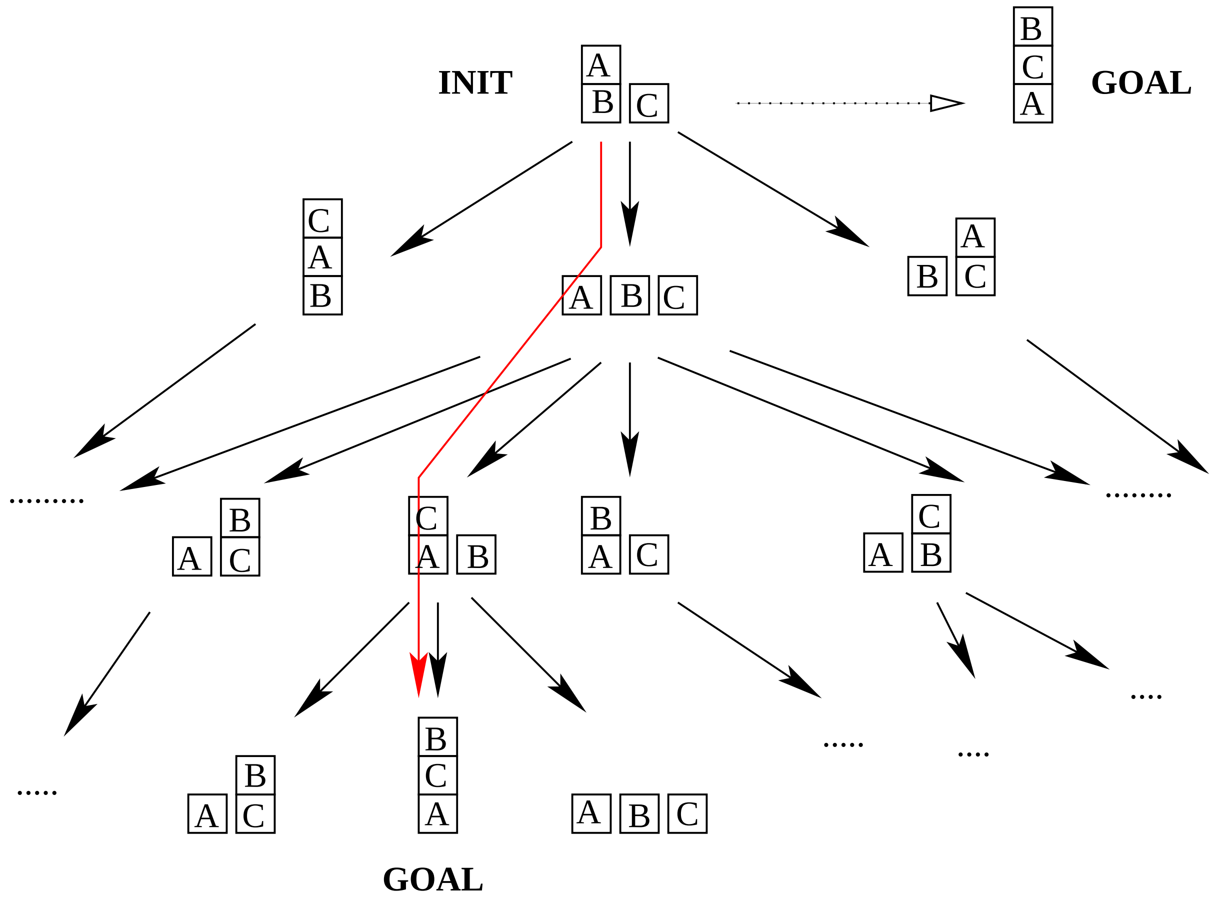
- Planning languages defined in terms of **variables** that can take some **values**
- The **states** are the possible value assignments to these **variables**
- The **number** of states is **exponential** in number of variables
- **Initial (belief) state** and **goals** expressed in terms of variables
- **Action effects** (state transitions) expressed **locally** often
  - ▷ **adding** values that become true, and
  - ▷ **deleting** values that become false

$$\textit{Model Description} \Longrightarrow \boxed{\textit{Planner}} \Longrightarrow \textit{Controller}$$

