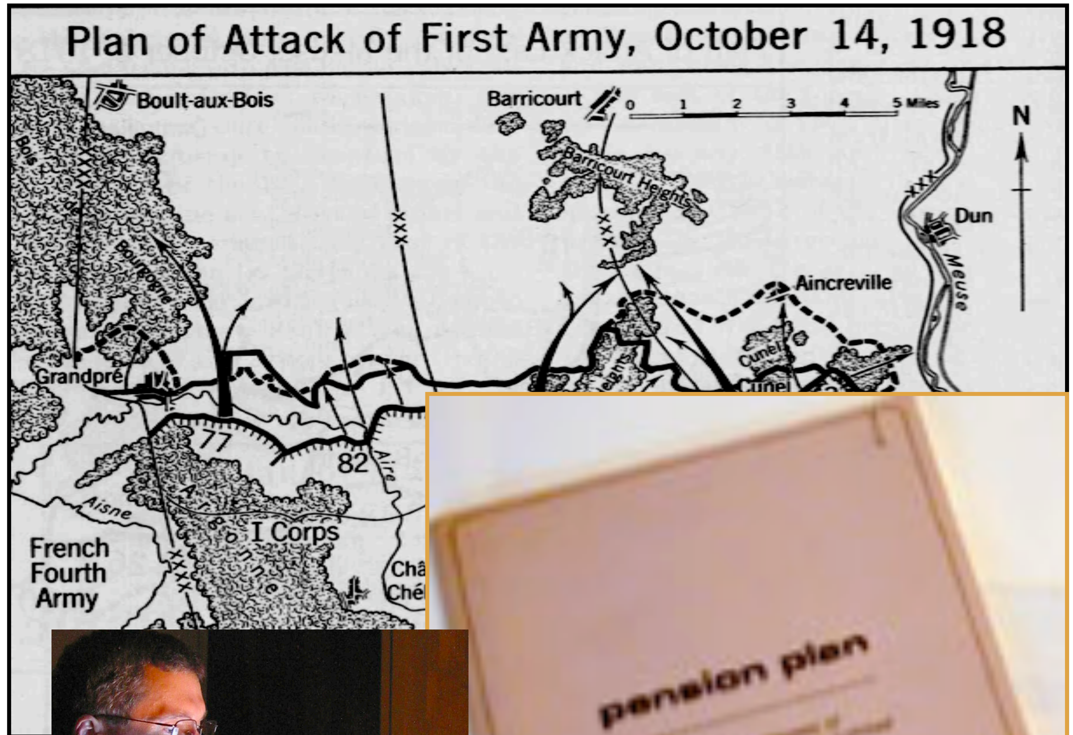
How do you plan if there are other agents and you don't know their plans?

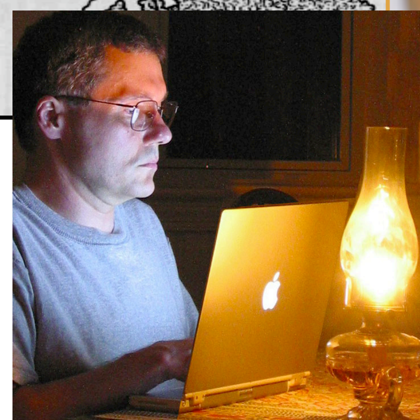
Dana Nau
University of Maryland
College Park, MD

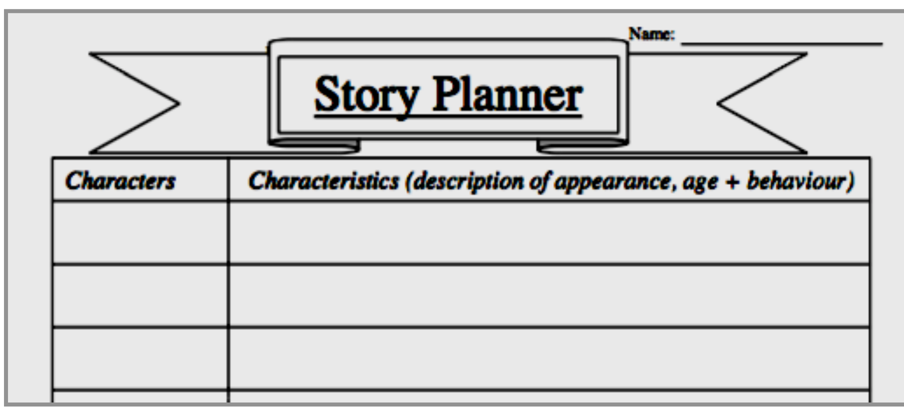
What is a Plan?

plan *n.*

1. A scheme, program, or method worked out beforehand for accomplishing an objective: ***a plan of attack***.
2. A proposed or tentative project: ***I had no plans for the evening***.
3. A program or policy stipulating a service or benefit: ***a pension plan***.
4. A systematic arrangement of elements or important parts: ***a seating plan***; ***the plan of a story***.

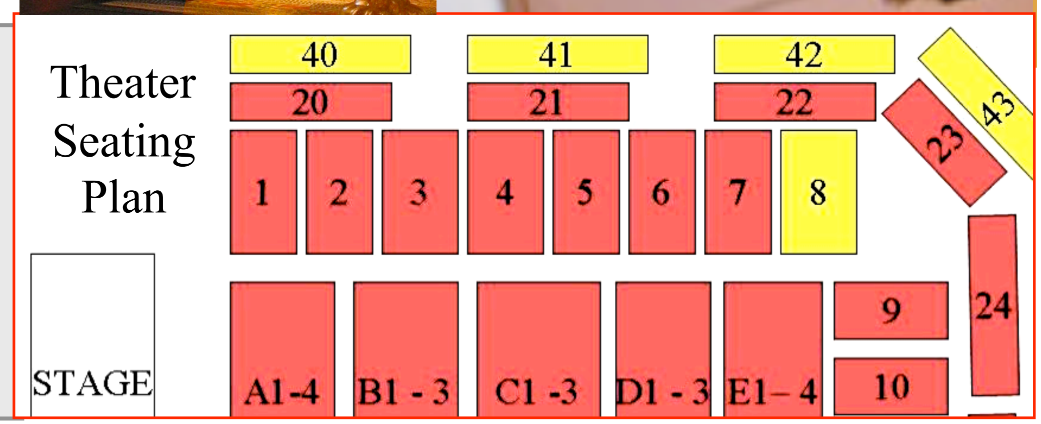





Name:

Story Planner

Characters	Characteristics (description of appearance, age + behaviour)



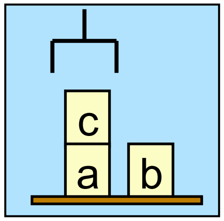
Plans in AI

- AI planning researchers use a more specialized definition:
- [a representation] of future behavior … usually a set of actions, with temporal and other constraints on them, for execution by some agent or agents. – Austin Tate, *MIT Encyclopedia of the Cognitive Sciences*, 1999

```
                              03  Establish datum point at bullseye (0.25, 1.
    B  VMC1  0.10     0.34    01  Install 0.15-diameter side-milling tool
                              02  Rough side-mill pocket at (-0.25, 1.25)
                                  length 0.40, width 0.30, depth 0.50
                              03  Finish side-mill pocket at (-0.25, 1.25)
                                  length 0.40, width 0.30, depth 0.50
                              04  Rough side-mill pocket at (-0.25, 3.00)
                                  length 0.40, width 0.30, depth 0.50
                              05  Finish side-mill pocket at (-0.25, 3.00)
                                  length 0.40, width 0.30, depth 0.50
    C  VMC1  0.10     1.54    01  Install 0.08-diameter end-milling tool
                              [...]
    T  VMC1  2.50     4.87    01  Total time on VMC1
    A   EC1  0.00    32.29    01  Pre-clean board (scrub and wash)
                              02  Dry board in oven at 85 deg. F
    B   EC1 30.00     0.48    01  Setup
                              02  Spread photoresist from 18000 RPM spinner
    C   EC1 30.00     2.00    01  Setup
                              02  Photolithography of photoresist
                                  using phototool in "real.iges"
005 D   EC1 30.00    20.00    01  Setup
                              02  Etching of copper
005 T   EC1 90.00    54.77    01  Total time on EC1
006 A   MC1 30.00     4.57    01  Setup
                              02  Prepare board for soldering
006 B   MC1 30.00     0.29    01  Setup
                              02  Screenprint solder stop on board
006 C   MC1 30.00     7.50    01  Setup
```

Part of a manufacturing process plan

Introduction



- AI planning researchers usually assume there is just one agent
  - » The plan executor
  - » Nothing happens unless the executor makes it happen
- Generalizing to multiple agents
  - » Two cases:

1. Team of agents with a common objective
   - How to do communication, coordination, information-gathering, …

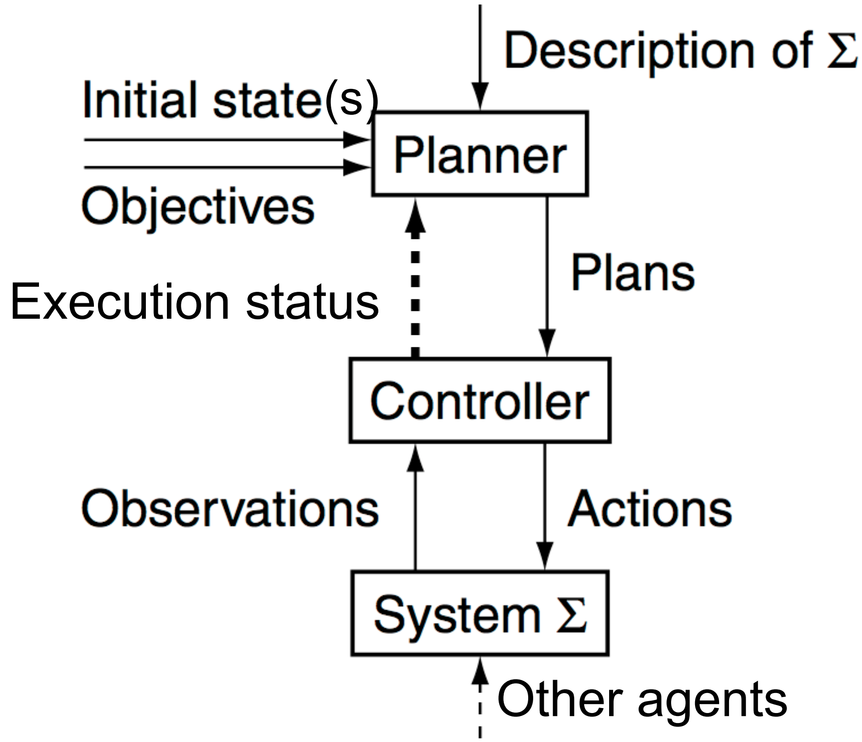
2. Objectives of other may differ from yours
   - How to accomplish your objectives?

