Artificial Intelligence

CSC 665

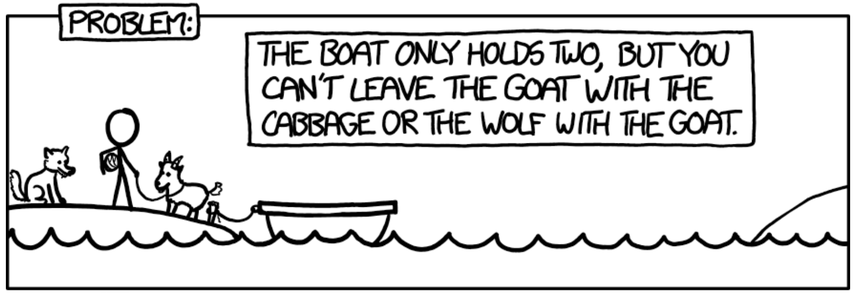
tyler dae devlin

Search I

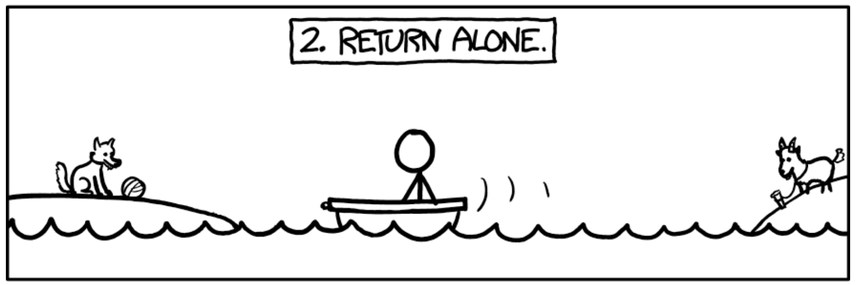
8.24.2023

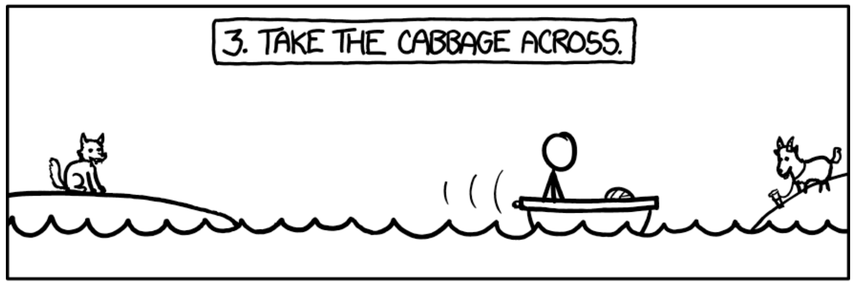
- **Search:** make decisions by looking ahead
- **Logic:** deduce new facts from existing facts
- **Constraints:** find a way to satisfy a given specification
- **Probability:** reason quantitatively about uncertainty
- **Learning:** make future predictions from past observations

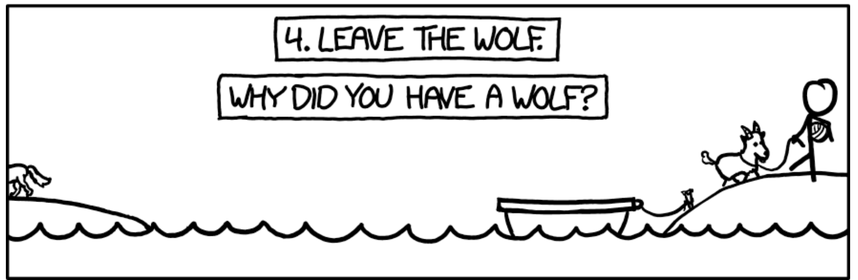
Goat example


PROBLEM:
THE BOAT ONLY HOLDS TWO, BUT YOU CAN'T LEAVE THE GOAT WITH THE CABBAGE OR THE WOLF WITH THE GOAT.

