Intelligent Game Agents

CMSC425.01 Spring 2019

Administrivia

- Exam being graded …
- Project 2b concepts out, write up soon (add animations to 2a)

Today’s questions

How, and why, should you
make game agents intelligent

Thoughts on game AI

- What is game AI for?

Thoughts on game AI

- What is game AI for?

  Major game opponents

  Individual game units

  Richer world of NPCs

Discussion question

- Do you want your opponents to be

Game AI,	made better
Humans,	through better networked games

What does AI mean here?

- How code Starcraft

  Hive mind?

  Individual zerg?

What does AI mean here?

- How code Starcraft

  Hive mind?

  Hard coded?

  Not adaptive

  Individual zerg?

  A* plus "attack"

- Observation:
  - Not that intelligent but powerful gameplay

Review: examples

- A*

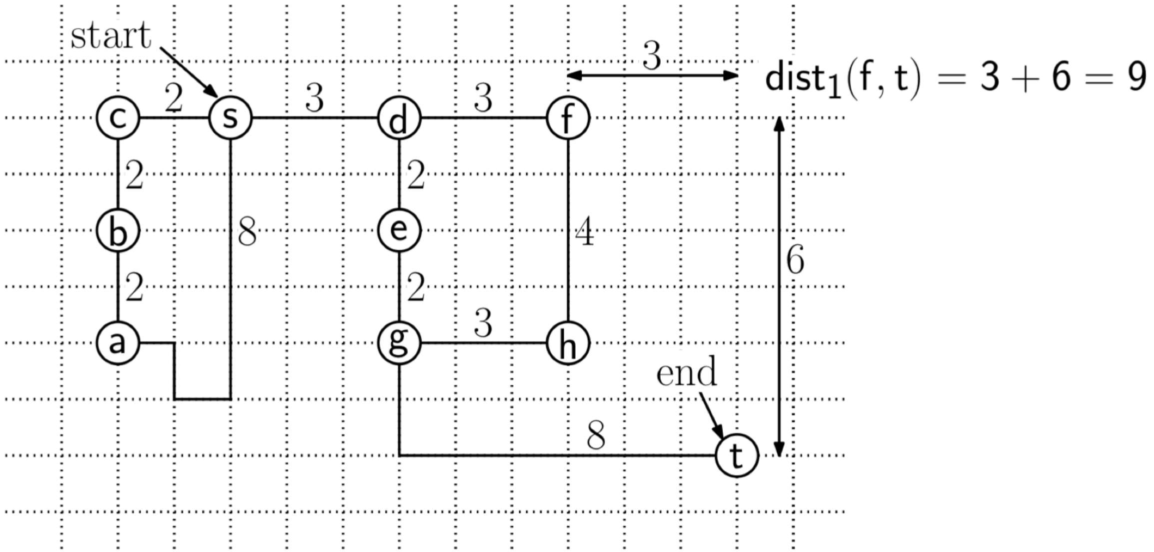
- Minowski sum of obstacles

A*

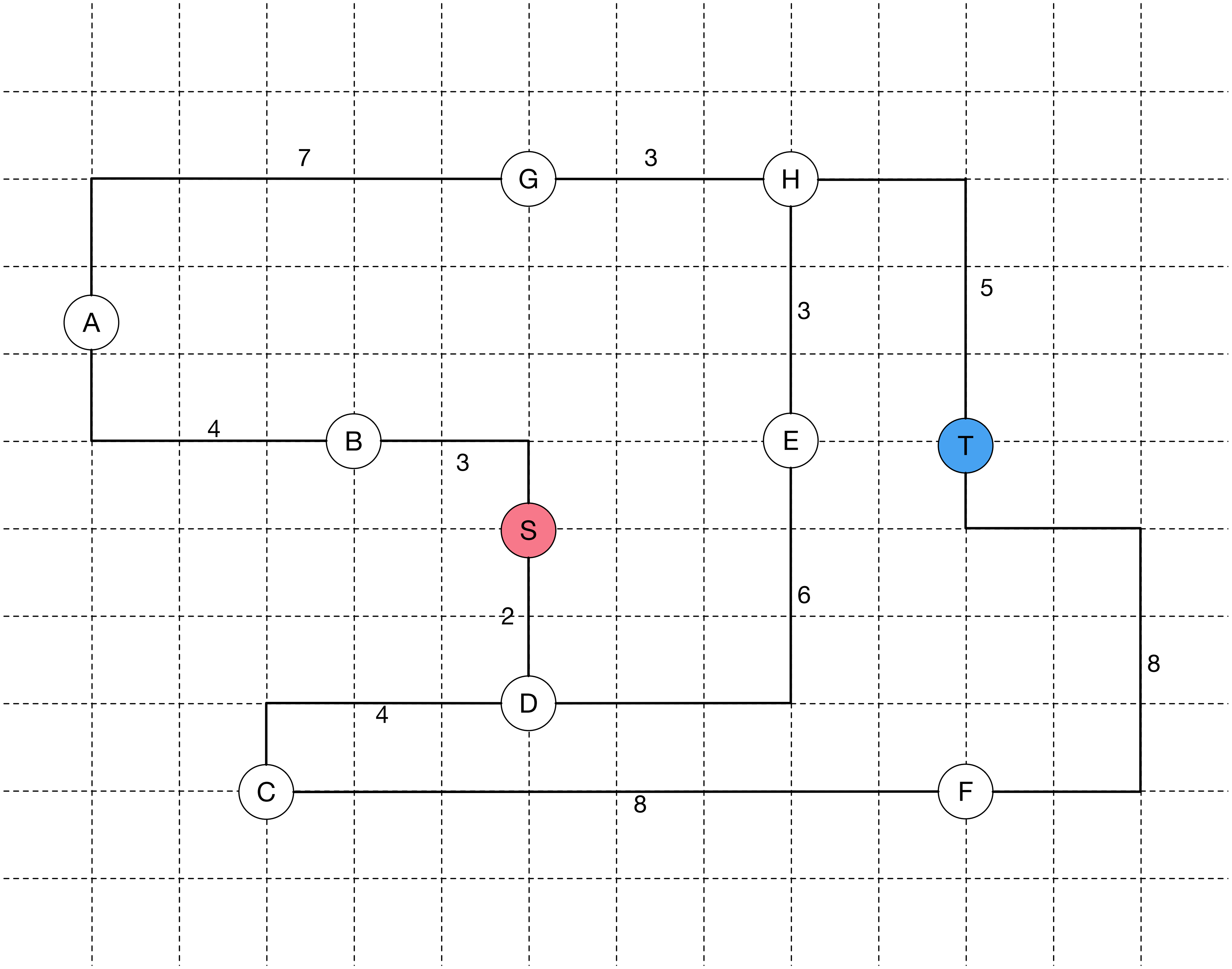
- Pick next node to expand based on sum of distance so far *and* heuristic