   This is the focus of my talk

Outline

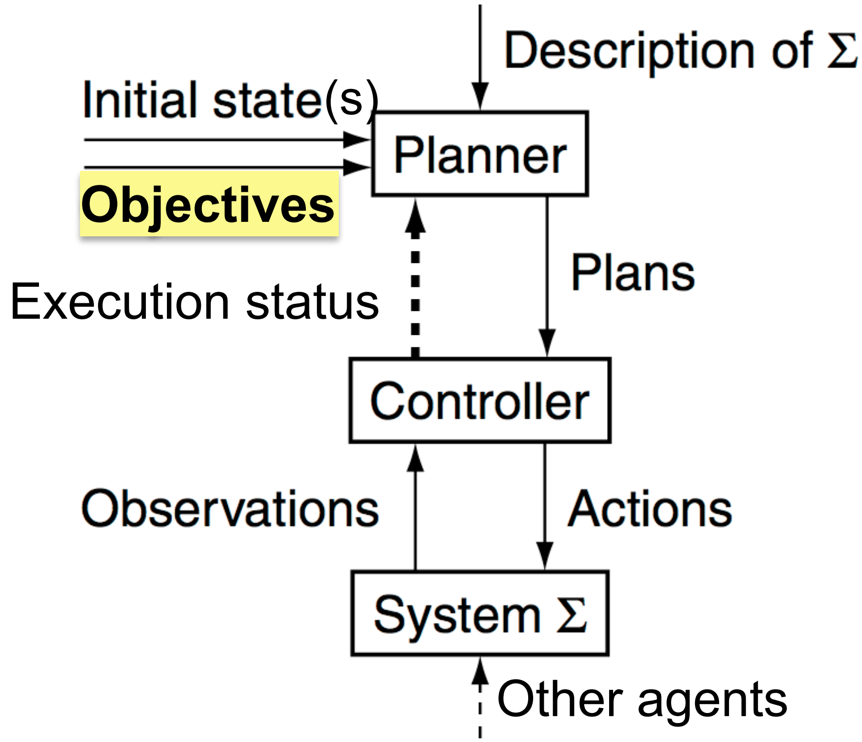
- Abstract model of AI planning
- Classification of multi-agent planning problems
- Issues and techniques
  - Ways to model the other agents
  - Ways to deal with combinatorial explosion
- Open problems, important trends

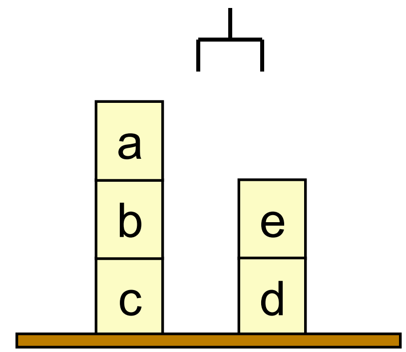
Abstract Model of AI Planning



- Basic ingredients:
  - The planner
  - The plan executor (controller)
  - The environment (system)

Abstract Model of AI Planning

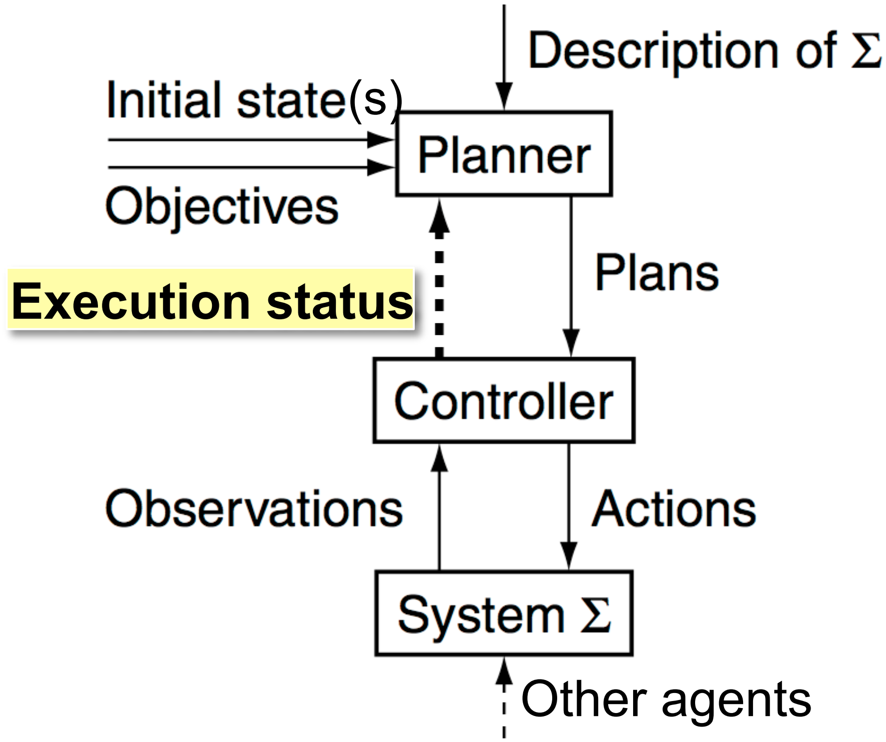




- **Classical goal**: get to any state that satisfies some property, e.g., on(a,b)
- **Utility**: a numeric measure of a state's desirability, e.g., the height of a stack

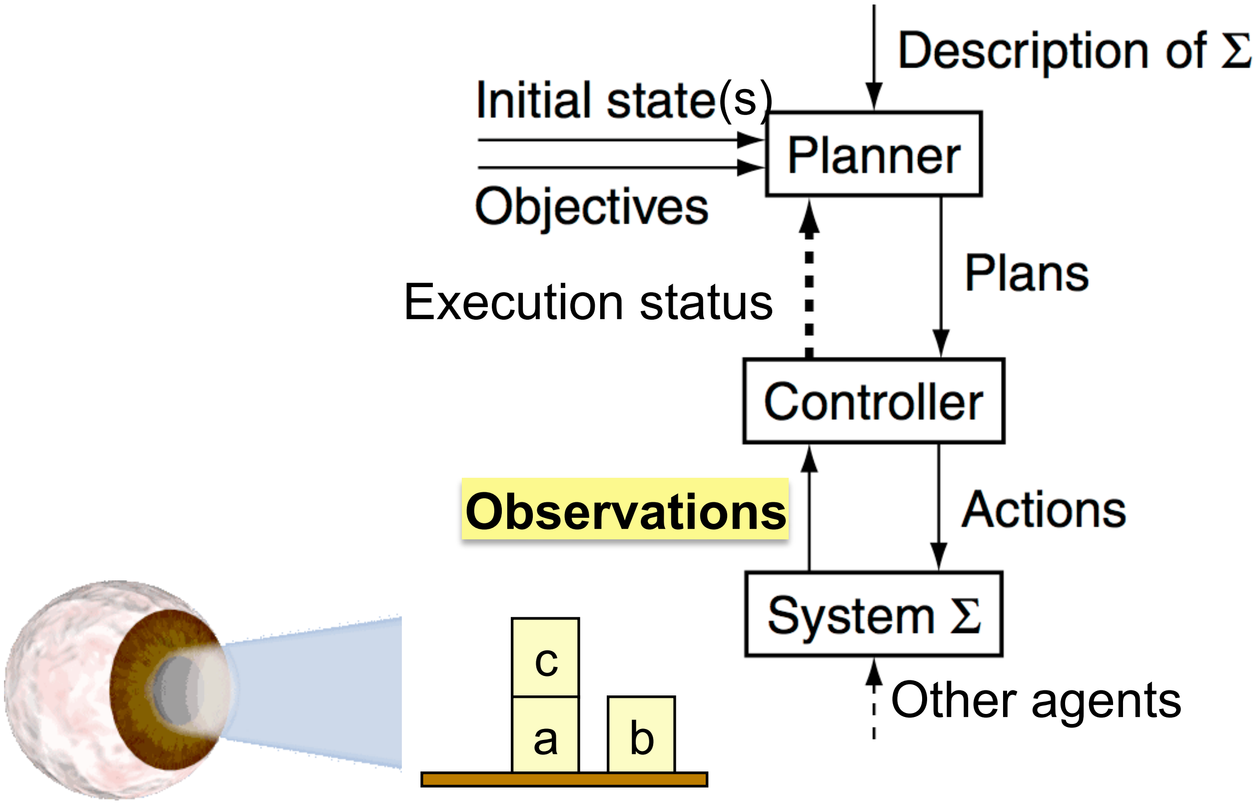
- Some other possibilities (e.g., tasks, extended goals)
  - For now, I'll ignore these

