Planning and Decision Making

See also the lecture notes on planning

Relevance to consciousness of an intelligent agent:

- Just as episodic memory is essential to our self-awareness, so too are our conscious intentions and expectations about our own future – both crucially determine the sense of who we are and what our role is in the world.
- Our intentions and expectations about our own future are shaped by the plans we make.
- Our plans are motivated by goals that we judge to be "rewarding"; e.g., we plan meals, entertainment, socializing, games and sporting activities because they are inherently rewarding; likewise, we plan to acquire possessions like vehicles, home furnishings, utensils, etc., because they are *means* to rewards -- bringing us security (mobility, shelter, food preparation, social success, etc.)
- Conscious planning is a symbolic activity and as such potentially related to language (though there's debate about that relationship); *collaborative planning* is at least *mediated* by language.

Planning (like language and learning) is quintessentially human

- Pinker: That's why we study lions, rather than the other way around
- Early human example – tracking an antelope, predicting where it is, using spears, etc.

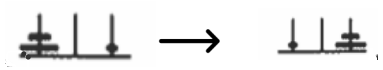
How can planning *work*?

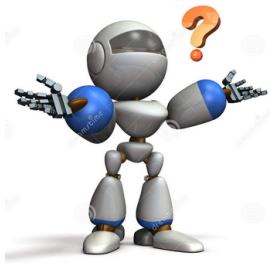
- low-level planning: step-by-step navigation via senses (cf., Prof. Tom Howard's work; Boston Dynamics)
- Specialized planning: route finding, chess, Go, air traffic management, … (see McDermott, ch. 2)
- No general, knowledge-based, multi-domain, deliberate planning yet

Closed-world planning (for arbitrary domain)

Focus of AI planning over many decades

- Initial state: a set of formulas (Predicates with constant arguments)
- Goal state: formulas specifying desired future state
- Operators: can be applied when their *precondition formulas* are true, and specify what formulas become true and which ones become false
- Search for a sequence of operator instances (a plan) that transform the initial state into a state where the goal formulas are true

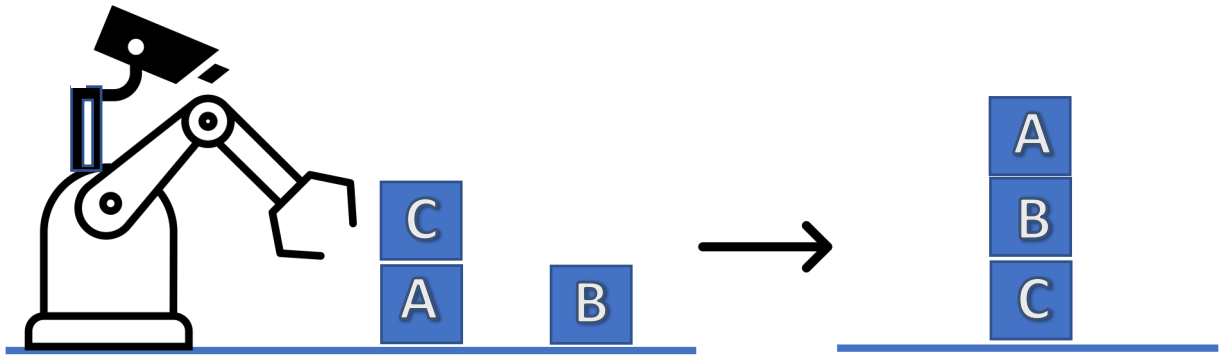




Types of planning

- Deductive planning: prove that a successful plan exists, "extract" plan from proof
- State-space planning: Find a path from the current state to the goals state in an implicit graph representing to actions and states of the world; good for simple puzzles.
- Regression planning: Work backward from the desired goal, to see what steps are needed to make the goal formulas true
- Forward planning: Search forward from a given state, trying actions that seem to get us closer to the goal state
- SAT planning: write down (as Boolean formulas) the constraints which the steps and states of a successful plan must satisfy, then find a satisfying instance of the constraint formulas
- Hierarchical planning: Plan in terms of "big steps" first, then plan the smaller steps that achieve the big steps (e.g., you plan to get a degree, then get a good job – this requires many substeps!)
- Planning under uncertainty and a changing world: Find a plan that's *likely* to succeed, given the uncertainties of action outcomes and of the world; *conditional planning* for various contingencies
- Adversarial planning: anticipating actions of opponent(s) in forming your own plan;
- Collaborative planning: Two or more agents planning & acting to achieve joint goals (cf. work by Prof. J.F. Allen)

