

# Metareasoning as a Formal Computational Problem

Vincent Conitzer  
Duke University

# **Approach:** define some **special cases** of the **metareasoning problem** as **formal computational problems**

- Allows for precise analysis using **complexity theory**
  - Will discuss some results by C. & Sandholm [IJCAI03] showing that solving them **optimally** is hard
- Relevant for **any general metareasoning system** that hopes to address these cases
  - Computational hardness **transfers** from the specific to the general
- Provides a good starting point for debate
  - Are these the “**right**” (important) special cases?
  - If so, **what can we do?**
    - Approximately optimal metareasoning?
    - Meta-metareasoning?

# Three metareasoning problems

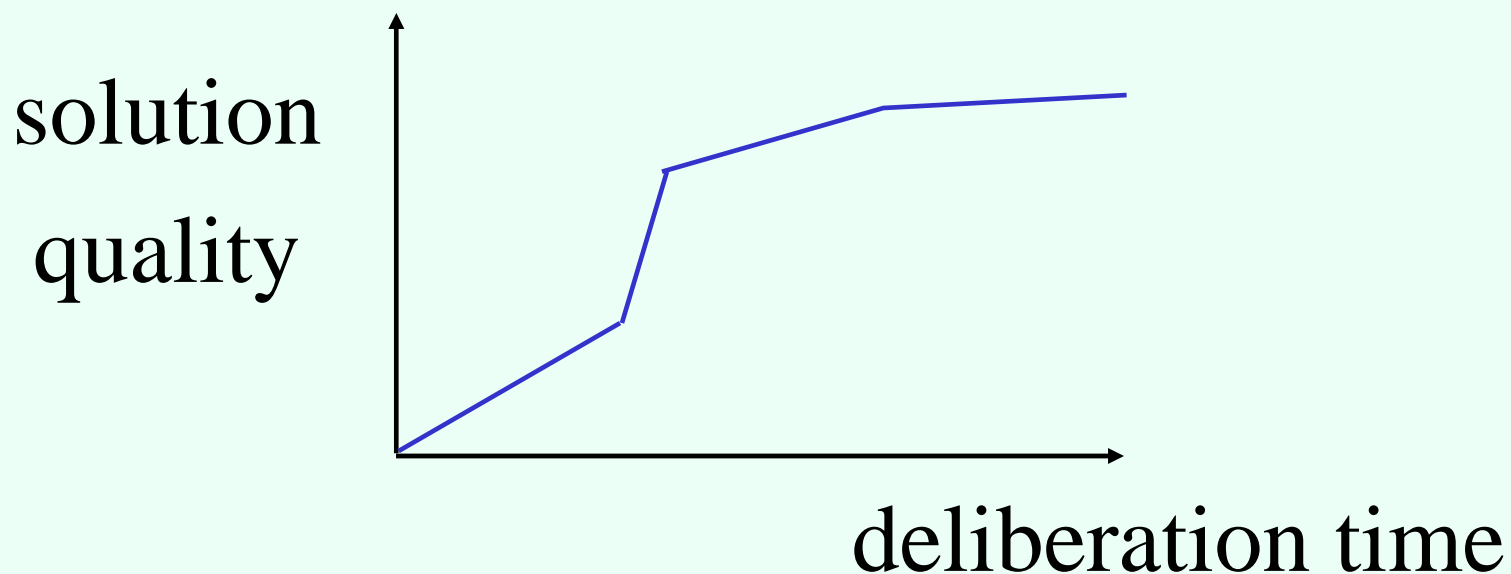
[taken from C. & Sandholm IJCAI03, including the complexity results below]

- **PERFORMANCE-PROFILES**
  - How should an agent distribute its deliberation time over multiple disjoint problems?
- **ACTION-EVALUATION**
  - How should an agent distribute its deliberation time over multiple options available to it?
- **STATE-DISAMBIGUATION**
  - What is the best plan for trying to discern the state of the world?