Abstract Model of AI Planning



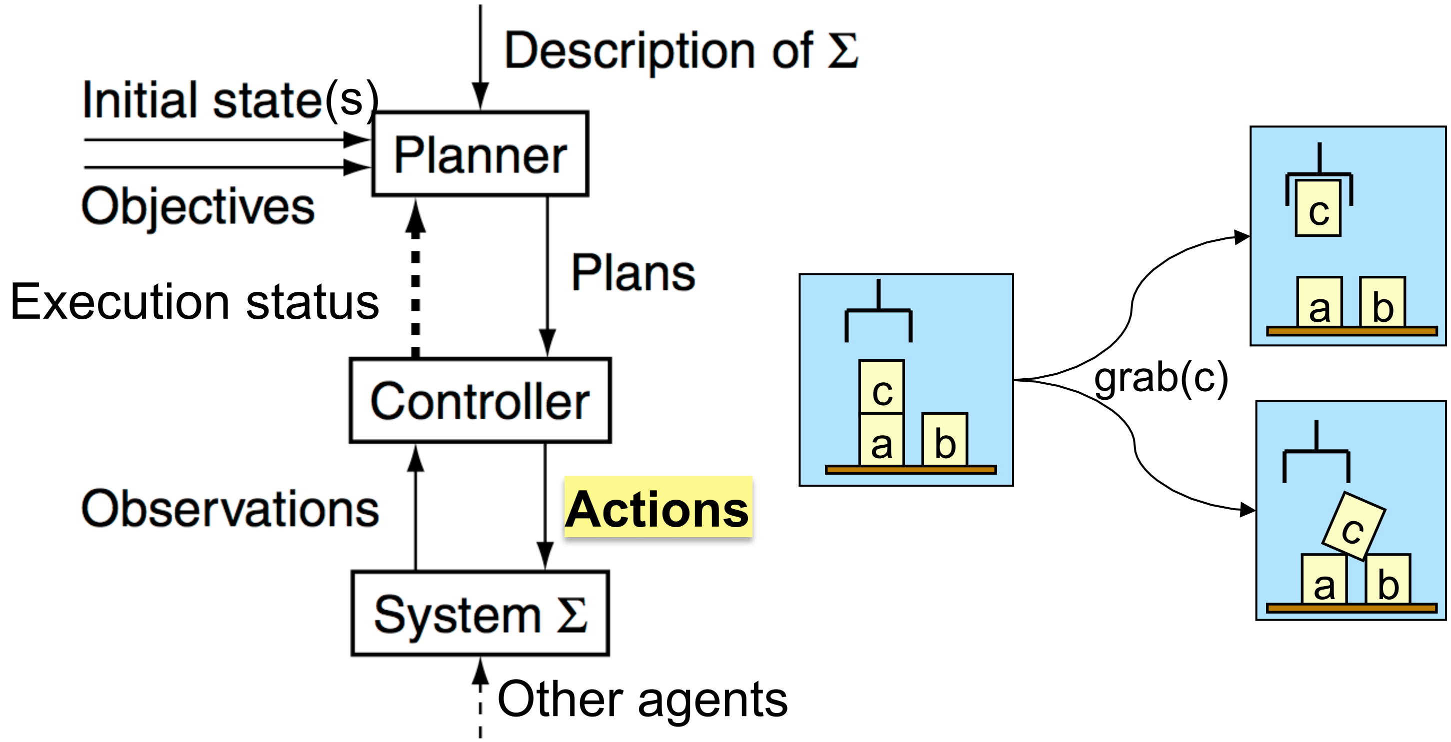
- **Offline**:
  - no feedback from the controller (e.g., classical planning)
- **Online**:
  - planning and execution are interleaved
  - planner gets information about execution status

Abstract Model of AI Planning



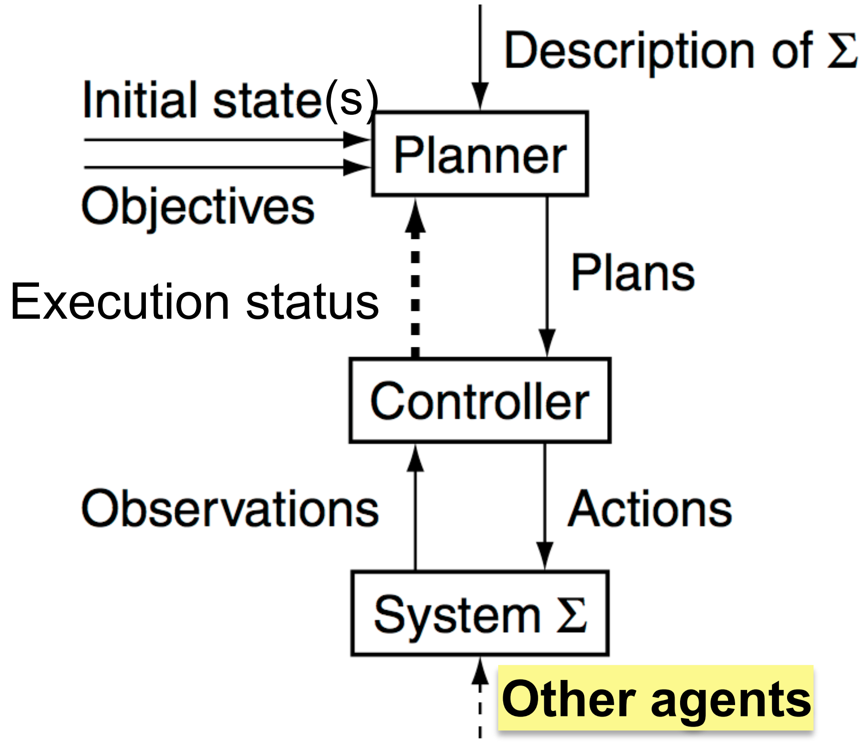
- **Full observability:**
  - » Controller's observations tell it the exact state of the world (e.g., MDP)
- **Partial observability**:
  - » Controller's observations give partial information (e.g., POMDP)

Abstract Model of AI Planning



- **Deterministic**: only one possible outcome (e.g., classical planning)
- **Stochastic**: probability distribution over outcomes (e.g., MDP)
- **Nondeterministic**: multiple possible outcomes, but no probabilities

Abstract Model of AI Planning



- Most AI planning research assumes there are no other agents
  - But some multi-agent problems can be reduced to single-agent planning problems
  - Encode the other agents' actions as outcomes of ***our*** actions

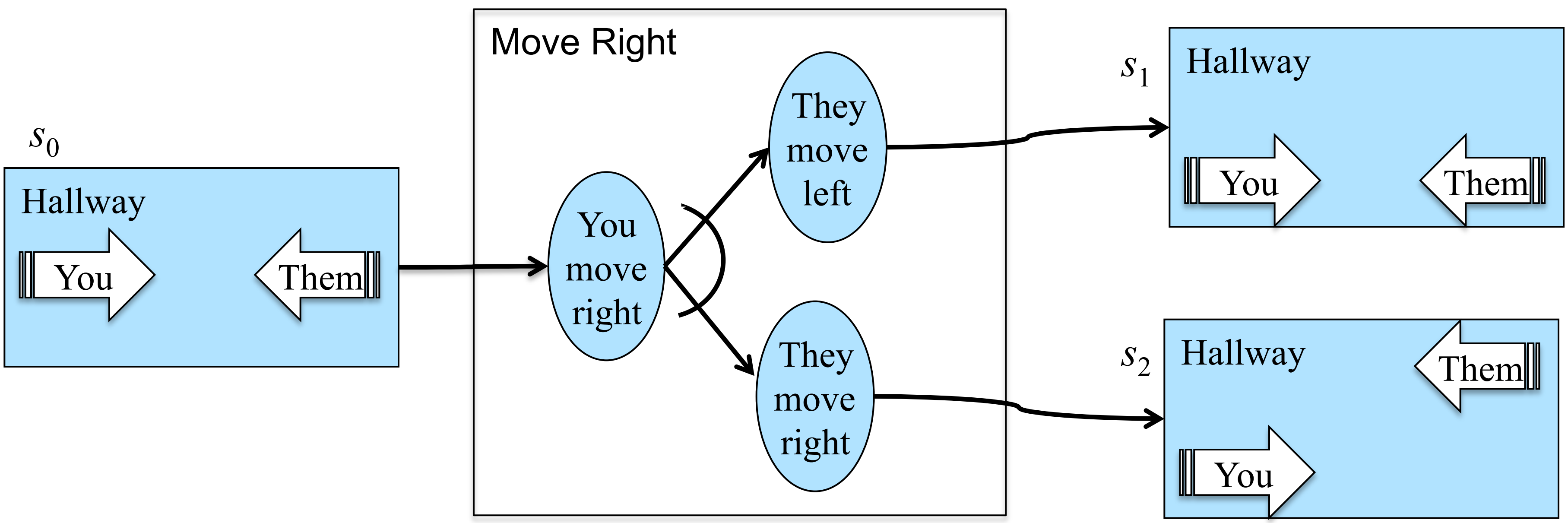
Classification of Multi-Agent Planning Problems

- Can classify multi-agent planning problems according to
  - The characteristics discussed earlier
  - How the other agents are modeled
- Can use single-agent planning techniques in two of the classes

Objectives	Execution	Observability	Agent model	Planning technique
goals	offline	full	capabilities	planning as model checking
utilities	offline	full	predictive	planning on MDPs
utilities	off/online	full	predictive	game-tree search
utilities	off/online	partial	predictive	information-set search

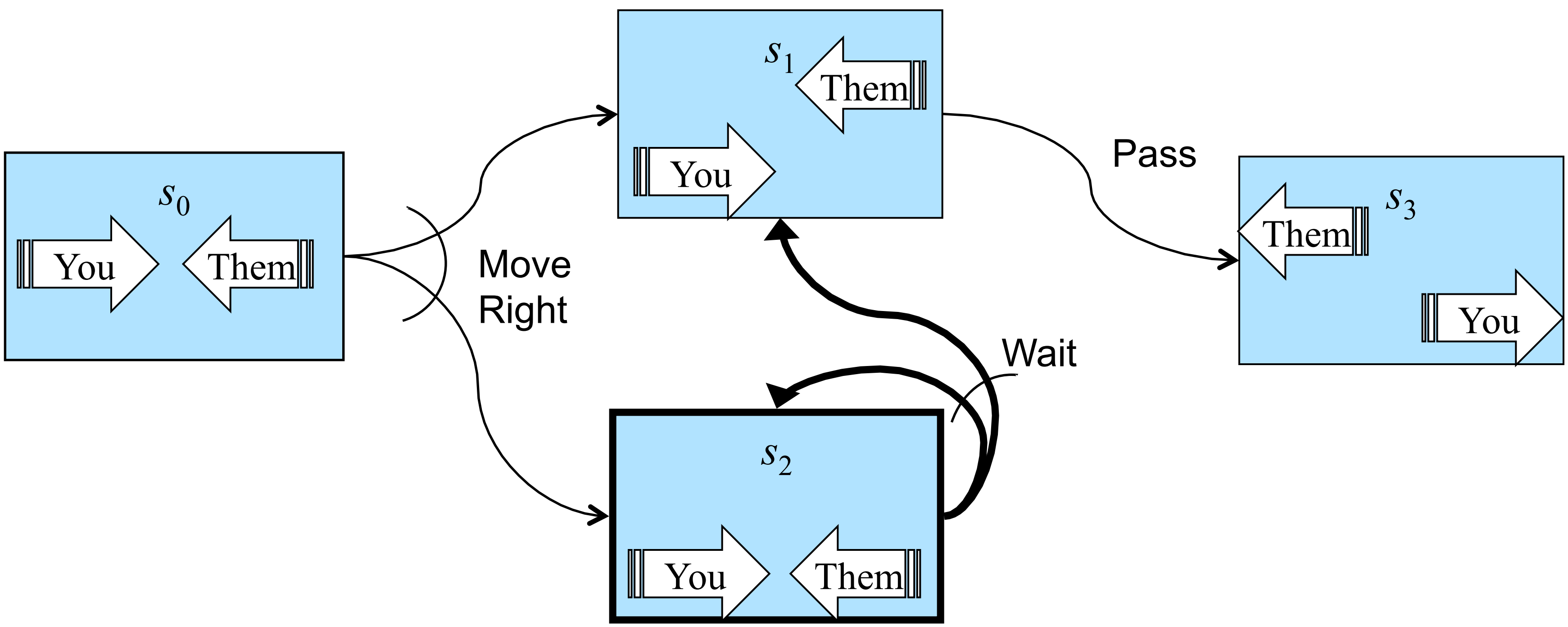
Planning as Model Checking

- Nondeterministic actions
  - Multiple outcomes, any of which might happen
  - Like MDP actions, but without the probabilities
- Can use these to model the other agents' **capabilities**
  - Encode the other agents' possible actions as outcomes of our actions