$$f(u) = d[u] + h(u) = d[u] + \text{dist}(u, t)$$

```
A-Star(G, s, t) {
  foreach (node u) {                              // initialize
    d[u] = +infinity;  mark u undiscovered
  }
  d[s] = 0;  mark s discovered                    // distance to source is 0
  repeat forever {                                // go until finding t
    let u be the discovered node that minimizes d[u] + dist(u,t)
    if (u == t) return d[t]                       // arrived at the destination
    else {
      for (each unfinished node v adjacent to u) {
        d[v] = min(d[v], d[u] + w(u,v))  // update d[v]
        mark v discovered
      }
      mark u finished                             // we're done with u
  }
}
```



A* Search – Each entry is $d[u] : f(u)$										
Stage	$d[s]$	$d[a]$	$d[b]$	$d[c]$	$d[d]$	$d[e]$	$d[f]$	$d[g]$	$d[h]$	$d[t]$
$h(u)$	15	13	15	17	12	10	9	8	5	0
Init	0:15	∞:13	∞:15	∞:17	∞:12	∞:10	∞:9	∞:8	∞:5	∞:0
1: s	**0**	8:13	–	2:17	3:12	–	–	–	–	–
2: d	↓	8:13	–	2:17	**3**	5:10	6:9	–	–	–
3: e		8:13	–	2:17	↓	**5**	6:9	7:8	–	–
4: f		8:13	–	2:17		↓	**6**	7:8	–	15:0
5: t		8:13	–	2:17			↓	7:8	–	**15**
Final	0	8	∞	2	3	5	6	7	∞	15



7
G
3
H
A
5
3
4
B
E
T
3
S
6
2
8
D
4
C
F
8

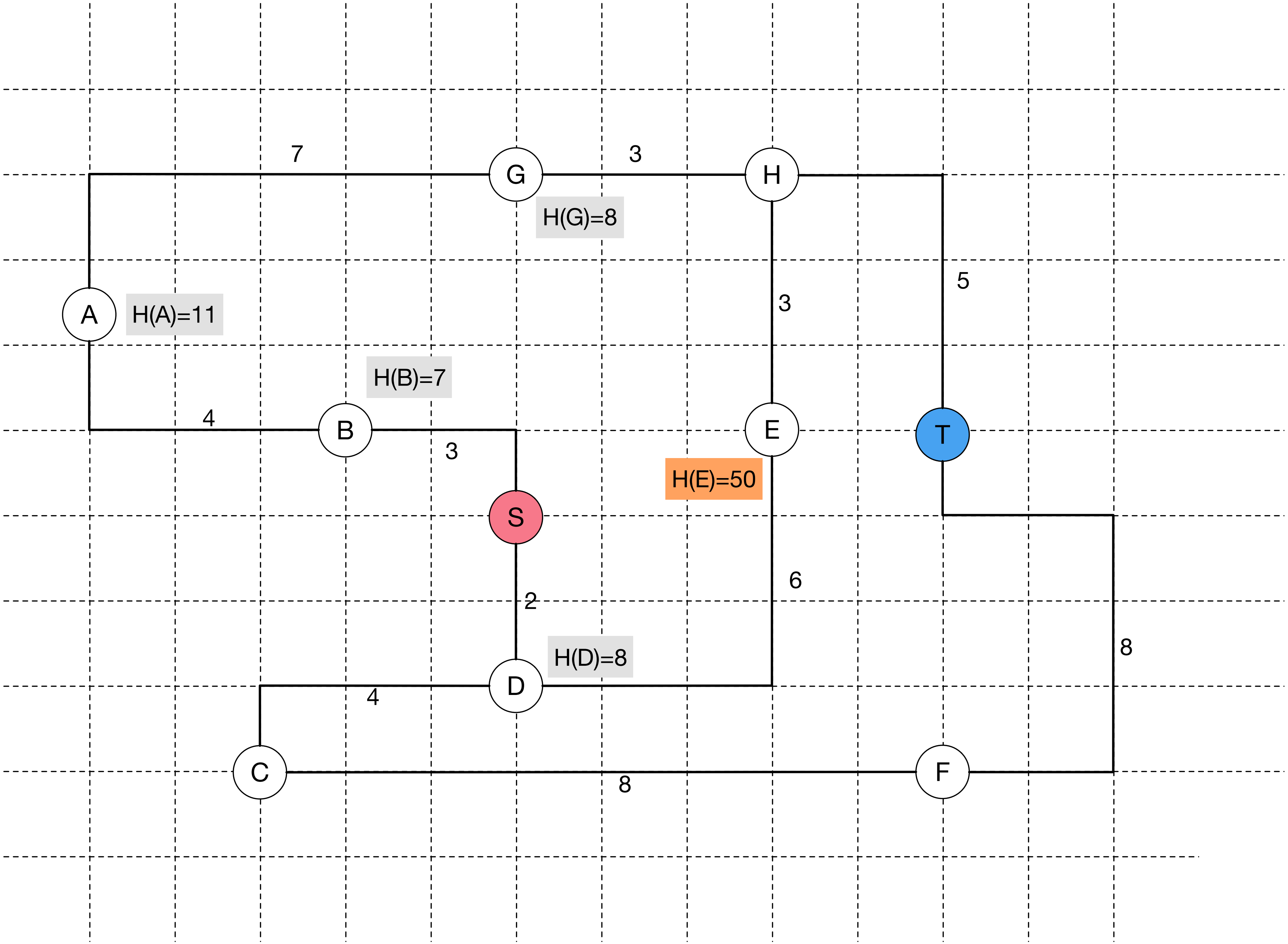
Good heuristics

- For A* to compute correctly the heuristic h(u) must be:

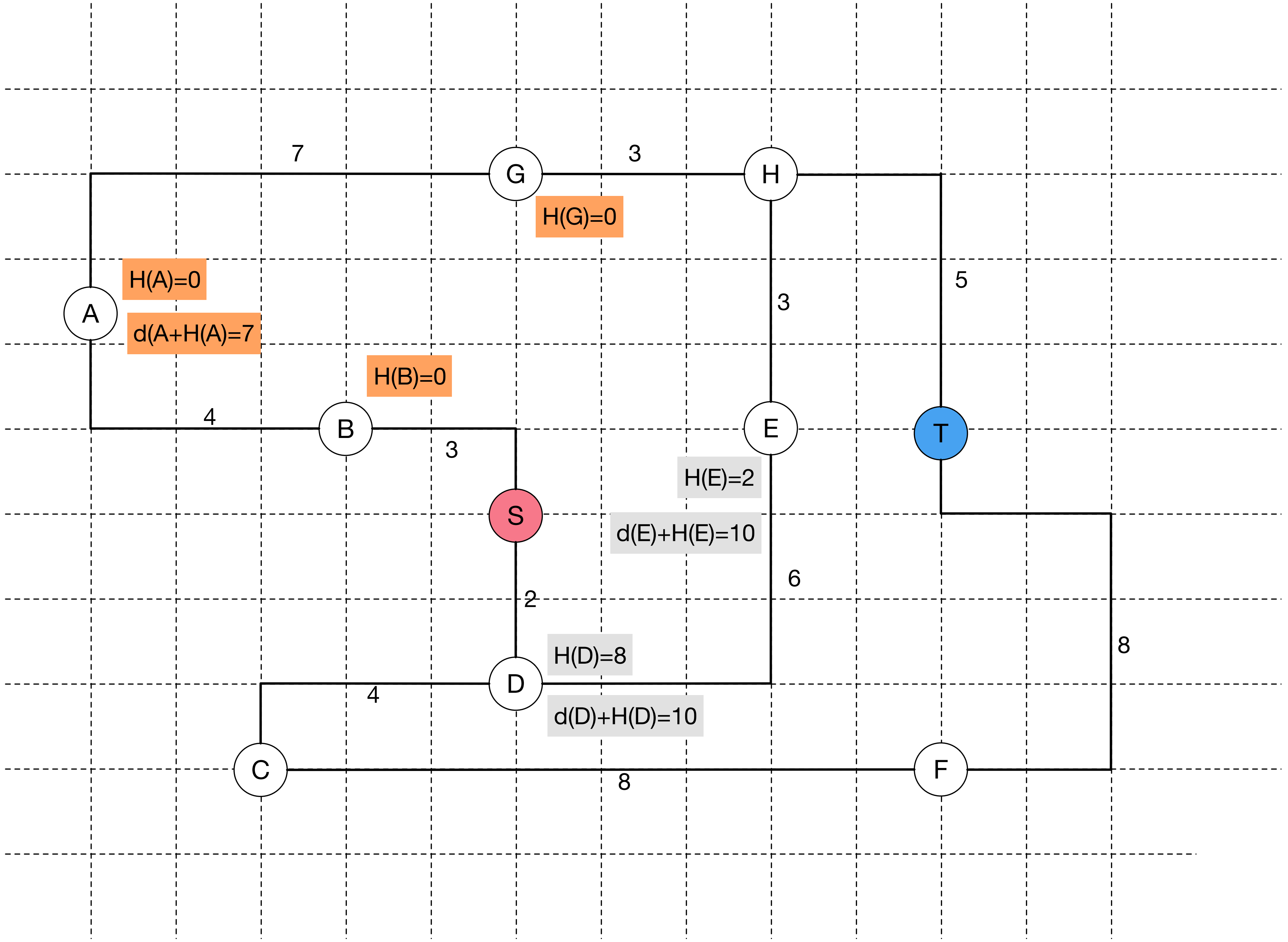
- Admissible: h(u) never overestimates the graph distance from node u to goal t

- Consistent: h(u') <= delta(u',u'') + h(u'')