AI Planning: Status

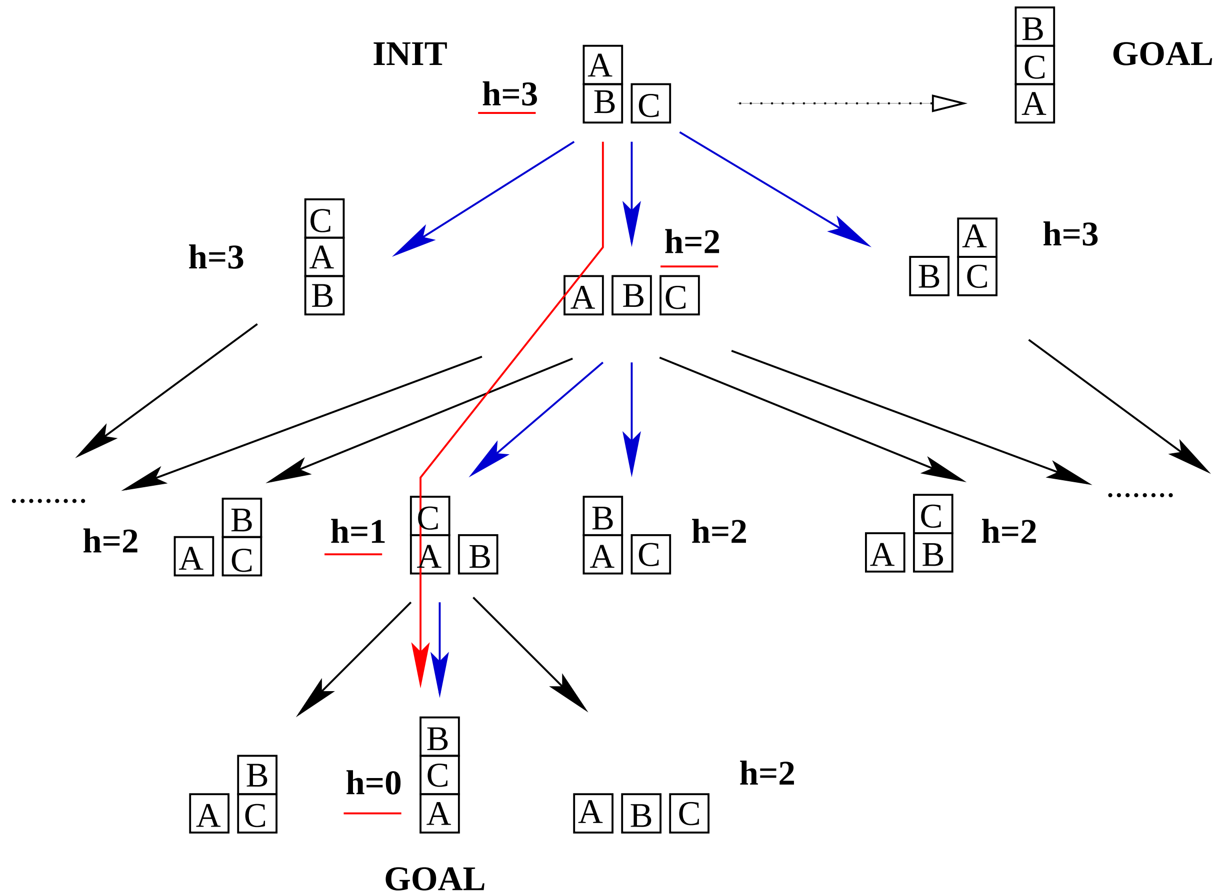
- The good news: **classical planning works** pretty well
  - ▷ *Large problems solved very fast (non-optimally)*
- **Model simple but useful**
  - ▷ *Operators not primitive; can be policies themselves*
  - ▷ *Fast closed-loop replanning able to cope with uncertainty sometimes*
- **Limitations:**
  - ▷ *Uncertainty, Incomplete Information, Preferences, . . .*
- **Beyond classical planning:**
  - ▷ *Top-down approaches:* MDP and POMDP solvers, etc
  - ▷ *Bottom-up approaches:* Transformations into classical planning . . .