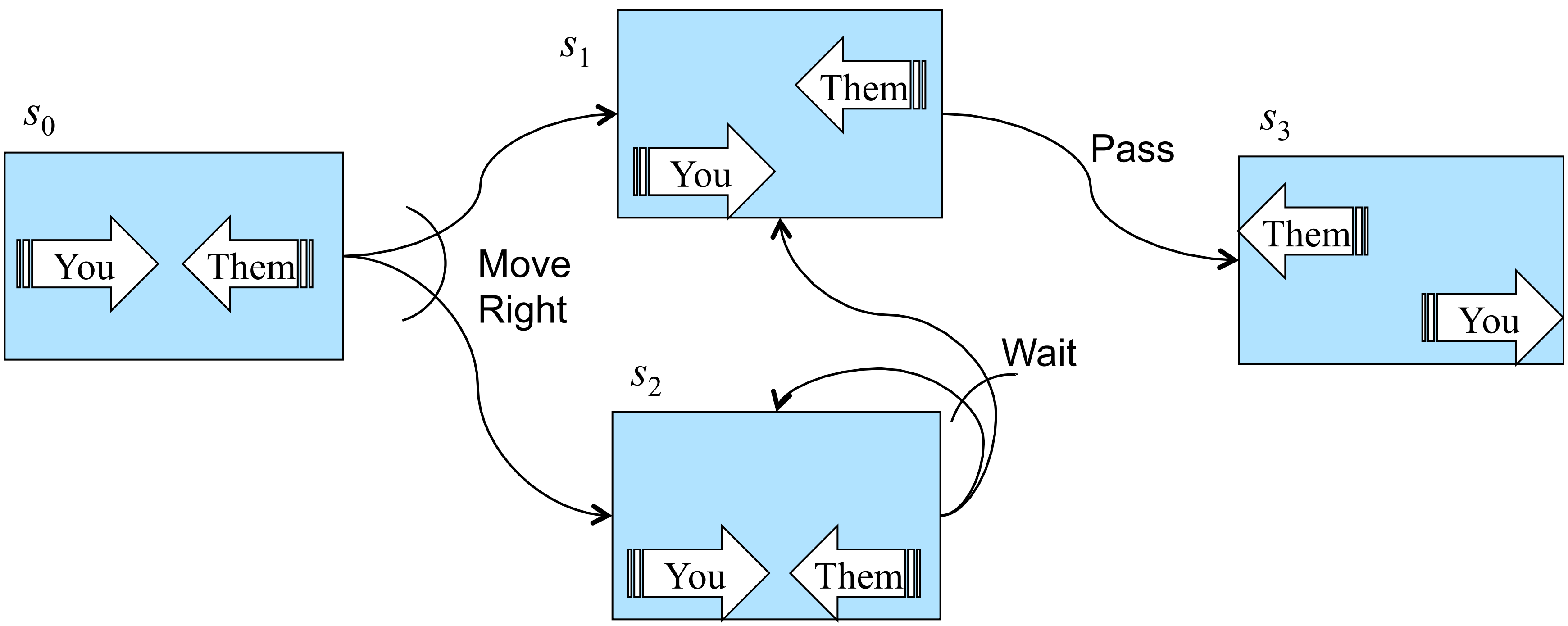
Types of Solutions

- For planning as model checking, there are three kinds of solutions
  - **weak**: at least one execution will reach a goal
  - **strong**: every execution will reach a goal
  - **strong-cyclic**: every *fair* execution will reach a goal
    - *Fair* execution: doesn't stay in a loop forever if the loop has an exit
    - Like assuming a nonzero probability for each possible outcome

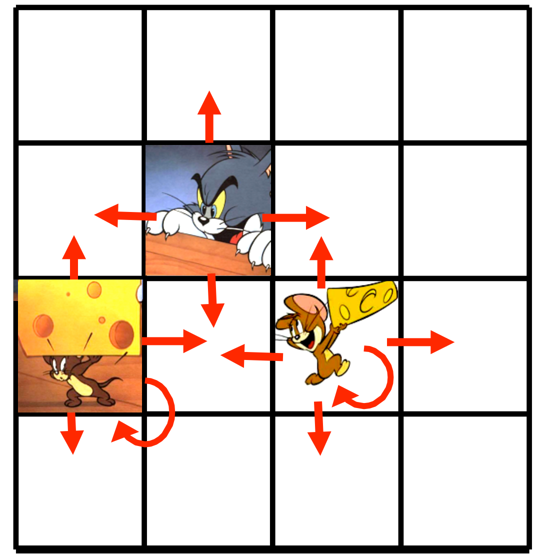


Policies and Execution Structures

- In classical AI planning, a ***plan*** is a linear sequence of actions
  - » Can't use that here
  - » Need actions to be contingent on the state of the world
- Instead, use a ***policy***: a function that maps states into actions
  - » e.g., $\pi_0 = \{(s_0, \text{MoveRight}), (s_1, \text{Pass}), (s_2, \text{Wait})\}$



Example: Hunter and Prey



- A hunter and *k* prey on a *n* x *n* grid
  - » Fully observable, offline planning, hunter's goal is to catch all of the prey
- Hunter's possible actions: *N, S, E, W, Grab*
  - » *Grab* is applicable when the hunter and a prey are at the same location

- Each prey can move *N, S, E, W,* or *Wait*
  - » Can't have multiple prey at a single location

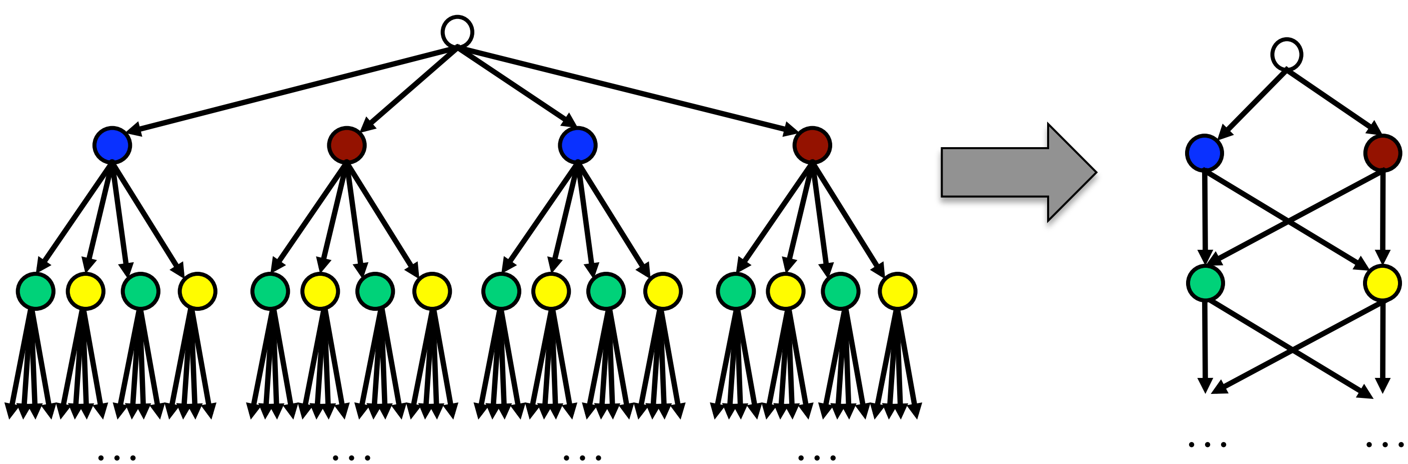
- **Combinatorial explosion**
  - » *k* prey, 5 actions per prey => each hunter's action has 5^k possible outcomes
  - » Search tree with branching factor 5^k
  - » Number of nodes at depth *d* is 5^{kd}
- **How to deal with this?**

Planning over Sets of States

- MBP searches a state space whose nodes are sets of states rather than individual states
  - » Represents sets of states as Binary Decision Diagrams (BDDs)
  - » Actions map BDDs into other BDDs
- This can reduce the search space in two ways
  - » Reduce the branching factor
  - » Fold the tree into a graph

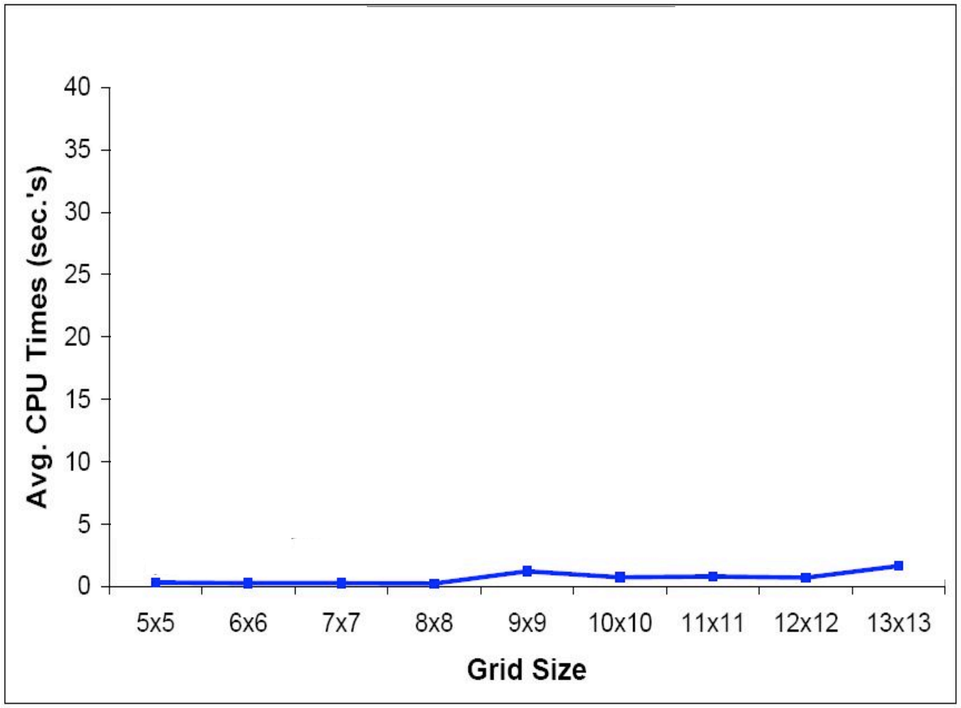
Giunchiglia & Traverso. Planning as model checking. *ECP*, 1999.