- *Goldilocks* – heuristics must be not too high, not too low



7
3
G
H
H(G)=8
5
3
A
H(A)=11
H(B)=7
4
B
3
E
T
H(E)=50
S
6
2
8
H(D)=8
D
4
C
8
F

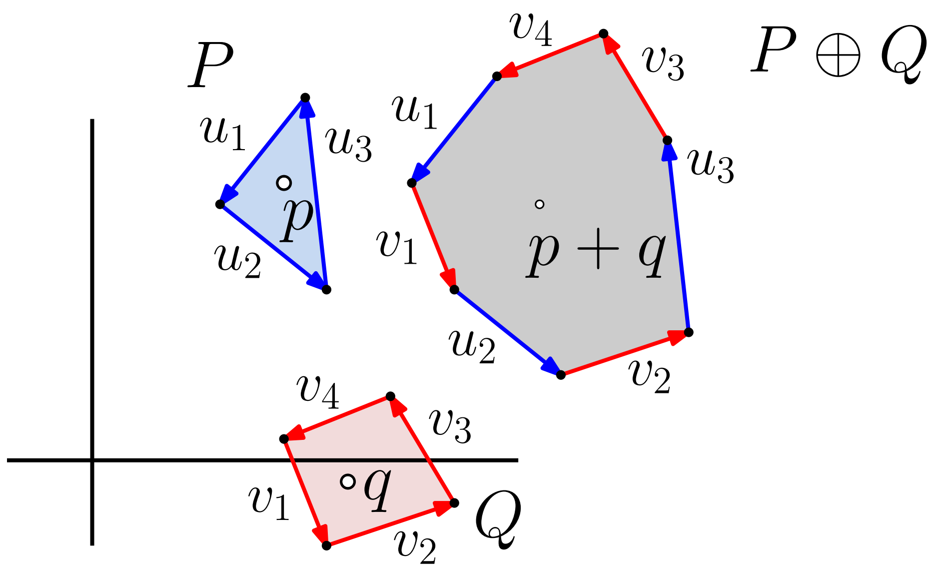


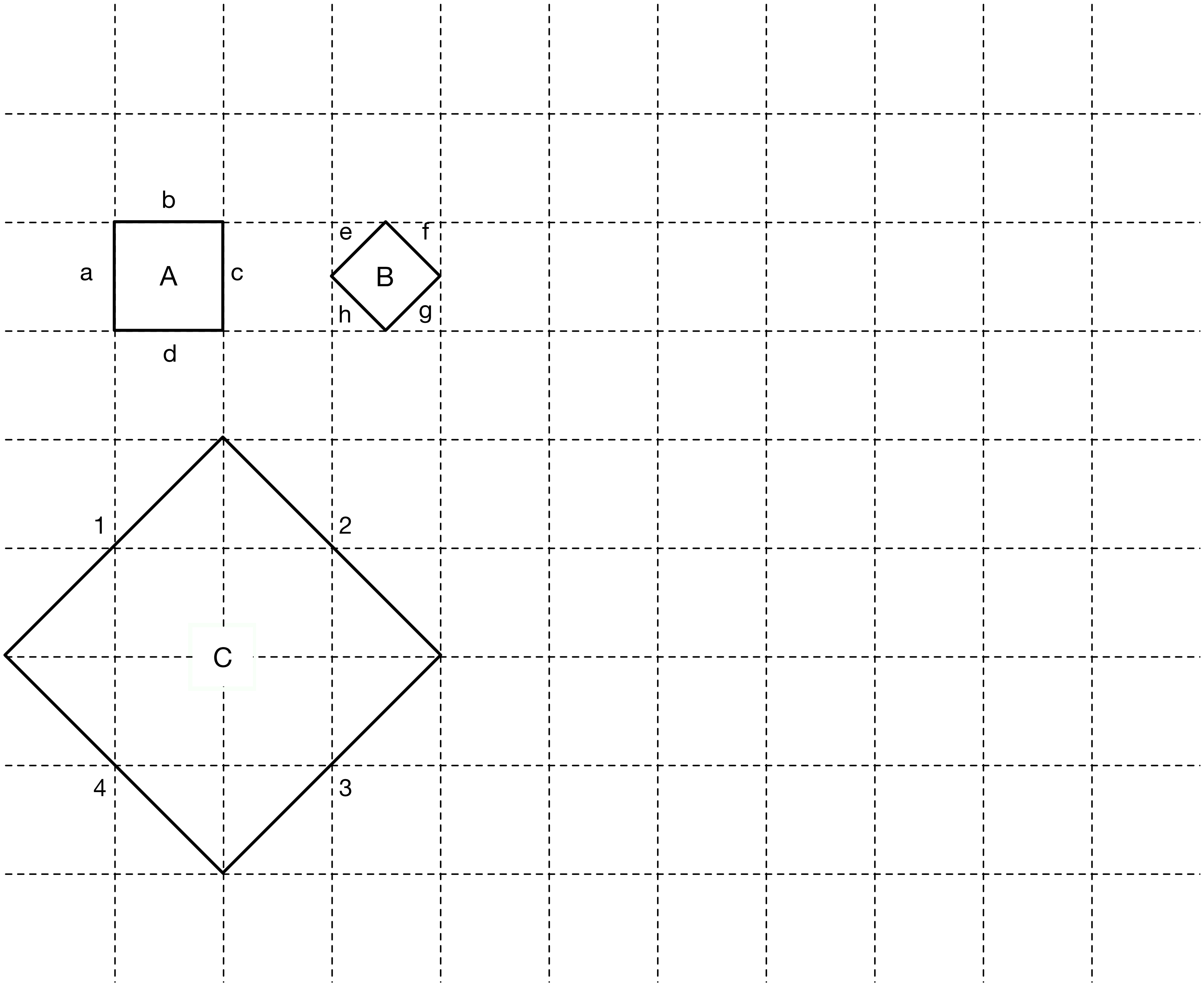
7
G
3
H
H(G)=0
H(A)=0
5
A
3
d(A+H(A)=7
H(B)=0
4
B
E
T
3
H(E)=2
S
d(E)+H(E)=10
6
2
H(D)=8
8
D
4
d(D)+H(D)=10
C
F
8

Algorithm: Computing Minowski sum

- Input: two polygons
- Output: polygon of M-sum

- Algorithm:
  - Take each edge in CCW direction
  - Sort by angle
  - Combine

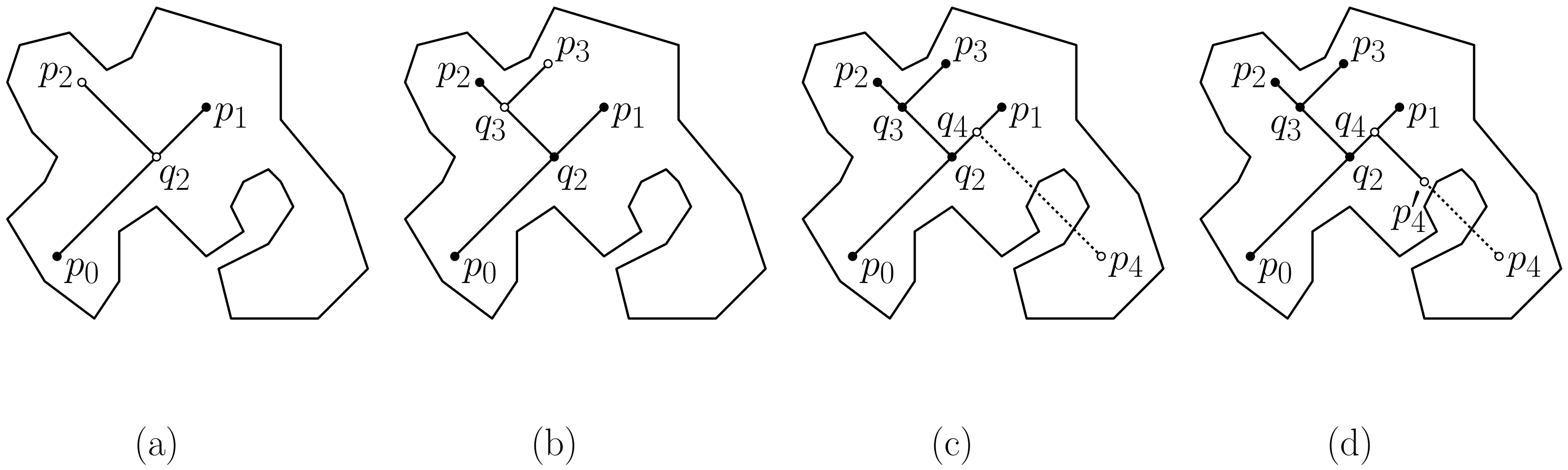



b
a
A
c
d
e
f
B
h
g
1
2
C
4
3

Finding paths in polygonal configuration space

- Version 1: Navmesh
- Others?