# PERFORMANCE PROFILES

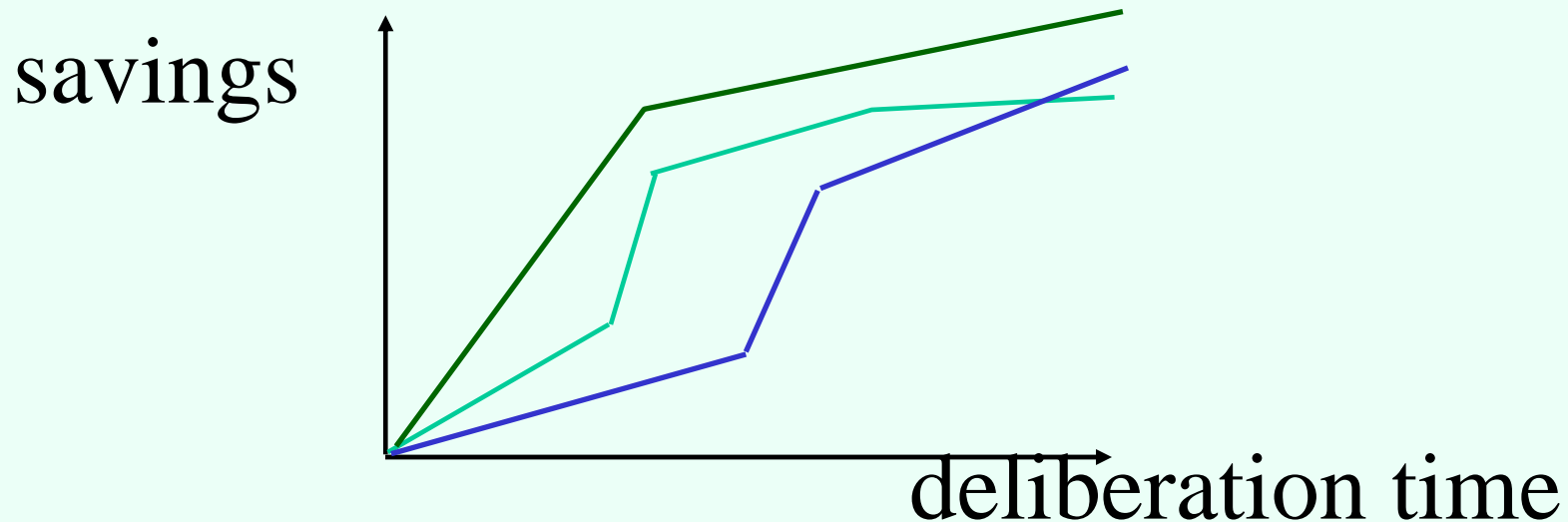
# What is a **performance profile**?

- Assume we can **perfectly** predict how the quality of our solution will improve as a function of the deliberation time spent on it
- This function is a *performance profile* (*curve*)