Example "stack" & "unstack" operators from Blocks World



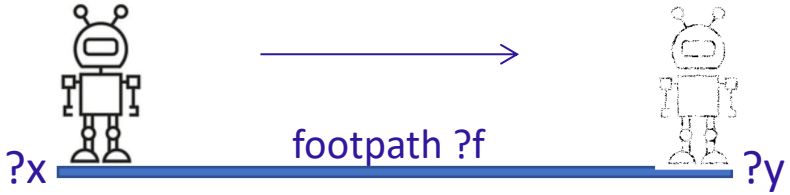

```
(defop stack (?x ?y ?z)
  ; take block ?x from (table or block) ?y and put it on block ?z
  :preconds ((block ?x)  (block ?z)
             (not (?x = ?z))  (not (?y = ?z))
             (clear ?x) ; nothing on ?x
             (on ?x ?y) ; ?x starts out on ?y
             (clear ?z))
  :effects ((on ?x ?z)
            (not (on ?x ?y))
            (not (clear ?z)); ?z now has ?x on it
            (clear ?y)); y is now clear (NB: regard table1 as always clear)
 )
```

```
(defop unstack (?x ?y)
  ; take block ?x off ?y and put it on the table
  :preconds ((block ?x)
             (block ?y)
             (clear ?x) ; nothing on ?x
             (on ?x ?y)); ?x starts out on ?y
  :effects ((on ?x table1)
            (not (on ?x ?y))
            (clear ?y))
 )
```

Example "walk" operator from Gridworld

- ME ("Motivated Explorer") refers to the agent itself ;
- Header syntax here slightly simplified;
- Syntax allows for computable functions like '*', '+', …, and lisp functions signalled by their final exclamation mark, (e.g., 'dist-in-miles!').

```
(defop walk (?x ?y ?f)
  :preconds ((is_at ME ?x)
             (is_tired_to_degree ME 0)
             (is_a_footpath_from+to ?f ?x ?y))
  :effects ((is_at ME ?y)
            (not (is_at ME ?x))
            (is_tired_to_degree ME (* 0.3 (dist-in-miles! ?x ?y)))
            (not (is_tired_to_degree ME 0)))
  :time-required (* 100 (dist-in-miles! ?x ?y))
  :value 0  )
```