- Version 8: Rapidly-expanded Random Trees (RRTs)



Decision making

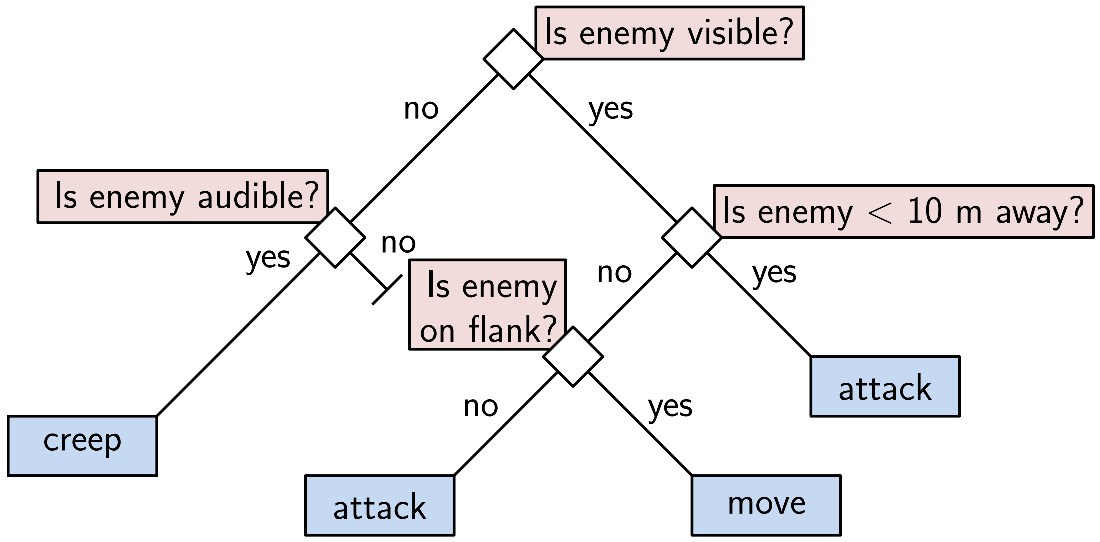
- Reactive decision making: Respond
  - Decision trees
  - Finite State Machines
  - Behavior trees

- Proactive decision making: Plan
  - Not this semester
  - Can use variation of A* on space of operators on world state
  - Robot planning
  - Done by game designers implicitly

Decision trees

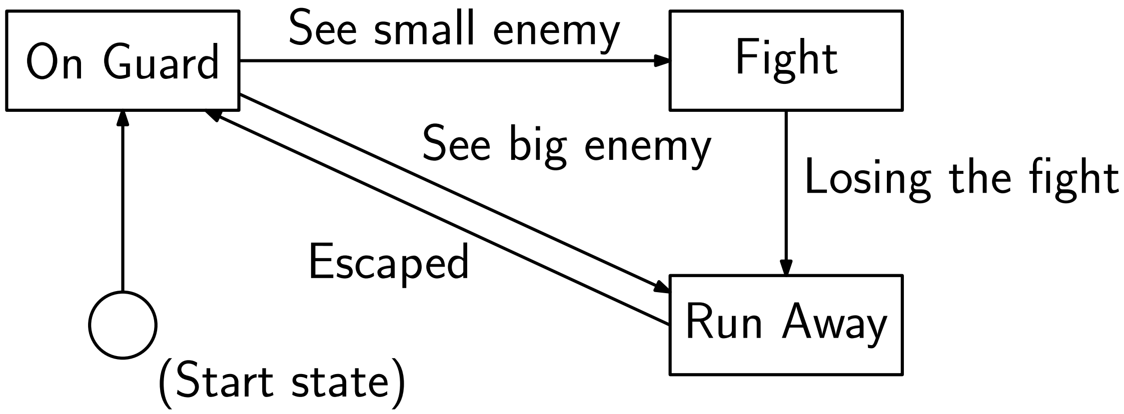
- Structured if

- Can
  - Randomize
  - Share subtrees
  - Have branching factor > 2



Finite State Machine

- Organize behaviors in graph
- Set transitions on state of game



(a)

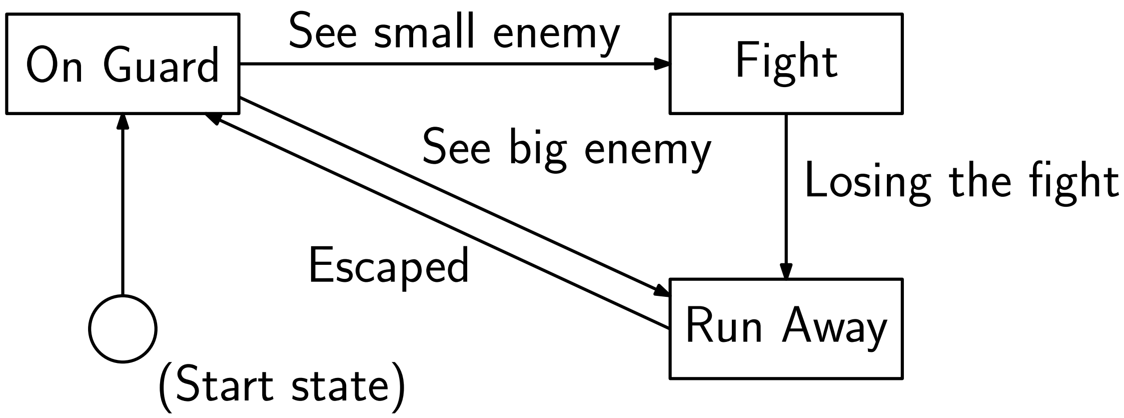
Start

	Small enemy	Big enemy	Losing	Escaped
Guard	Fight	Run	–	–
Fight	–	–	Run	–
Run	–	–	–	Guard