# The performance-profiles problem

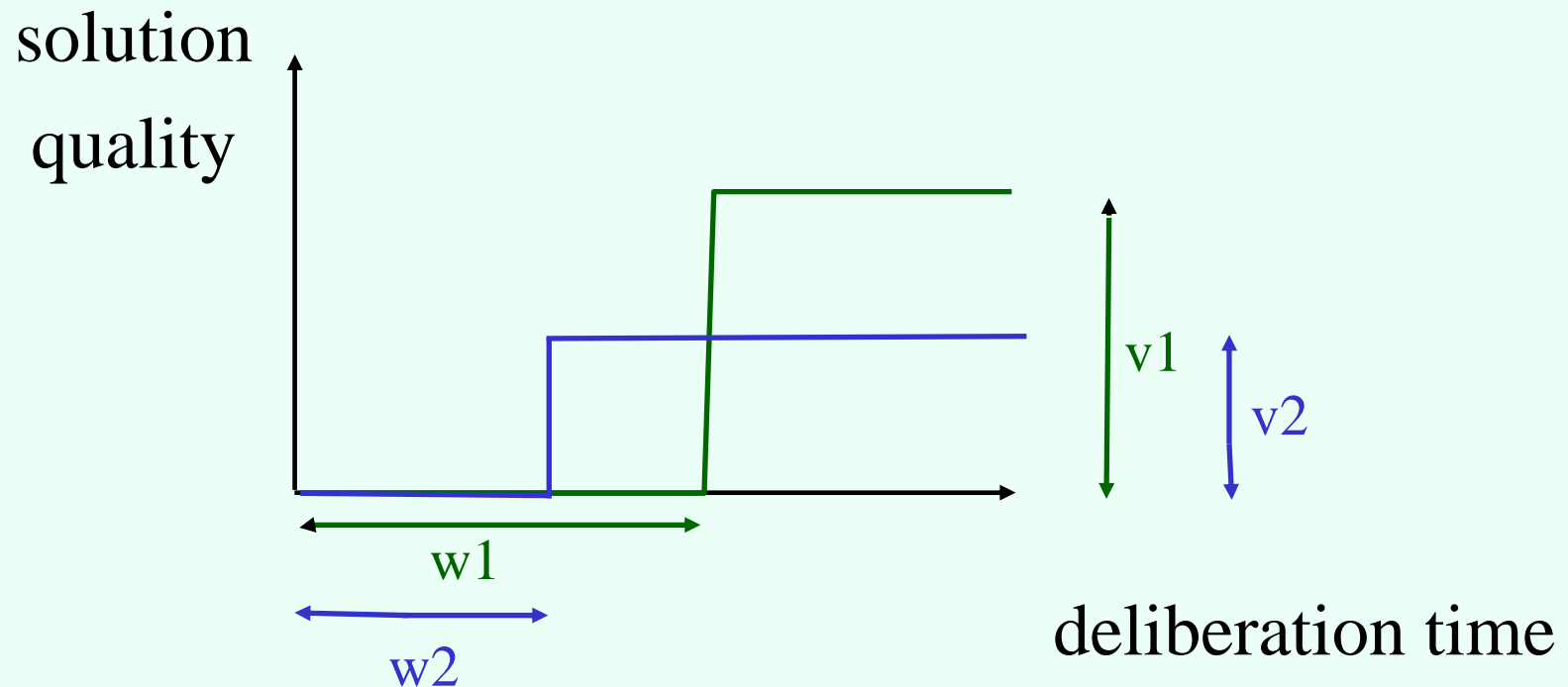
- Suppose:
  - We have **multiple disjoint problems** (with performance profiles) to compute on;
  - Our utility is the **sum** of the solution qualities.
- Example: a newspaper company has to solve **vehicle routing problems** to deliver the newspaper orders for the day
  - One for each of the cities it delivers in
  - Has until 5am to improve the solutions
  - Wants to achieve maximum (overall) savings over standard route
- How should it schedule its computation time?



# Complexity result

- **Thm.** PERFORMANCE-PROFILES is NP-hard.
  - Even with piecewise linear performance profiles
  - Reduction is from KNAPSACK

- Basic idea:



- Solvable in polynomial time with *concave* profiles
  - [Boddy and Dean, 94]

# ACTION EVALUATION

# An action evaluation tree

- Robot is considering whether to dig for gold
- The prior probability of there being gold is  $1/8$
- Robot can test for gold before digging
  - Alternatively: robot could *reason* about whether there is gold

• Say that:

$$P(\text{test positive}|\text{gold}) = 14/15$$

$$P(\text{test positive}|\text{no gold}) = 1/15$$

• Then (using Bayes' rule)