Time required & degree of fatigue: computed as function of distance

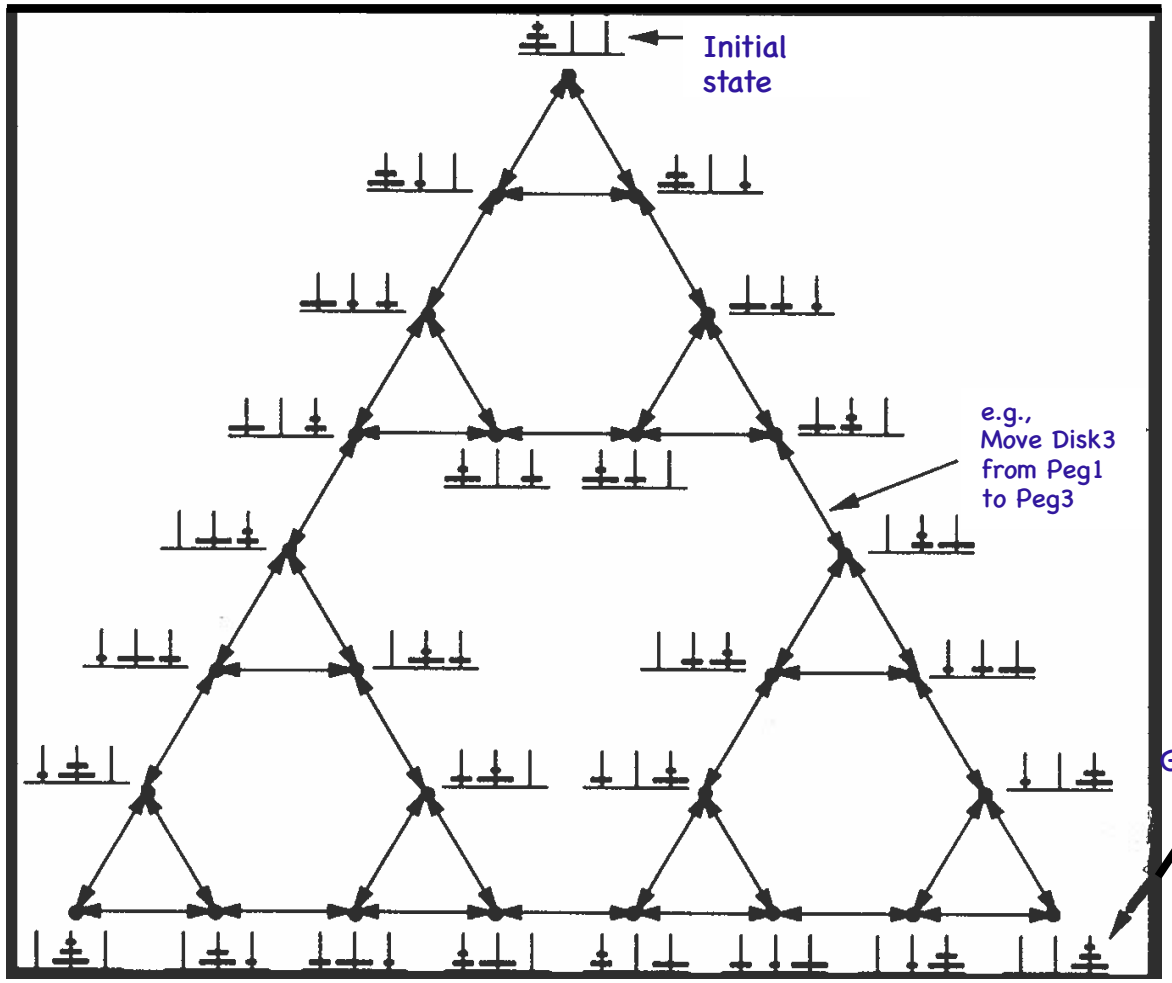
"To walk from ?x to ?y, ME must not be tired and there must be a footpath from ?x to ?y; as a result I will be at ?y (not at ?x), and will be tired to some degree."

Predicate names are chosen to make translation into English easy. E.g.,

```
(is_a_footpath_from+to Maple-Road Home School)
```

becomes *"Maple-Road is a footpath from Home to School"*
('+' is used for argument insertion points).

Towers of Hanoi: State Space



We could use symbols:
((D3 D2 D1) ( ) ( )),
((D3 D2) ( ) (D1)),
((D3) (D2) (D1)), etc.

Towers of Hanoi: STRIPS formulation

Here is one possible formulation of the initial state, goal state, and operators:

Initial state: (Disk D1) (Disk D2) (Disk D3) (Peg P1) (Peg P2) (Peg P3)
(Smaller D1 D2) (Smaller D2 D3) (Smaller D1 D3)
(On D3 P1) (On D2 D3) (On D1 D2)
(Clear D1) (Clear P2) (Clear P3)

Goal state: (On D3 P3) (On D2 D3) (On D1 D2)
; NB: Incomplete, compared to a state-space representation

```
(defop move-to-disk (?x ?y ?z)
  :preconds (Disk ?x) (On ?x ?y) (Clear ?x) (Clear ?z) (Smaller ?x ?z)
  :effects (On ?x ?z)  (Clear ?y) (not (Clear ?z))

(defop move-to-peg (?x ?y ?z)
  :preconds (Disk ?x) (On ?x ?y) (Clear ?x) (Clear ?z) (Peg ?z)
  :effects (On ?x ?z) (not (On ?x ?y)) (Clear ?y) (not (Clear ?z))
```