Example – Classical Planning



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

How is the problem solved?



- Provided with **heuristic evaluation** h, plan found **greedily**
- Heuristic h provides **estimates of cost-to-go**

Where do heuristic evaluations come from?

- **Approximate distances** $h(s)$ computed from a **simplification** of the problem (relaxation)
- Most common simplification is to drop **deletes** from action effects
- **Problem without deletes is tractable** and can be solved efficiently (linear-time)
- Heuristic $h(s)$ represents **cost of simplified problem** from s
- Many other ideas have been tried but **experiments** show that **they do not work** as well; **scalability is a tough filter!**
- Approaches based on **SAT** have been shown to work well too.

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

- they are **fast**, **effective**, and **domain-independent**

  they apply to all problems fitting the model

- 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 provide agent with **sense of direction**; **'gut feelings'**

  a guide to action that avoids infinite regresses in the decision process (Damasio, Gigerenzer, . . .)

Scalability important and likely to be relevant for understanding cognition too

Heuristic and Value Functions in other Planning Models

- A **greedy action** a is one that minimizes **expected cost-to-go** given by **value or heuristic function** V. If action costs uniform:

In Classical Planning:	$\arg\min_a V(s')$
In MDPs:	$\arg\min_a \sum_{s'} P_a(s'|s)V(s')$
In POMDPs:	$\arg\min_a \sum_o b_a(o)V(b_a^o)$

- If value function $V(\cdot)$ good enough, **greedy action is optimal**

- Many methods for obtaining such functions

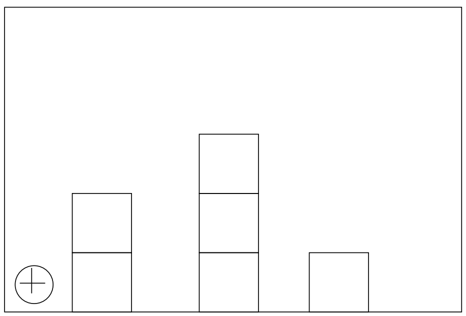
- Distinction between **programmed/learned/derived** behaviors echoed in **value functions**:

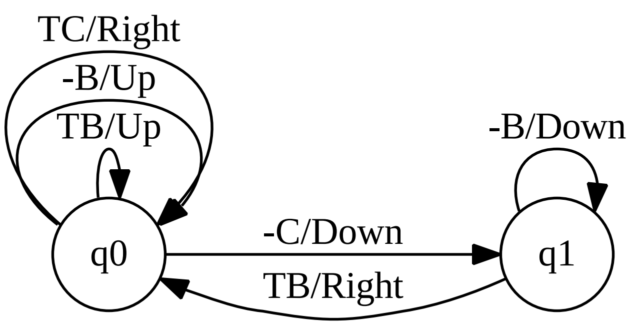
  ▷ Evaluation functions **hardwired** in Chess
  ▷ Valuation functions **learned** from experience in Reinforcement Learning
  ▷ Heuristic functions **derived** from relaxed models in Planning

Transformations are also powerful

- **Problem** P: find **green block** using visual-marker (circle) that can move around one cell at a time (à la Chapman and Ballard)

- **Observables:** Whether cell marked contains a green block (G), non-green block (B), or neither (C); and whether on table (T) or not (–)





- Finite state controller on the right **solves** the problem

- Controller obtained by running a **classical planner** over **transformed problem**