Cimatti *et al.* Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 2003.

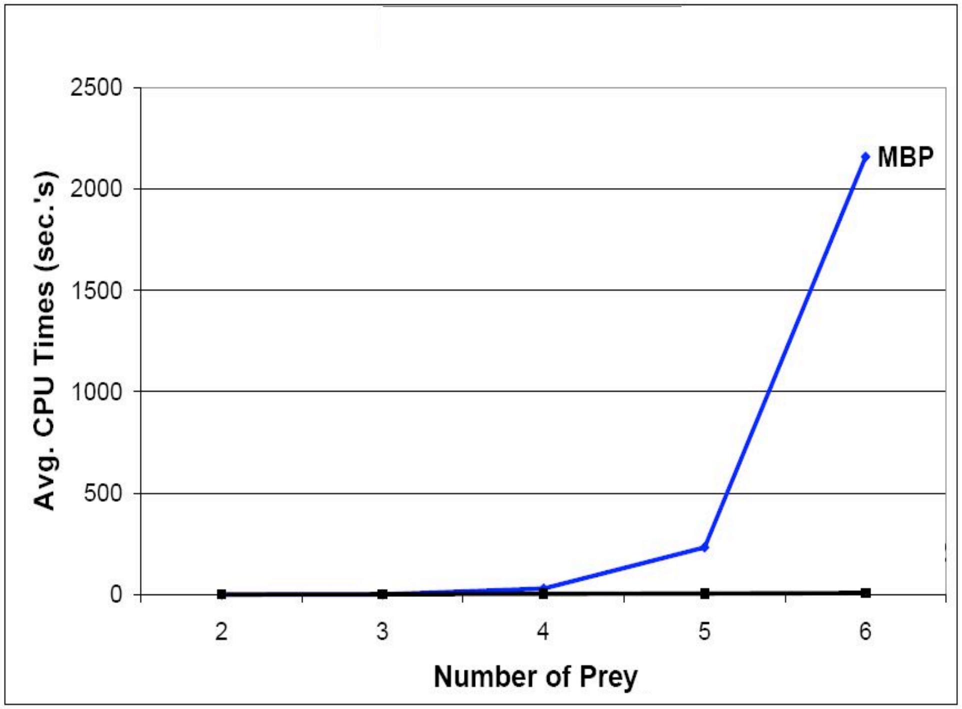


Examples of MBP's Performance

- MBP does well with one prey, regardless of grid size
- One prey on a 13x13 grid:
  - generate policy in about 2 seconds

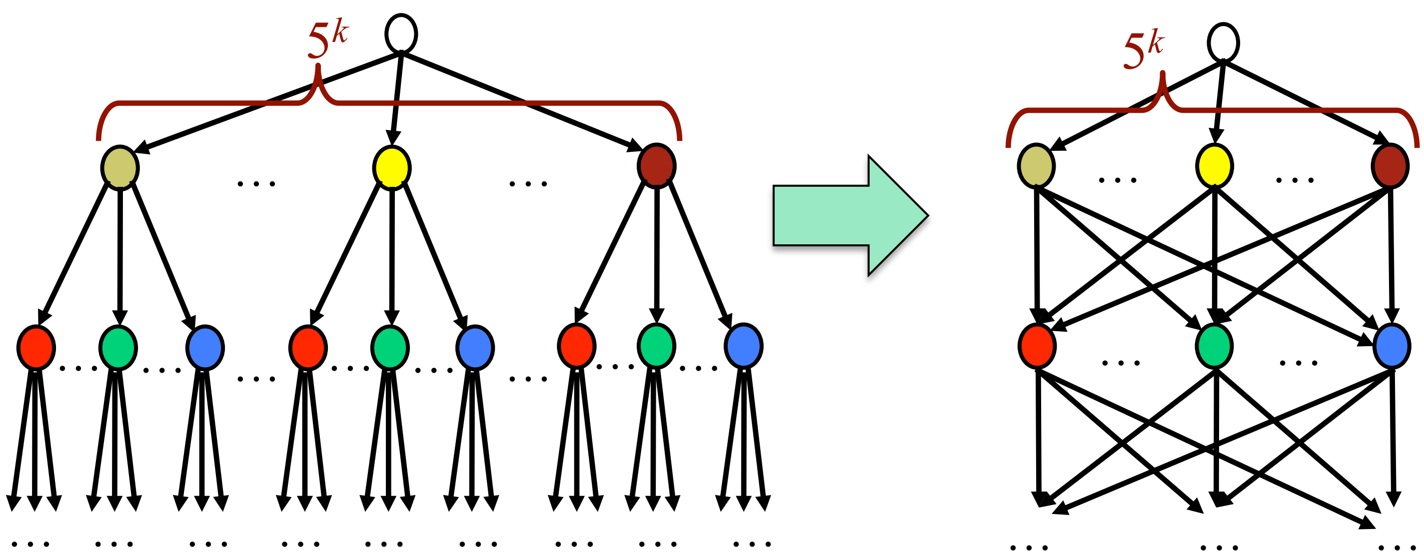


- MBP does badly with multiple prey
- Six prey on a 4x4 grid
  - > 35 minutes to generate policy

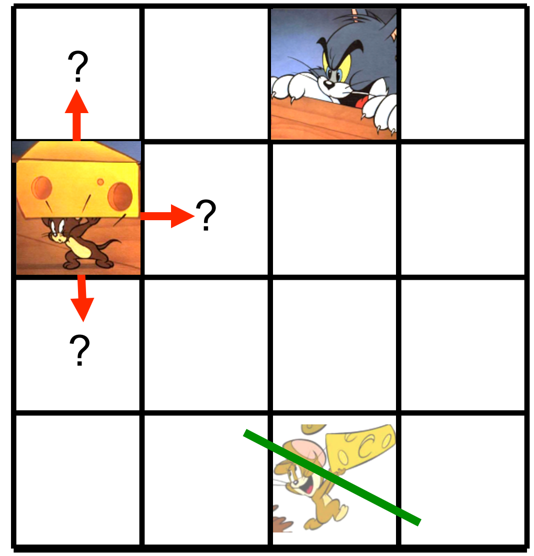


Why Multiple Prey Are a Problem

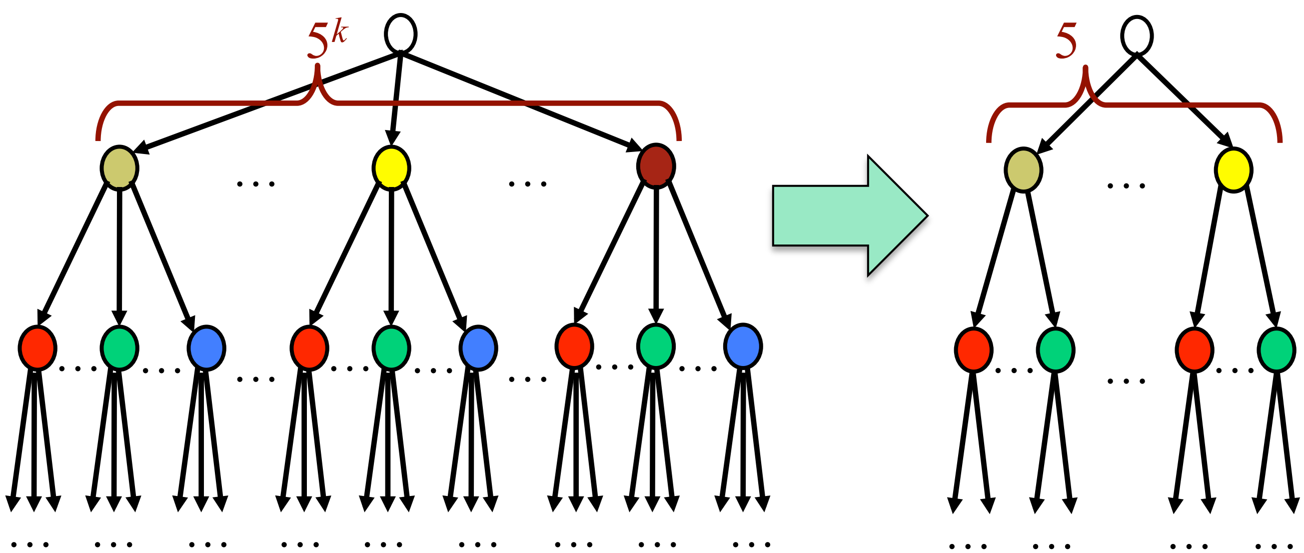
- k prey => hunter's actions each have up to 5^k outcomes
  - BDDs can't collapse them because they aren't independent
    - Can't have multiple prey at a single location
- The BDDs fold the tree into a graph, but the branching factor is still 5^k



Focusing

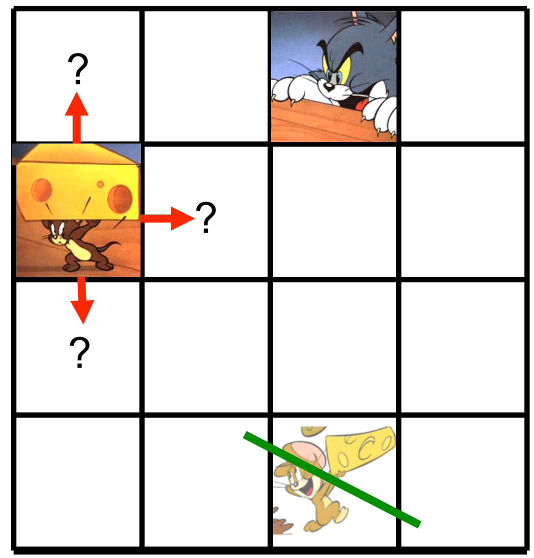


- MBP had problems with multiple prey because it tried to plan for all 5^k combinations of their actions
- Better to ***focus*** on one prey at a time
  - » Ignore the others until you've caught that one
- Reduces the branching factor
  - » ≤ 5 outcomes per action, rather than 5^k



- How to accomplish this?