$$P(\text{test positive}) = 7/40$$

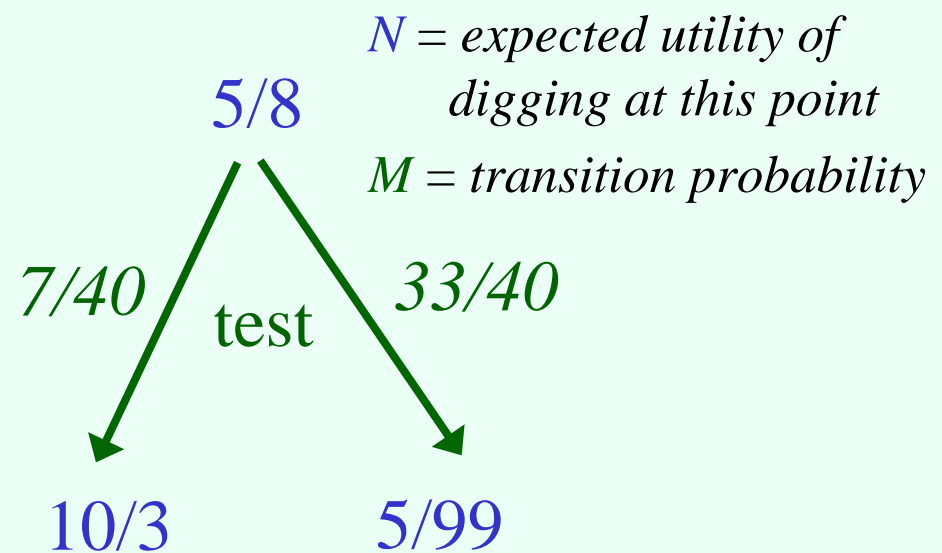
$$P(\text{gold}|\text{test positive}) = 2/3$$

$$P(\text{gold}|\text{test negative}) = 1/99$$

• Say utility of finding gold is 5

• Then the **action evaluation tree** to the right represents the predicament

• In general, action evaluation trees can have multiple tests/reasoning steps

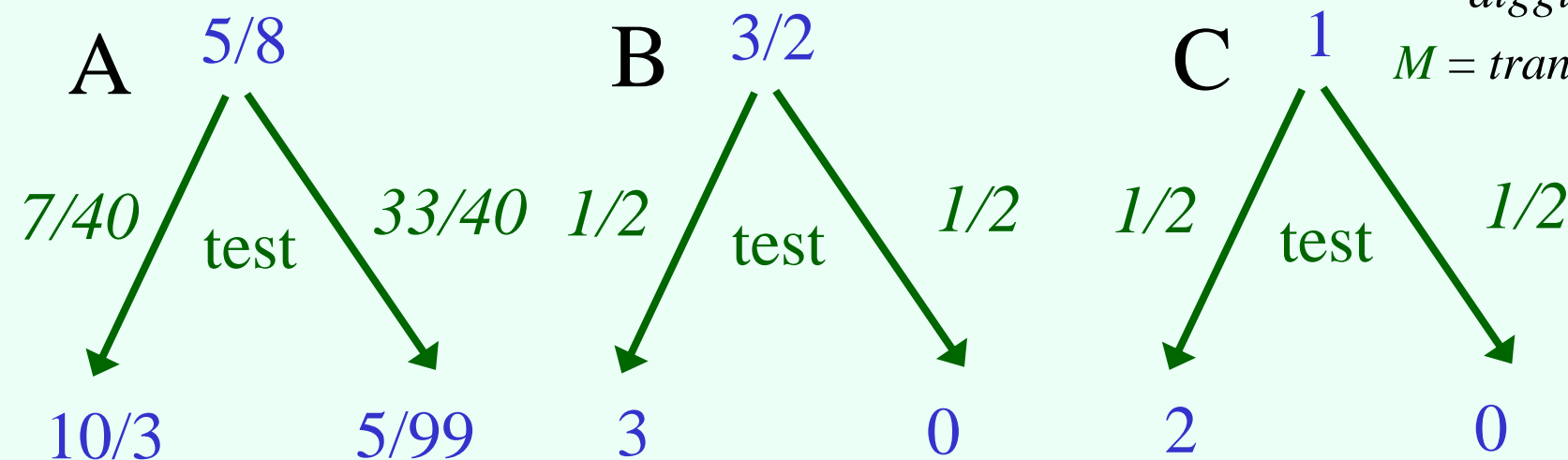


# Action evaluation problem: example

- Suppose robot can dig in (at most one of) locations A, B, and C
- A may have gold (as before)
- B may have silver, C may have copper
- (For simplicity) B and C have errorless tests
- Tests take different amounts of time
  - Say, tests at A and C take 2 units, test at B 3 units
  - **Deadline:** 5 units of time available for testing