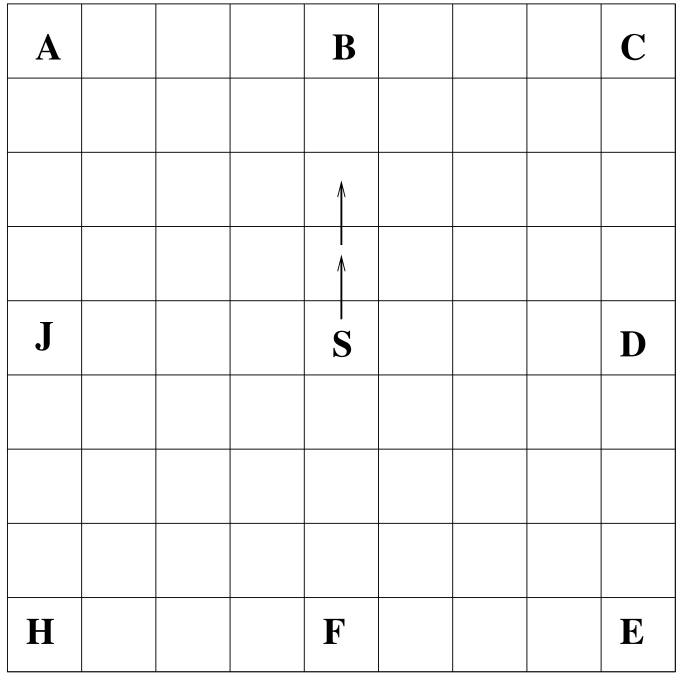
- Controller works for **any number of blocks** and **any configuration**!

Planning and Plan Recognition

- Plan Recognition related to Plan Generation but had not built on it until recently
- Rather Plan Recognition addressed as Deduction, Evidential Reasoning (HMMs,DBNs), Parsing (Grammars), etc; or through specialized methods

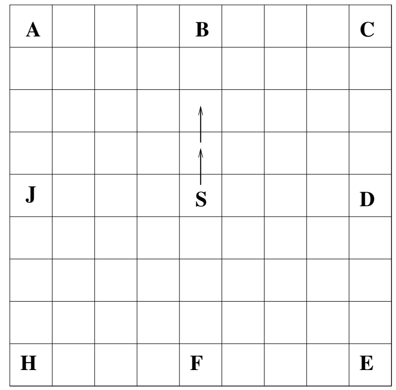
Next: **How to do plan recognition using a Classical/MDP/POMDP Planner?**

Example



- Agent can **move** one unit in the four directions
- Possible **targets** are A, B, C, . . .
- Starting in S, he is **observed** to move up twice
- **Where** is he going? Why?

Example (cont'd)



- From Bayes, **goal posterior** is $P(G|O) = \alpha\, P(O|G)\, P(G)$, $G \in \mathcal{G}$
- If **priors** $P(G)$ given for each goal in $\mathcal{G}$, the question is what is $P(O|G)$
- $P(O|G)$ measures **how well goal** G **predicts observed actions** O
- In **classical** setting,
  - ▷ G predicts O **worst** when needs to get off the way **to comply with** O
  - ▷ G predicts O **best** when needs to get off the way **not** to comply with O

Posterior Probabilities from Plan Costs

- From Bayes, **goal posterior** is $P(G|O) = \alpha \, P(O|G) \, P(G)$,
- If **priors** $P(G)$ given, set $P(O|G)$ to