(b)

Finite State Machine

- State can be
  - Behavior - character actions
  - Emotional state - predisposition (confidence/fear, anger, health,energy)

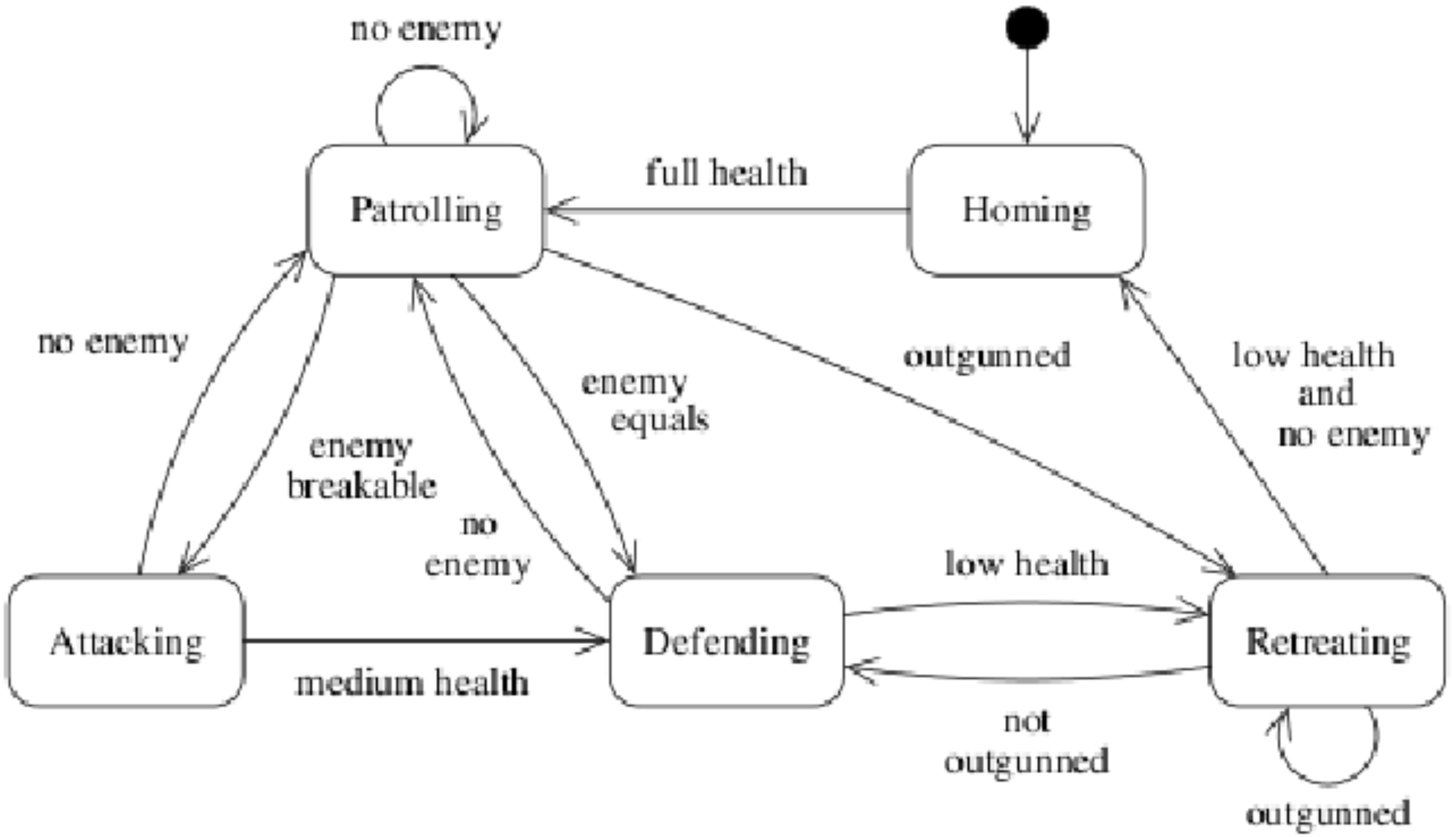


(a)

Start

	Small enemy	Big enemy	Losing	Escaped
Guard	Fight	Run	–	–
Fight	–	–	Run	–
Run	–	–	–	Guard

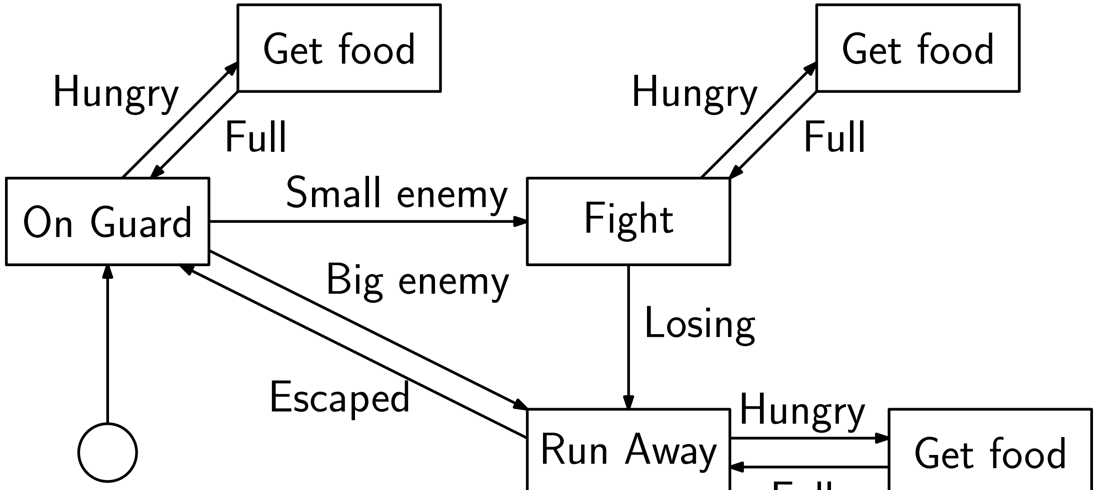
(b)



no enemy
full health
Patrolling
Homing
no enemy
outgunned
low health
and
no enemy
enemy
equals
enemy
breakable
no
enemy
low health
Attacking
Defending
Retreating
medium health
not
outgunned
outgunned