$N$  = expected utility of digging here now

$M$  = transition probability



- **Optimal testing plan:** test at B first; if positive, test at A; otherwise at C

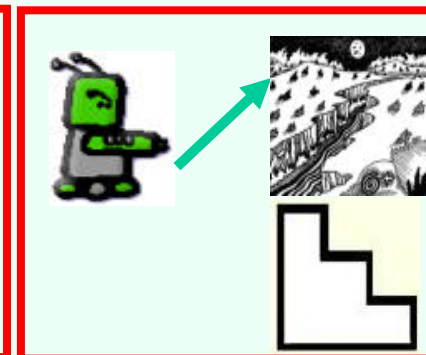
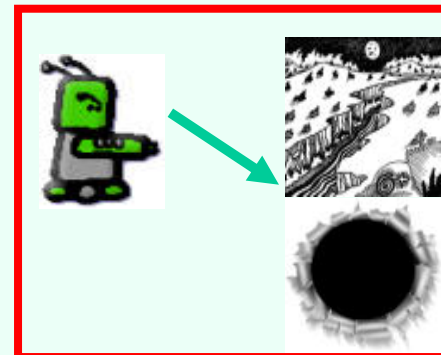
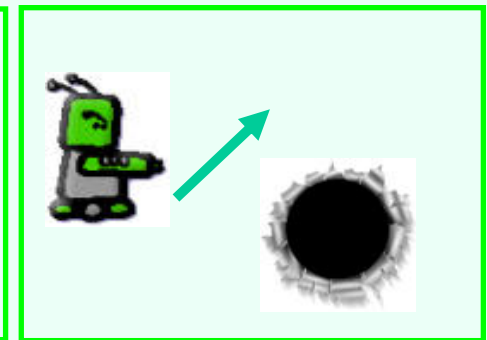
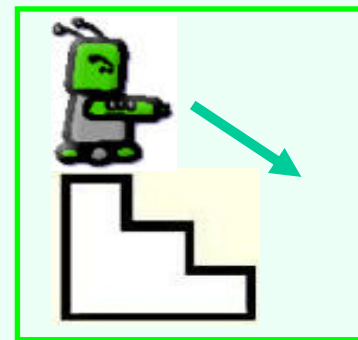
# General definition and complexity

- General ACTION-EVALUATION problem:
  - Given a set of action evaluation trees, what is the expected utility of the best deliberation plan?
- **Thm.** ACTION-EVALUATION is NP-hard even when every tree has depth at most 1, branching factor 2, & all leaf values -1, 0, or 1
- **Open question.** Is the general problem even harder (for example, PSPACE-hard?)

# STATE DISAMBIGUATION

# State disambiguation: example

- A robot encounters a gap in the floor in front of it
- The gap can be a **staircase (S)**, **hole (H)**, or **canyon (C)**
- There are three actions: **descend (d)**, **jump (j)**, or **walk away (w)**
- $u(S, d) = 2$  (new floors are very interesting)
- $u(H, j) = 1$  (passing a hole is somewhat interesting)
- $u(S, w) = u(H, w) = u(C, w) = 0$
- $u(S, j) = u(H, d) = u(C, j) = u(C, d) = -\infty$  (the robot is destroyed)