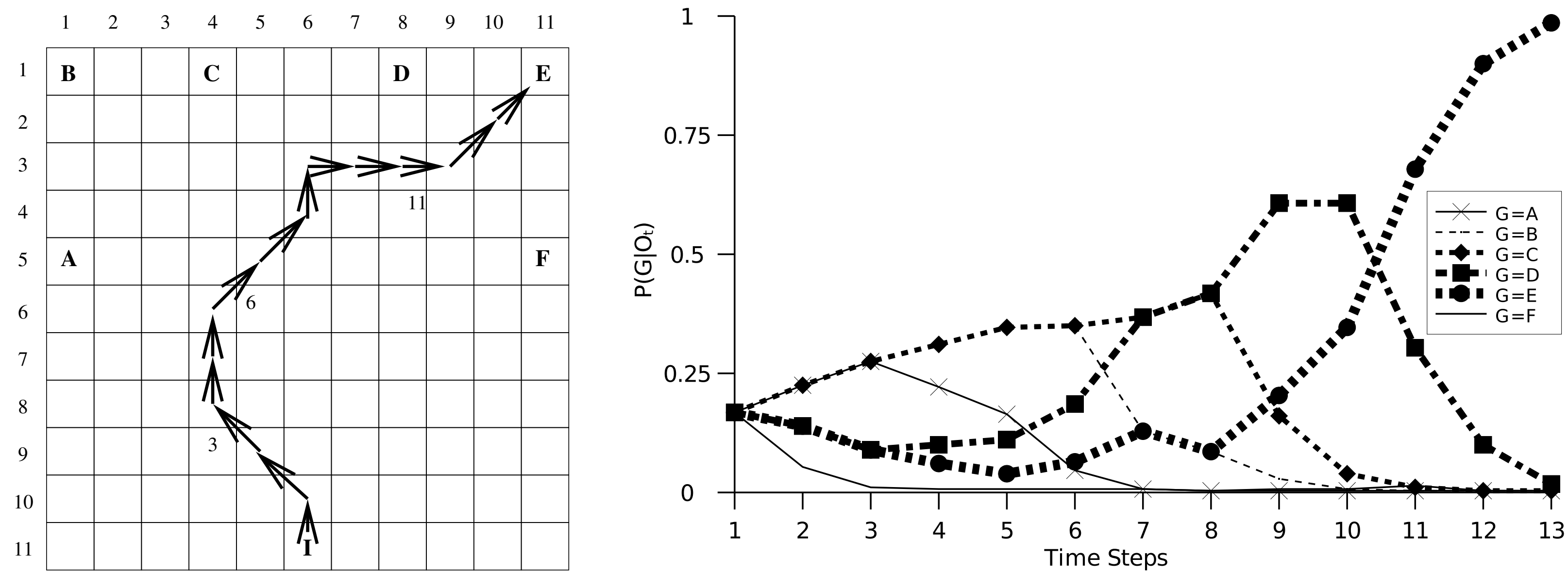
$$\textbf{function}(c(G+\overline{O}) \, - \, c(G+O))$$

  - ▷ $c(G+O)$: cost of achieving G **while complying with** O
  - ▷ $c(G+\overline{O})$: cost of achieving G **while not complying with** O

– Costs $c(G+O)$ and $c(G+\overline{O})$ **computed** by **classical planner**

– Goals of **complying** and **not complying** with O translated into normal goals

Example Revisited: Noisy Walk



- 'Noisy walk' and possible targets; **posterior** $P(G|O)$ of each target G as a function of time (Ramirez & G. 2010)
- $P(O|G)$ set to $\mathbf{sigmoid}(\beta\,\Delta(G,O))$, where $\Delta(G,O) = c(G+\overline{O}) - c(G+O)$
- This follows from Boltzmann dist. $exp\{-\beta\, c(G+X)\}$ for $P(X|G)$, $X \in \{O, \overline{O}\}$.

Plan Recognition over MDPs and POMDPs

- In MDPs, given $V_G(s)$, define $P(a|s;G)$
- Then $P(O|s_0;G)$ for $O = a_0, s_1, a_1, s_2, \ldots$ follows from basic probability laws
- In POMDPs, given $V_G(b)$, define $P(a|b;G)$
- Then $P(O|b_0;G)$ for $O = a_0, o_1, a_1, o_2, \ldots$ follows from basic probability laws
- In both cases, posteriors $P(G|O)$ follow from Bayes Rule