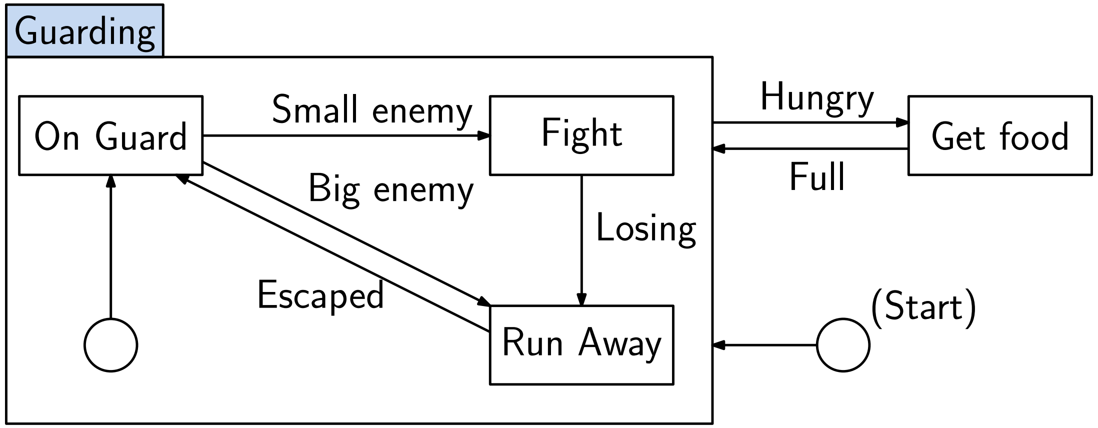
Controlling FSM complexity

- State with branches to many others

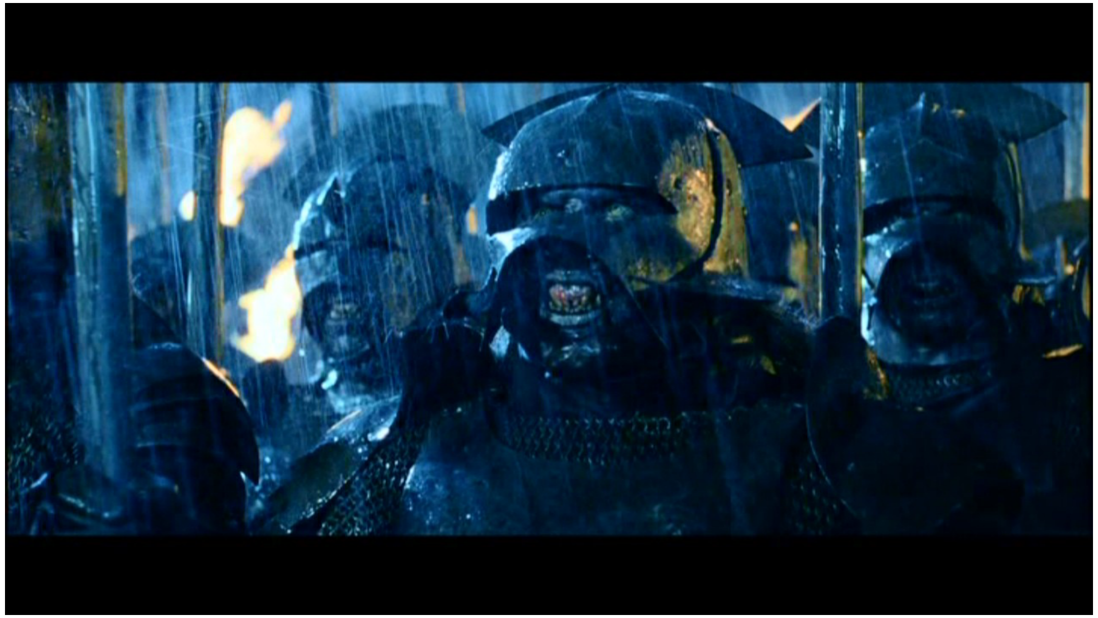


Controlling FSM complexity

- Hierarchical FSM (StateCharts)
  - Superstates + generalized transitions
- Part of UML (if you don't know, look up …)



Tuning FSMs



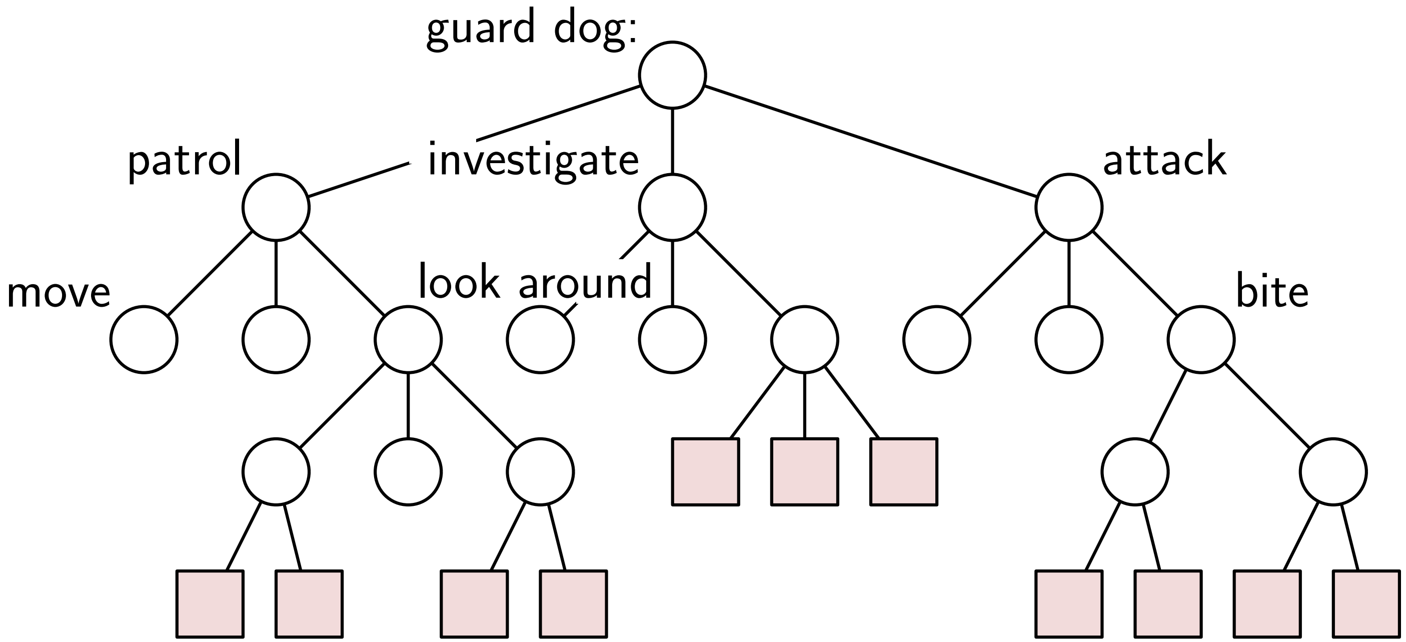
- Variations on one character template
- EG, Orcs in LOTR
- Massive Software
  - Each agent owns character profile
  - Randomize when populating game