Focusing via HTN Planning



- Input:
  - » ***Operators***: like in classical planning
  - » ***Tasks*** (activities to carry out) rather than goals
  - » ***Methods***: tell how to accomplish a task by decomposing it into a set of *subtasks*
- Planning by problem reduction:
  - » Decompose tasks recursively into subtasks
  - » Stop when all remaining tasks are *primitive* (i.e., correspond to actions)

- **Method** for catch-all-prey
  - **if** number of uncaught prey ≥ 1
  - **then subtasks:**
    - select(*p*), catch(*p*), catch-all-prey
  - **else subtasks:** *(none)*
- **Method** for catch(*p*)
  - **if** hunter is at *p*'s location
  - **then subtasks:** grab(*p*)
  - **else subtasks:**
    - move-toward(*p*), catch(*p*)

SHOP2

- SHOP2 is my lab's HTN planning system
  - http://www.cs.umd.edu/projects/shop
  - Won award at the 2002 International Planning Competition
  - Has been used in hundreds (thousands?) of projects worldwide

Nau *et al*. SHOP2: an HTN Planning System. *JAIR*, 2003.

Nau *et al.* Applications of SHOP and SHOP2. *IEEE Intelligent Systems*, 2005.

- SHOP2 only works in deterministic domains
  - But we can generalize it to handle nondeterminism

Generalizing to handle Nondeterminism

- ND-SHOP2: add code to
  - plan for all of the nondeterministic outcomes
  - detect cycles that have no acceptable exits

Kuter & Nau. Forward-chaining planning in nondeterministic domains. *AAAI*, 2004.

Planning for all outcomes

```
loop
  if S = ∅ then return(π)
  select a state s ∈ S and remove it from S
  if s satisfies g then insert s into solved
  else if s ∉ S_π then
```

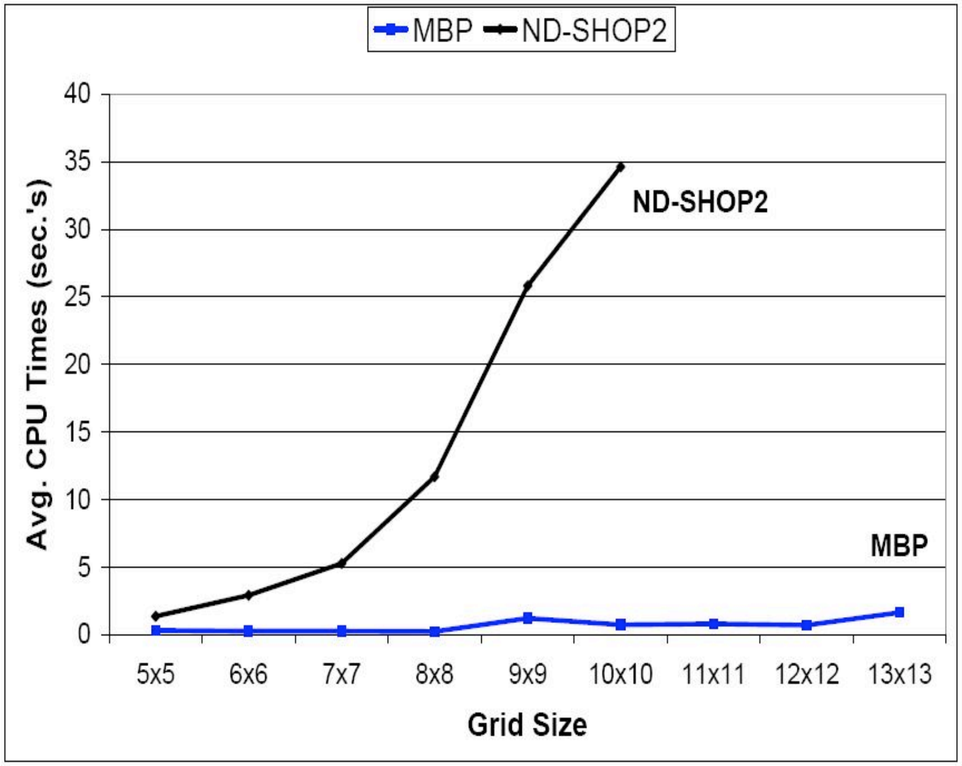
SHOP2 ***or any other*** forward-search planner

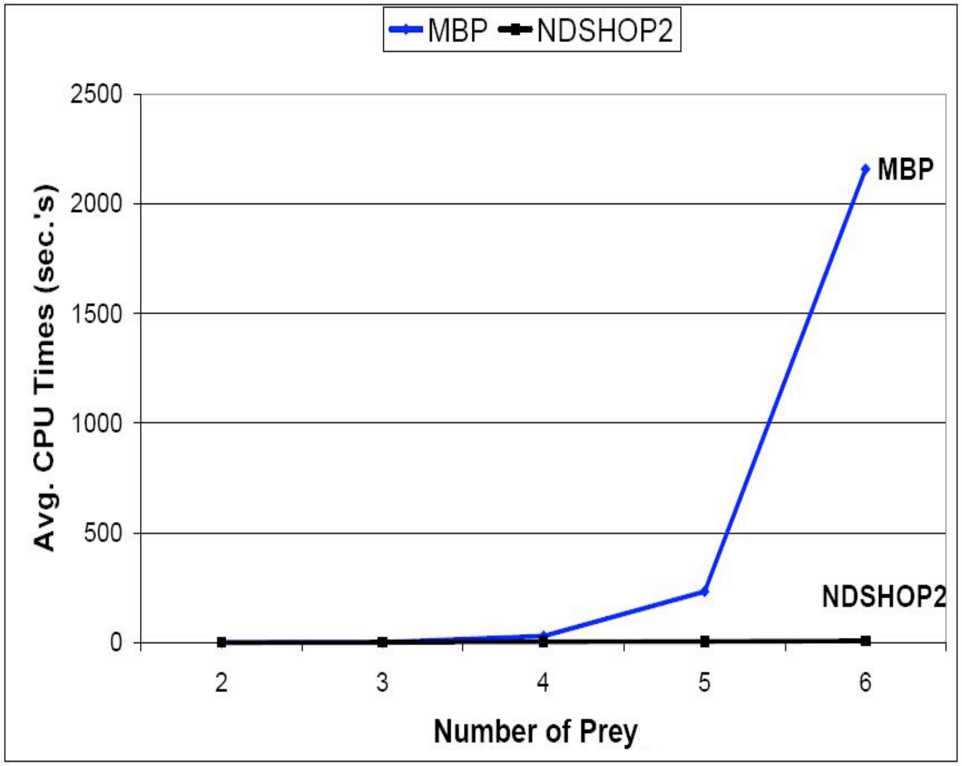
```
    actions ← {a | a is applicable to s
                   and acceptable(s, a, x, D) holds}
    if actions = ∅ then return(failure)
    nondeterministically choose a ∈ actions
    s′ ← result(s, a)
    π′ ← append(π, a)
    x′ ← progress(s, a, x, D)
```

Check for cycles that have no exits

```
  else if s has no π-descendants in (S ∪ solved) \ S_π
    then return(failure)
```

ND-SHOP2 versus MBP on Hunter-Prey





One prey on a large grid

- MBP outperforms ND-SHOP2
- MBP can plan for large sets of states at once, but ND-SHOP2 must plan for them separately