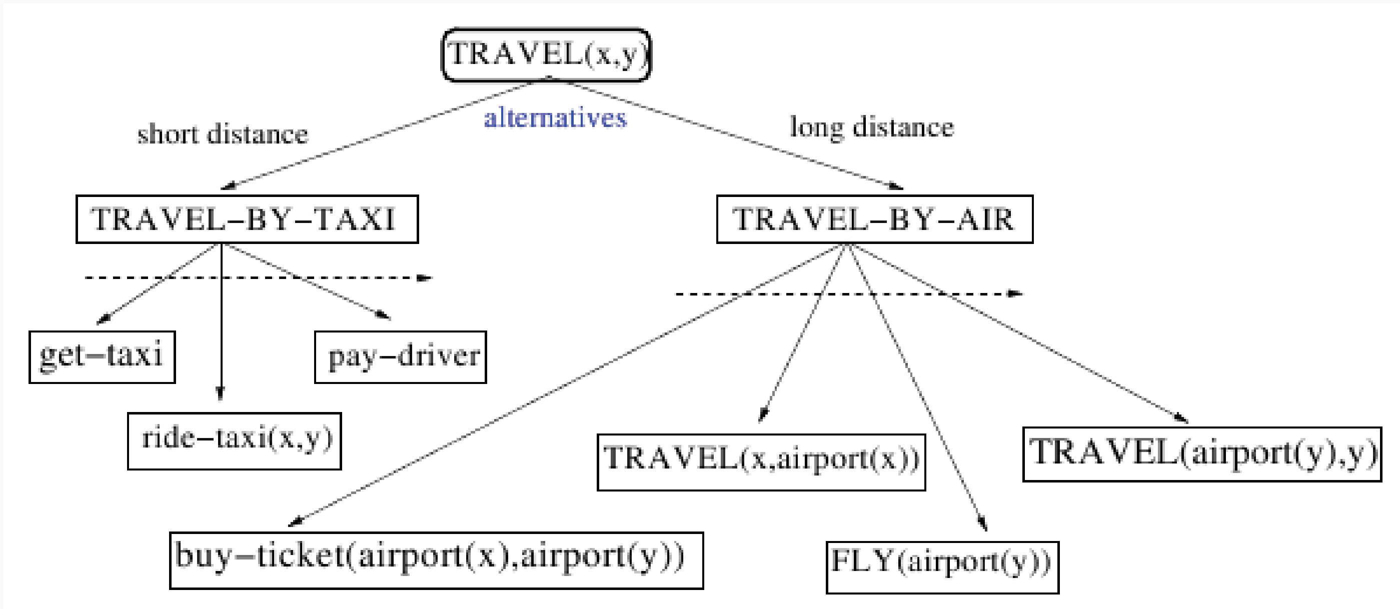
Example: Plan Recognition over POMDPs

- Agent is looking for item A or B which can be in one of three drawers 1, 2, or 3
- Agent doesn't know where A and B are, but has **priors** $P(A@i)$, $P(B@i)$
- He can open and close drawers, **to look** for item in open drawer, and grab an item from drawer if known to be there
- The sensing action, however, is not perfect, and may fail to see item even if in drawer
- Agent observed to do $O = \{open(1), open(2), open(1)\}$
- If possible goals G are to have A, B, or both, and priors given, what's posterior $P(G|O)$?

What about Hierarchical Task Network (HTN) Planning?

- **HTN Planning** is a different type of planning where model features **control knowledge**
- This extra knowledge takes the form of **high-level tasks** and **methods** for **decomposing** them into **subtasks**
- The **primitive tasks** can't be decomposed and represent the domain **actions**
- HTN Planning quite popular in both **planning applications** and **plan recognition**, where **libraries** commonly expressed as HTN methods
- In many cases, and often in plan recognition, HTN libraries define **acyclic AND/OR Graphs**

HTN Planning: An Example



State: set of atoms: $At(loc)$.

Tasks: primitive or compound.

Task Network: set of tasks T + order/state constraints ϕ.

Method: a way to solve a compound task e using a network d.

Plan: a sequence of primitive tasks.

How to Do Recognition of HTN tasks?

Three possible answers:

- **Transform** recognition into **parsing** over suitable grammar, and use corresponding parsing algorithm
- Use **specialized algorithms**
- **Compile** into **classical planning**, and do plan recognition with a **classical planner** (for the compilation, Lekavý & Návrat 2007; Alford, Kuter, Nau 2009)

What about Variables?