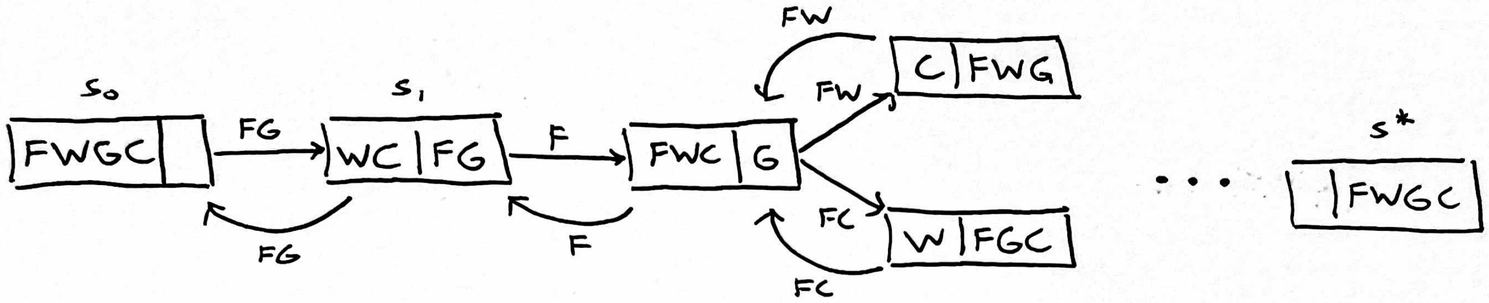

SOLUTION:
1. TAKE THE GOAT ACROSS.


2. RETURN ALONE.


3. TAKE THE CABBAGE ACROSS.


4. LEAVE THE WOLF.
WHY DID YOU HAVE A WOLF?

[search tree for goat problem on board]



$\text{Actions}(s_0) = \{FG\}$

$\text{Actions}(s_1) = \{F, FG\}$

$\text{Succ}(s_0, FG) = s_1 = WC|FG$

$\text{Succ}(s_1, FG) = s_0 = FWGC|$

$\text{Succ}(s_1, F) = FWC|G$

$\text{Cost}(s,a) = 1 \qquad \forall s, a$

$$\text{IsEnd}(s) = \begin{cases} \text{True} & \text{if } s = s^* \\ \text{False} & \text{otherwise} \end{cases}$$

Modeling

Modeling a search problem

Start state: s_0

Possible actions: Actions(s)

Action cost: Cost(s, a)

Transition model: Succ(s, a)

Goal test: IsEnd(s)

Modeling a search problem

Start state: $s_0 \in S$

Possible actions: $\text{Actions}(s) \subseteq A$

Action cost: $\text{Cost}(s, a) \in \mathbb{R}_{\geq 0}$

Transition model: $\text{Succ}(s, a) \in S$

Goal test: $\text{IsEnd}(s) \in \{\text{True}, \text{False}\}$

state space S, action set A, non-negative real numbers $\mathbb{R}_{\geq 0}$

Search graph

- The functions of the search problem induce a **graph**
- **Nodes** are states in S
- There is a **directed edge** $s \to s'$ if $\text{Succ}(s, a) = s'$ for some action $a \in \text{Actions}(s)$
- **Edges are labeled** with the costs given by $\text{Cost}(s, a)$
- **Goal nodes** are determined by $\text{IsEnd}(s)$

[live coding: modeling search]

Inference

Backtracking search

Global state: minimum cost path, set of explored nodes

function search(s, path) :

- **if** IsEnd(s) :
  - update the minimum cost path
- **for each** action $a \in$ Actions(s) :
  - **if** Succ(s, a) hasn't been explored yet:
    - add it to the explored set
    - extend path with Succ(s, a) and Cost(s, a)
    - recurse: search(Succ(s, a), path)

Separating modeling from inference means we only have to write the inference code once.

[live coding: backtracking search]