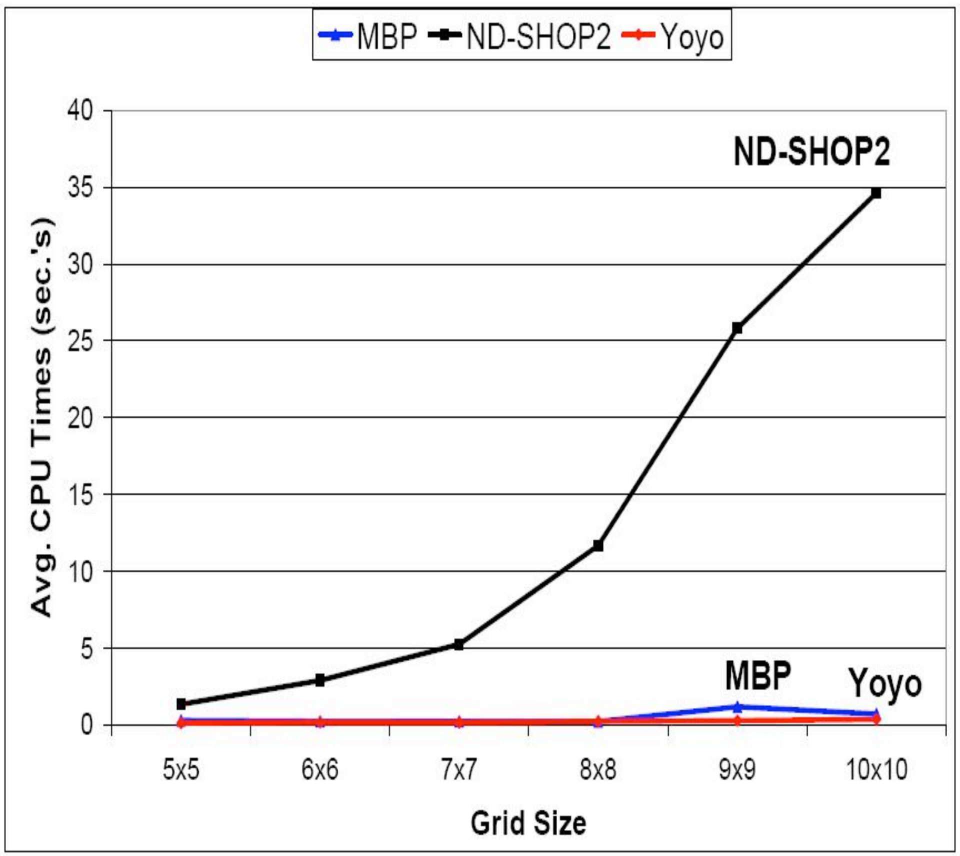
Many prey on a small grid

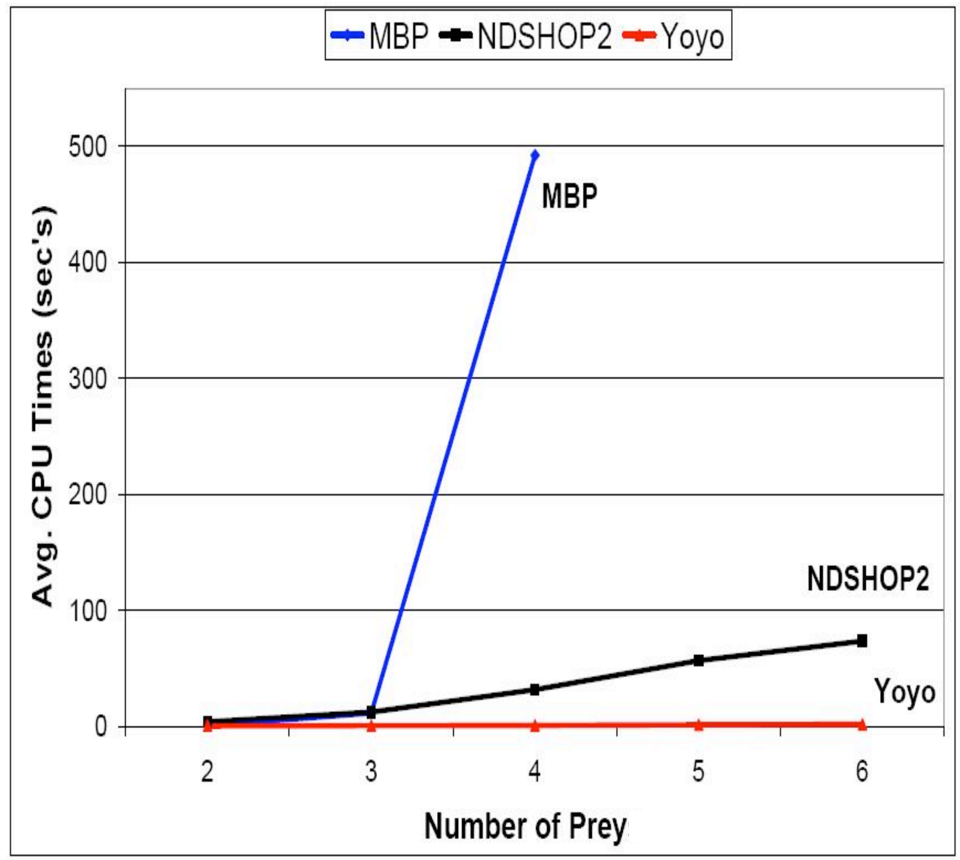
- ND-SHOP2 outperforms MBP
- ND-SHOP2 can focus on one prey at a time, but MBP can't

BDDs plus HTNs

- **Yoyo**:
  - Combines ND-SHOP2's task decomposition with MBP's BDDs
  - Outperforms both MBP and ND-SHOP2

Kuter *et al*. Task decomposition on abstract states, for planning under nondeterminism. *Artificial Intelligence*, 2009.





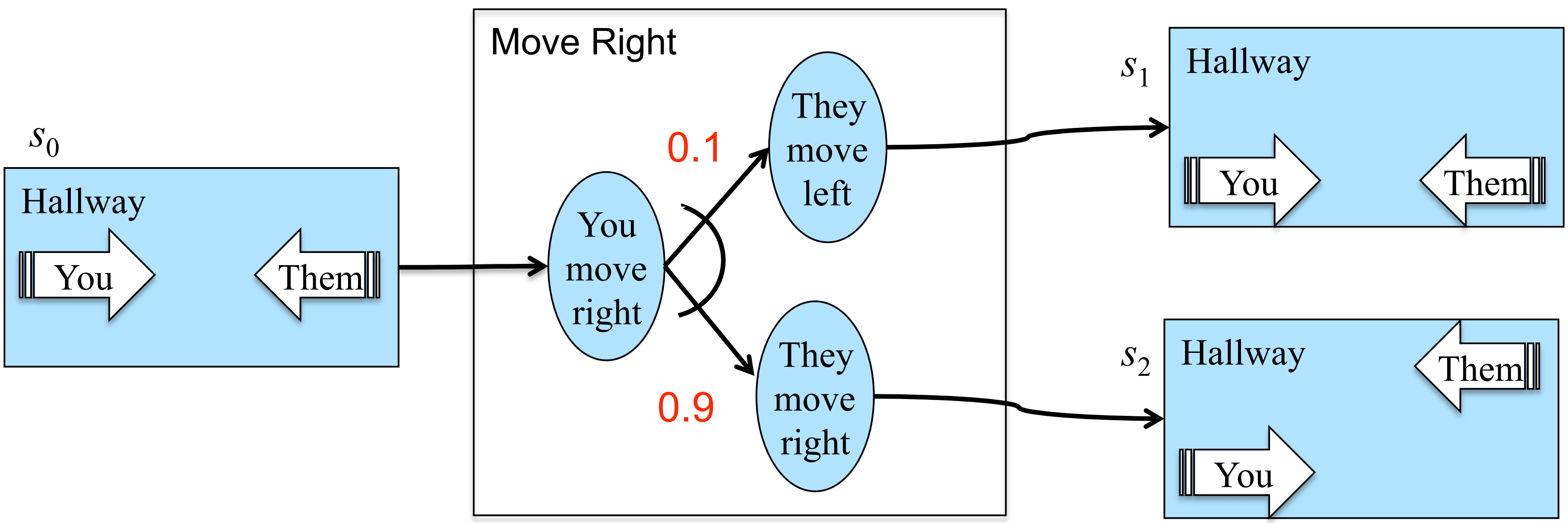
Classification of Multi-Agent Planning Problems

- Can classify multi-agent planning problems according to
  - The characteristics discussed earlier
  - How the other agents are modeled
- Can translate two of the classes into single-agent planning problems

Objectives	Execution	Observability	Agent model	Planning technique
goals	offline	full	capabilities	planning as model checking
utilities	offline	full	predictive	planning on MDPs
utilities	off/online	full	predictive	game-tree search
utilities	off/online	partial	predictive	information-set search

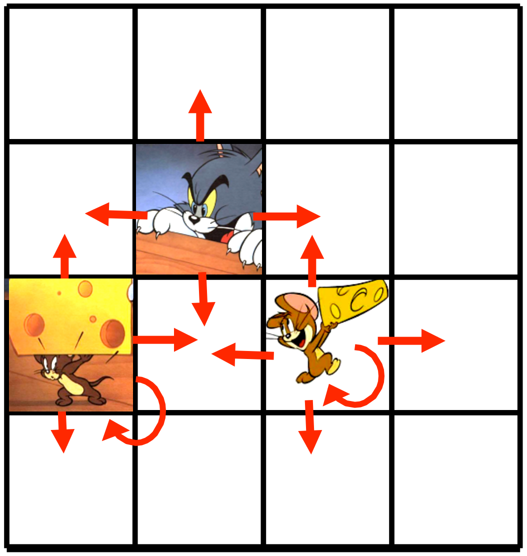
Modeling Other Agents' Actions

- Stochastic actions
  - Multiple outcomes, with probabilities for each outcome
  - Can use these to encode **predictive models** of other agents
    - Probabilities of various behaviors
- Reduce multi-agent planning to planning on MDPs



Combinatorial Explosion in MDPs

- Consider an MDP version of the Hunter-Prey problem
  - Give the hunter a utility function
    - E.g., amount of time to catch all prey
  - Predictive agent model
    - Probabilities for each prey's action
    - Encode as probabilistic outcomes of the hunter's actions



- Same combinatorial explosion as before:
  - Branching factor = 5^k

- There are some techniques analogous to the previous ones

Reducing the State Space in MDPs

- Can compress MDP state spaces in a manner similar to BDDs
  - » Algebraic Decision Diagrams (ADDs)
    - like BDDs but with numeric formulas
  - » I won't discuss ADDs *per se*
    - But in one of my later examples, I'll do an analogous classification using an *ad hoc* technique

Hoey *et al.* SPUDD: Stochastic Planning using Decision Diagrams. *UAI*, 1999.

Reducing the State Space in MDPs

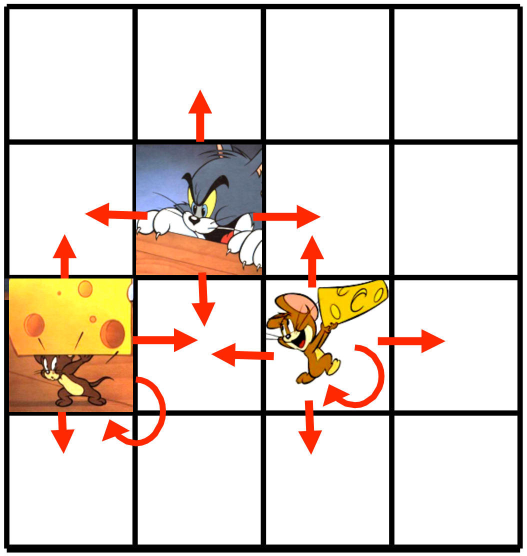
- Can incorporate focusing into several MDP planning algorithms
  - » Basic idea:
    - Run forward-search HTN planning in parallel with a forward-search MDP algorithm
    - Each time the MDP algorithm needs to know a node's successors, use HTN decomposition to compute the ones we're focusing on

Kuter *et al.* Using domain-configurable search control for probabilistic planning. *AAAI*, 2005.

- Open problem:
  - » Is it feasible to combine ADDs with focusing, like we did in Yoyo?
  - » It should be, but I don't think anyone has tried

How to Build Predictive Models?

- Need accurate probabilities for predicting the agents' actions
  - » How to get them?
- One way is from previously observed behavior
  - » E.g., in each state of the world, what are the *observed* probabilities for the prey's actions?
- Problem: may not have statistically significant data for every state of the world
- One way to handle this:
  - » Partition the states or histories into equivalence classes
    - Sets of states where you think (or hope) the prey will behave similarly
  - » In each equivalence class, what are the observed probabilities for the prey's actions?