- Current planners **ground** all actions compiling **variables** away

- In some applications (Koller and Hoffmann 2010), this can be a bottleneck

- Prior grounding, however, is not strictly required, it's done for **efficiency**

- In other applications, reasoning about **variable bindings** seems required; e.g.,

  Jack went to the store. He found some milk on the shelf. He paid for **it** *and left.*
  What does '**it**' refer to?

- Yet, this doesn't seem to require variables **in the planner** either; one can try the possible substitutions of **it**, and then see which **ground plan** makes most sense for each goal (e.g., $G =$ 'buy milk').

- More precisely, if the observations O contain 'variables' (pronouns), one could set $c(G+O)$ to $\min_{O_i} c(G+O_i)$, where O_i are the possible groundings of O

Summary

- Planning is the **model-based approach** to autonomous behavior
- **Models** describe actions, sensors, preferences, and goals
- Derivation of controller from model **intractable** in all cases
- Automatically derived heuristics computationally useful in **classical planning**
- Similar **value functions** used to solve **MDPs** and **POMDPs**
- **Plan recognition** over a given planning model, solvable with **planner** over model
- Key idea is definiton of **likelihoods** $P(O|G)$ from **costs**
- **Plan libraries** addressed in this way by compiling them into classical problems

References

[AKN09] R. Alford, U. Kuter, and D. Nau. Translating HTNs to PDDL: a small amount of domain knowledge can go a long way. In *Proc. IJCAI*, pages 1629–1634, 2009.

[AZK05] D. Avrahami-Zilberbrand and G. A. Kaminka. Fast and complete symbolic plan recognition. In *Proceedings of IJCAI*, pages 653–658, 2005.

[BBS95] A. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

[Ber95] D. Bertsekas. *Dynamic Programming and Optimal Control, Vols 1 and 2.* Athena Scientific, 1995.

[BG00] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proc. of AIPS-2000*, pages 52–61. AAAI Press, 2000.

[BG01] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1–2):5–33, 2001.

[BG09] B. Bonet and H. Geffner. Solving POMDPs: RTDP-Bel vs. Point-based algorithms. In *Proceedings IJCAI-09*, pages 1641–1646, 2009.

[BST09] C. L. Baker, R. Saxe, and J. B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[BTS07] C.L. Baker, J.B. Tenenbaum, and R.R. Saxe. Goal inference as inverse planning. In *Proceedings of the 29th annual meeting of the cognitive science society*. Citeseer, 2007.

[GG09] C. W. Geib and R. P. Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173(11):1101–1132, 2009.

[HN01] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[KA86] H. Kautz and J. F. Allen. Generalized plan recognition. In *Proc. AAAI-86*, pages 32–37, 1986.

[KH10] A. Koller and J. Hoffmann. Waking up a sleeping rabbit: On natural-language sentence generation with FF. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling*, 2010.

[KLC99] L. P. Kaelbling, M. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1999.

[LN07] M. Lekavý and P. Návrat. Expressivity of Strips-like and HTN-like planning. In *Proc. 1st KES Int. Symp.KES-AMSTA 2007*, pages 121–130, 2007.

[McD98] D. McDermott. PDDL – the planning domain definition language. At `http://ftp.cs.yale.edu/pub/mcdermott`, 1998.

[PGT06] J. Pineau, G. Gordon, and S. Thrun. Anytime point-based approximations for large pomdps. *JAIR*, 27:335–380, 2006.

[PW02] D.V. Pynadath and M.P. Wellman. Generalized queries on probabilistic context-free grammars. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):65–77, 2002.

[RG09] M. Ramirez and H. Geffner. Plan recognition as planning. In *Proc. 21st Intl. Joint Conf. on Artificial Intelligence*, pages 1778–1783. AAAI Press, 2009.

[RG10] M. Ramirez and H. Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Proc. AAAI-10*. AAAI Press, 2010.

[RG11] M. Ramirez and H. Geffner. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *Proc. IJCAI-11*, 2011.

[RHW08] S. Richter, M. Helmert, and M. Westphal. Landmarks revisited. In *Proc. AAAI*, pages 975–982, 2008.