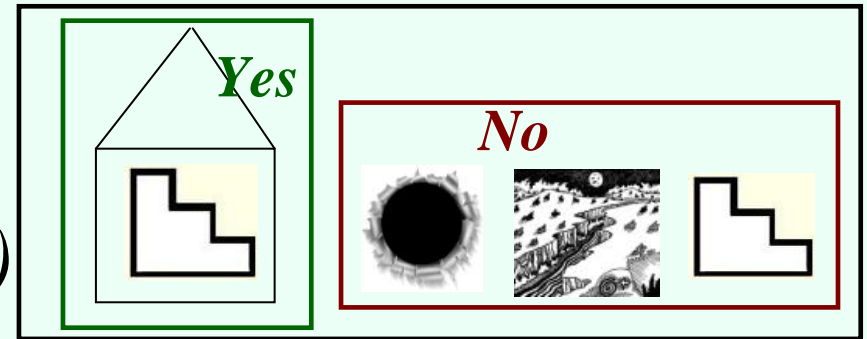
# State disambiguation: example (cont.)

- To find out what is in front of robot, it has some tests/queries that it can run

- Test (1) answers: “Am I inside a building?”

- “Yes” is consistent only with S
- “No” is consistent with H, C, and S (staircases occur outside, too)

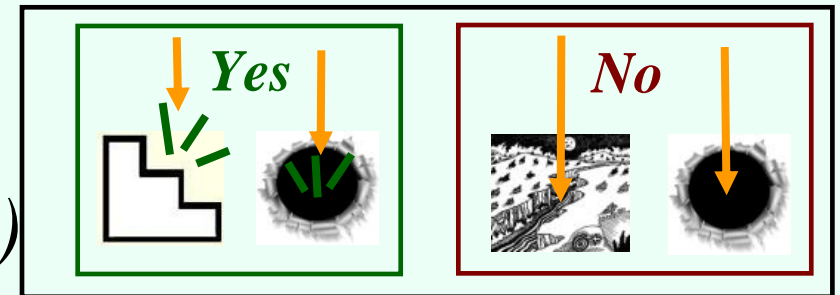
*Test (1)*



- Test (2) answers: “If I drop an item in front of me, do I hear a sound?”

- “Yes” is consistent with S, H
- “No” is consistent with C, H (some holes are very deep)

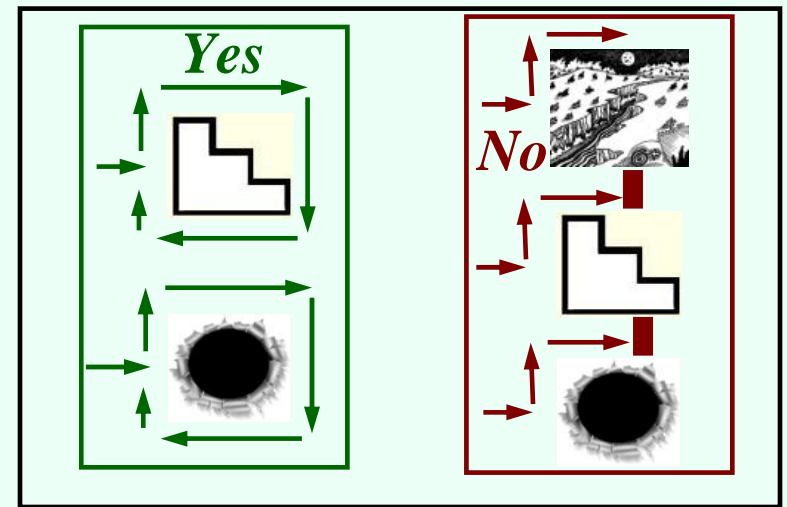
*Test (2)*



- Test (3) answers: “Can I walk around the gap?”