Behavior trees

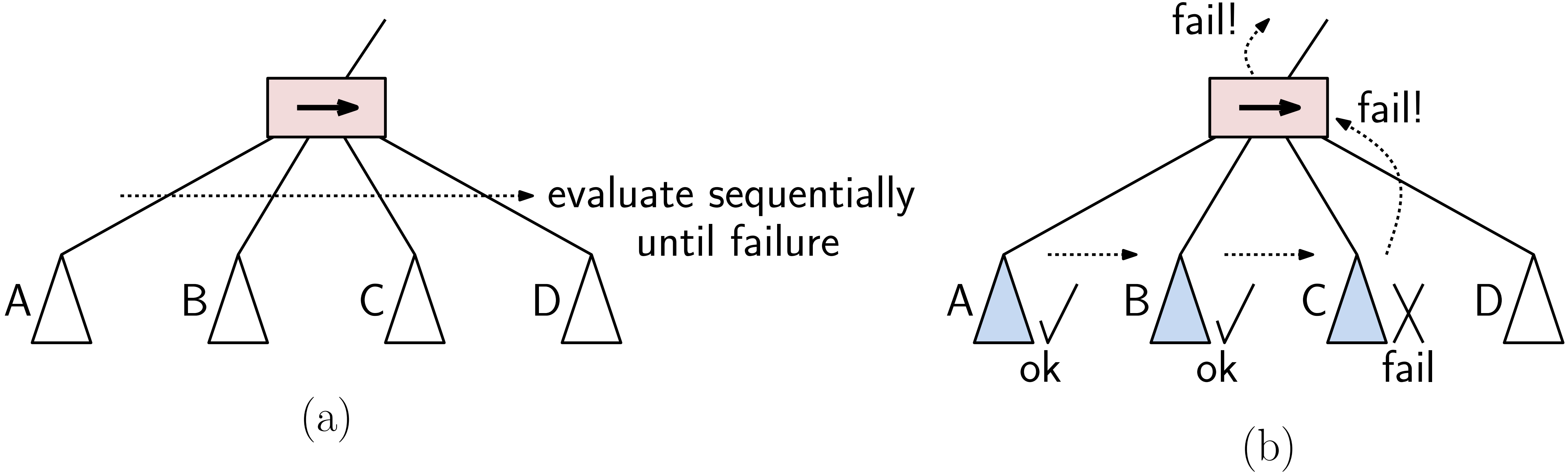
- Lightweight way to design action plans
- Plan
  - Sequence of actions
    1) Go to door
    2) Use key open door
    3) Go through door
  - With preconditions
    2)* Must have key

Behavior trees

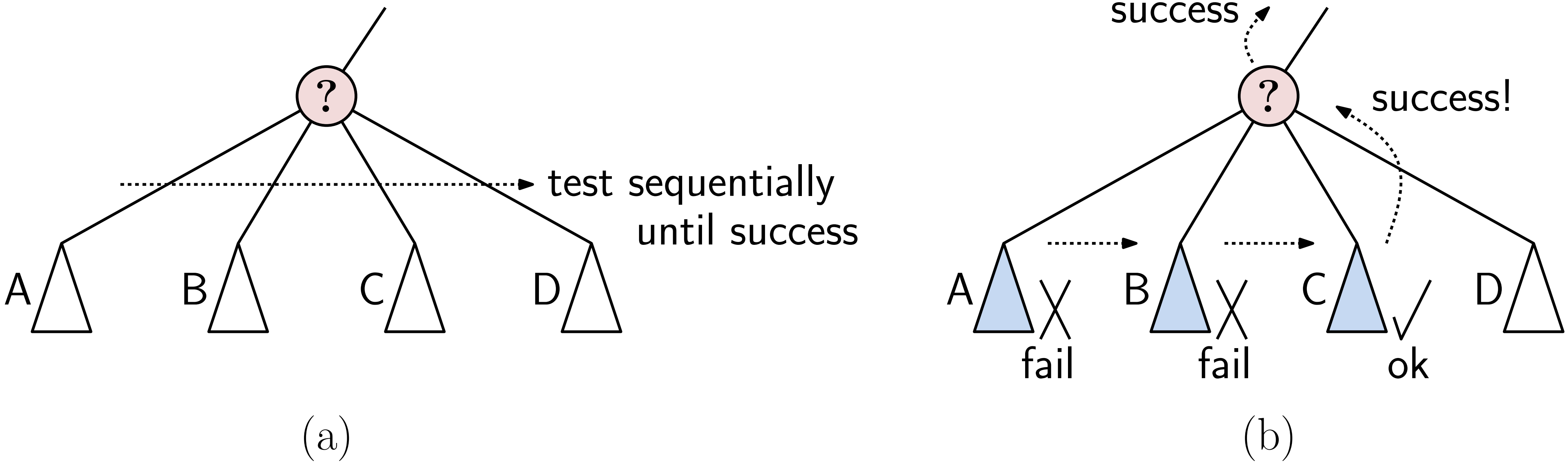
- Lightweight way to design action plans
- Plan
  - Sequence of actions
    1) Go to door
    2) Use key open door
    3) Go through door
  - With preconditions
    2)* Must have key



Sequential AND node



Sequential OR node



Goal: Move into room