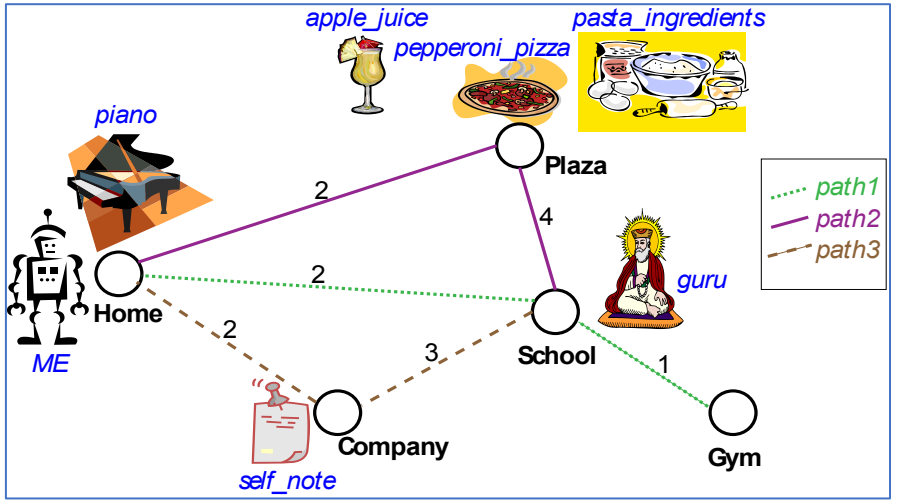
Note:
Under a Closed World Assumption, statements like (Peg D1), (Disk P3), (Smaller D1 P1), are false "by omission" from the state description.

When we implement a negative effect like (not (On ?x ?y)), we don't actually assert this, rather, we delete (On ?x ?y) from the state description.

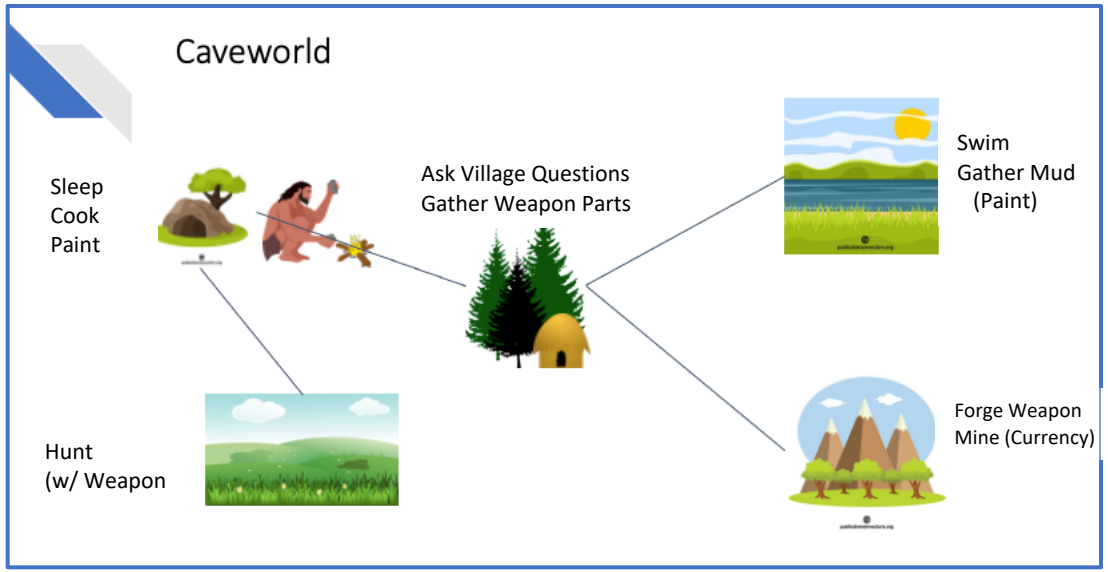
Planning & decision making by a Gridworld agent

Two examples of Gridworlds

Daphne Liu



Clay Emmel



(is_at ME Home)
(is_thirsty_to_degree ME 5)
(is_tired_to_degree ME 2)
(is_at Juice3 Grotto) (Potable Juice3)
(is_a_footpath_from+to Shady-Lane Home Grotto)

World state

(sleep)
Value 2

(walk Shady-Lane Home Grotto)
Value 0

(is_tired_to_degree ME 0)

(is_at ME Grotto)

(take Juice3)
Value 0

Expected cumulative Value = 2

(have ME Juice2)

Expected cumulative Value = 5

(drink Juice3)
Value 5

Question: *Could such a value-seeking agent considered to have motivation & free will?*

(is_thirsty_to_degree ME 0)