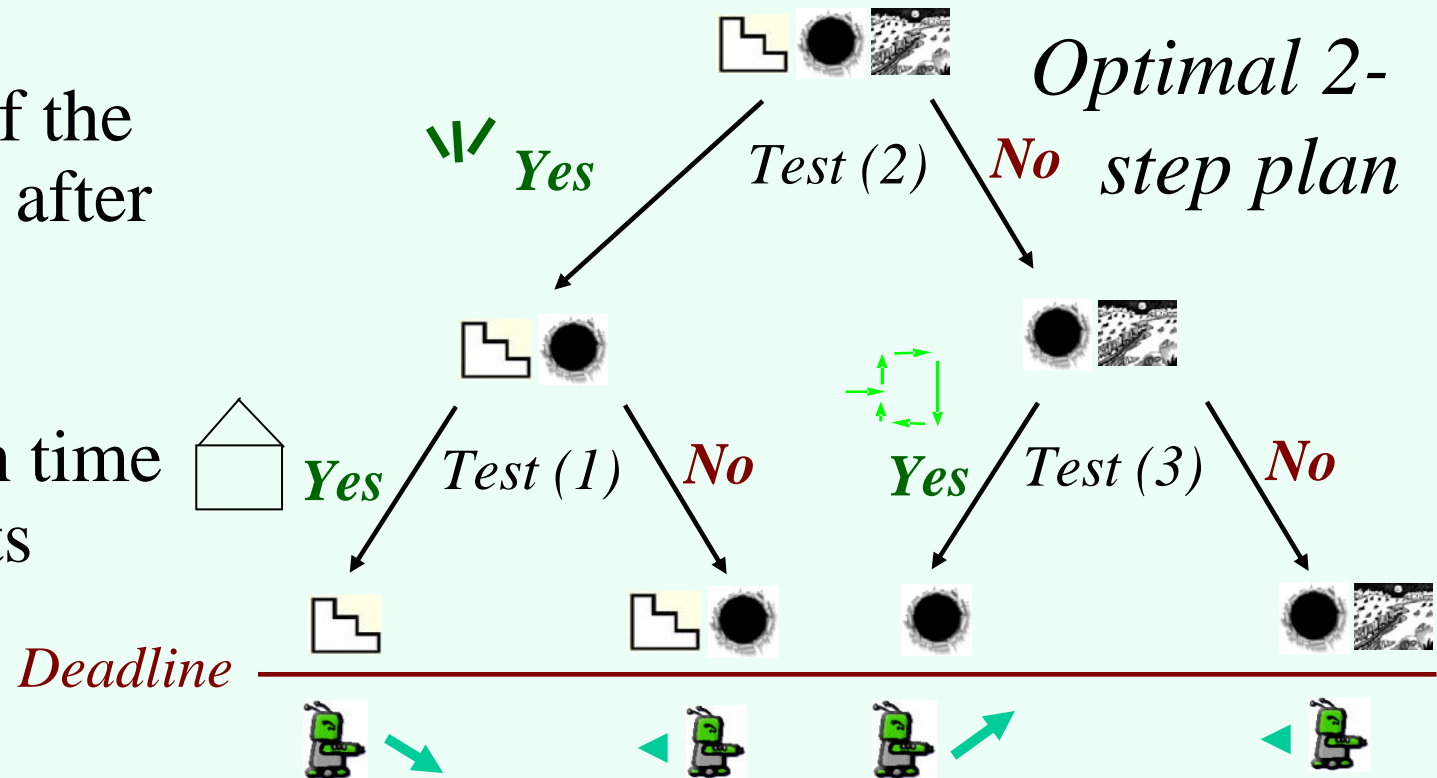
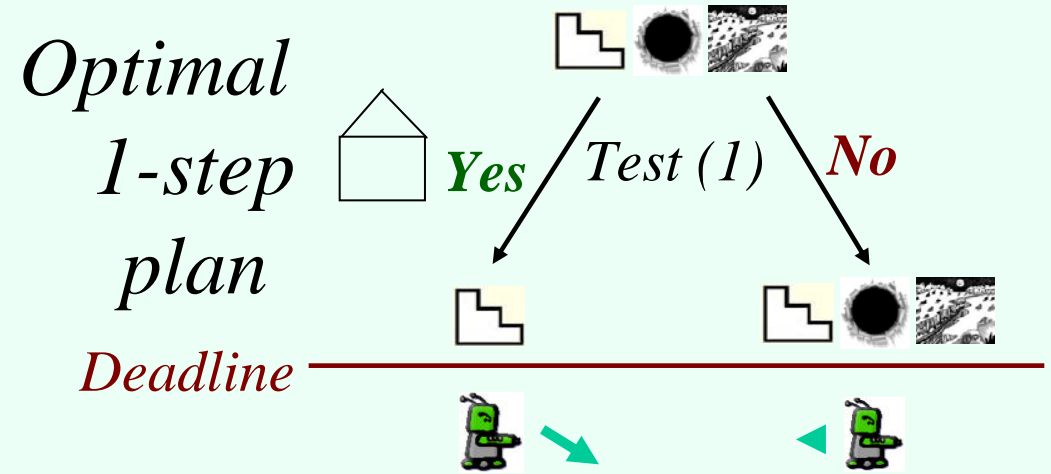
- “Yes” is consistent with S, H
- “No” is consistent with S, H, C (the robot may be prevented from walking around by a wall in any case)

*Test (3)*



# State disambiguation: solutions

- Given number of tests robot can do...
- ... robot wants to determine a **contingency plan** for which tests to do...
- ... to **maximize expected utility** (of the best action to take after testing)
- On the right: the optimal plans with time for one or two tests



# The general state-disambiguation problem

- (There are some obvious generalizations of this, all of which are at least as hard as the basic variant.)
- We are given:
  - A set of possible **world states**, with a **prior distribution**
  - A **utility** for every state (how much is it worth to know *for certain* that the world is in that state)
    - Not knowing the state for certain always gives utility 0
  - A set of **queries** (or tests), each with a set of **answers**
    - For each query, for each state, some (at least 1) answers are consistent with that state, others are not; one of the consistent answers is chosen at random
  - A **deadline** giving the maximum number of testing/reasoning steps
- We are asked what the expected utility of the best plan is

# Complexity results

- **Theorem.** STATE-DISAMBIGUATION is NP-hard even when there is only **one** consistent answer for each state, for each query.
  - Proof reduces from SET-COVER
- **Theorem.** STATE-DISAMBIGUATION is PSPACE-hard in general.
  - Proof reduces from STOCHASTIC SATISFIABILITY

What are the implications of these results for metareasoning?

# Managing time...

- All the above problems have a **deadline**  $T$  for reasoning
- Time spent on **metareasoning** also counts towards deadline!
  - Likely to take nontrivial amount of time (in the worst case), given complexity results
- **Solution 1: allocate  $B$  time to metareasoning**
  - Search for a solution that requires at most  $T-B$  reasoning time
- **Solution 2 (more general):**
  - Spend  $B_1$  time on metareasoning to find a solution that requires at most  $T-B_1$  time
  - If solution satisfactory, use it;
  - Otherwise, spend  $B_2$  metareasoning time to find a solution that requires at most  $T-B_1-B_2$  time
  - Etc.
- One termination condition: stop when solution gets worse
- Could determine  $B_i$  **dynamically** (meta-metareasoning...)

# Interleaving reasoning and metareasoning

- Above setup assumes metareasoning comes first, reasoning comes second
- Not always sensible
- If we decide reasoning action  $a$  will be first, might as well take it
  - If outcome of  $a$  is “yes” then no need to explore contingency “no”
- However, in general, need to consider contingencies to figure out which reasoning action is best...

# Alternative: costly reasoning

- In above problems, reasoning was limited by **deadline**
- Alternative model: reasoning is unlimited but **costly** (e.g., cost 1 per action)
  - Try to optimally trade off reasoning benefits and costs
- Can affect complexity of problems
  - E.g., PERFORMANCE-PROFILES becomes easy: can now determine optimal reasoning amount **for each task separately**
  - Tasks were connected only by deadline!
  - Lots of open questions...

# Conclusion

- Showed several metareasoning (sub)problems are **computationally hard**
  - Unlikely that any metareasoning system can solve them optimally in all cases
- How can we address this?
  - Metareasoning systems that **avoid** these problems
  - **Approximation algorithms** for metareasoning
  - **Meta-metareasoning...**

**THANK YOU FOR YOUR ATTENTION!**