Iterated Prisoner's Dilemma (IPD)

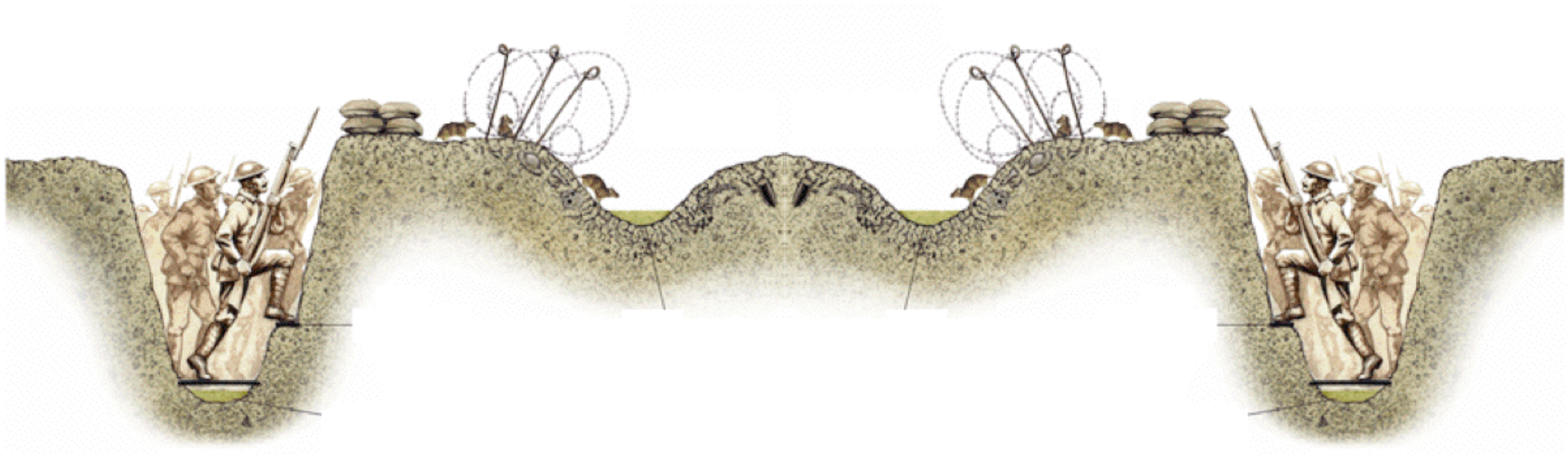
- Prisoner's Dilemma
  - » Dominant strategy is to Defect
- ***Iterated*** Prisoner's Dilemma
  - » Same pair of players, multiple times
  - » No dominant strategy
  - » Performance depends the strategies of all the players
- Axelrod (1984), *The Evolution of Cooperation*
  - » Best strategy in Axelrod's tournaments was ***Tit-for-Tat (TFT)***
    - On 1st move, cooperate. On nth move, repeat the other player's (n–1)-th move
  - » Could establish and maintain advantageous cooperations with many other players
  - » Could prevent malicious players from taking advantage of it

Prisoner's Dilemma

Player$_2$ / *Player*$_1$	Cooperate	Defect
Cooperate	3, 3	0, 5
Defect	5, 0	1, 1

Example:

- A real-world example of the IPD, described in Axelrod's book:
  - Trench warfare in World War I

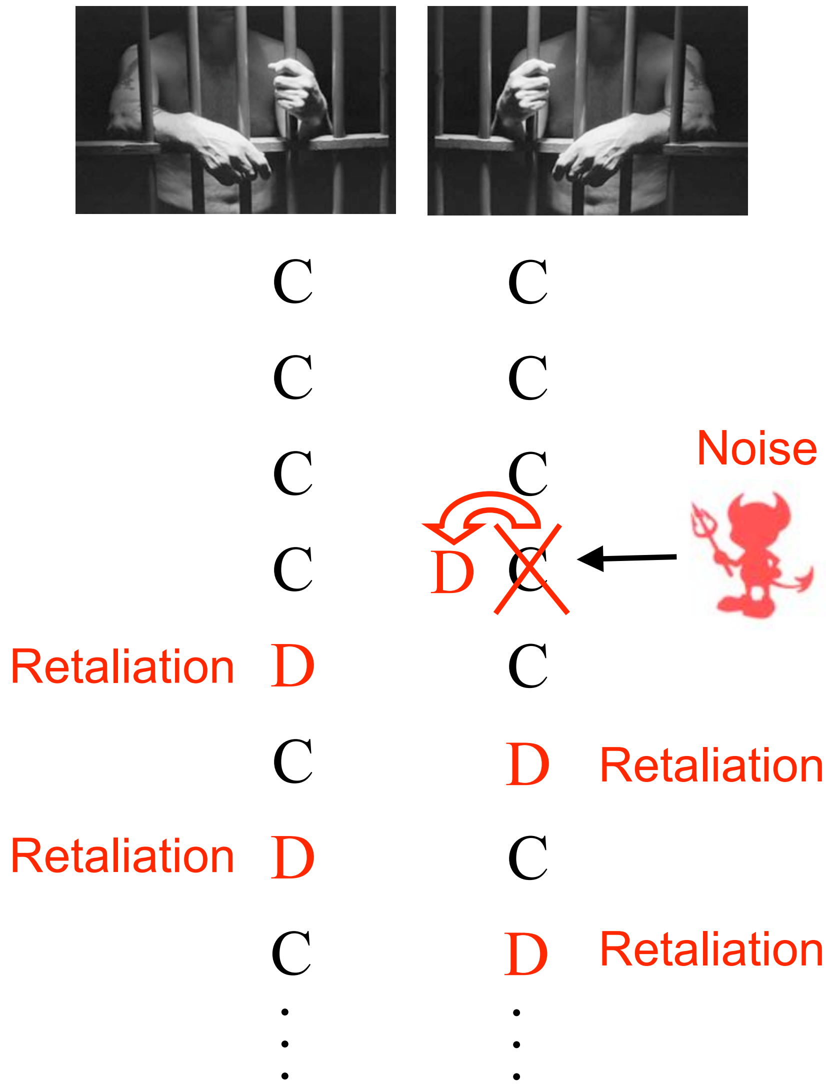


- Incentive to cooperate:
  - If I attack the other side, then they'll retaliate and I'll get hurt
  - If I don't attack, maybe they won't either
- Result: evolution of cooperation
  - Even though the two infantries were *supposed* to be enemies, they avoided attacking each other

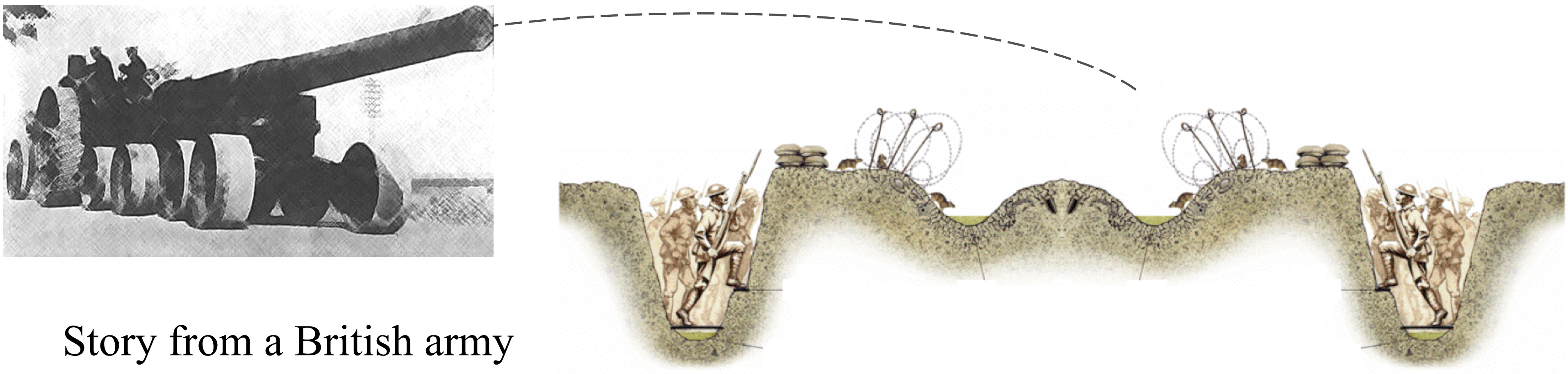
IPD with Noise

- To model accidents or misinterpretations, introduce *noise*
  - » Nonzero probability (*e.g.*, 10%) that a "noise gremlin" will change some of the actions
    - change Cooperate (C) to Defect (D)
    - and vice versa

- Noise makes it hard to maintain cooperation
  - » Consider two players who both use Tit-for-Tat
  - » One accident or misinterpretation can cause a long string of retaliations



Example of Noise



Story from a British army officer in World War I:

- I was having tea with A Company when we heard a lot of shouting and went out to investigate. We found our men and the Germans standing on their respective parapets. ***Suddenly a salvo arrived*** but did no damage. Naturally both sides got down and our men started swearing at the Germans, when all at once ***a brave German got onto his parapet and shouted out: "We are very sorry about that; we hope no one was hurt. It is not our fault. It is that damned Prussian artillery."***

The salvo wasn't the German infantry's intention

- They didn't expect it nor desire it

Discussion

- The German soldier shouted:

  "We are very sorry about that; we hope no one was hurt. It is not our fault. It is that damned Prussian artillery."

- This apology avoided a conflict
  - It was convincing because it was consistent with the German infantry's past behavior
  - The British had ample evidence that the German infantry wanted to keep the peace

- If you can tell which actions are *affected* by noise, you can avoid *reacting* to the noise

- IPD agents often behave deterministically
  - For others to cooperate with you it helps if you're predictable
- This makes it feasible to build a model from observed behavior

The DBS Agent

- From the other player's recent behavior, build a behavioral model π
  - » A set of four rules of the form **(m,m') => p**
    - **m** = our last move (C or D)
    - **m'** = their last move (C or D)
    - **p** = P(their next move will be C)
- ***Noise Filtering***:
  - » If π predicts $p = 1$ or $p = 0$ and the prediction is wrong
    - *Defer judgment*: assume it's noise
  - » If the disagreement continues
    - Assume the other player's behavior has changed
    - Recompute π based on their recent behavior
- ***Move generation***: …

Partition of game histories into four equivalence classes:

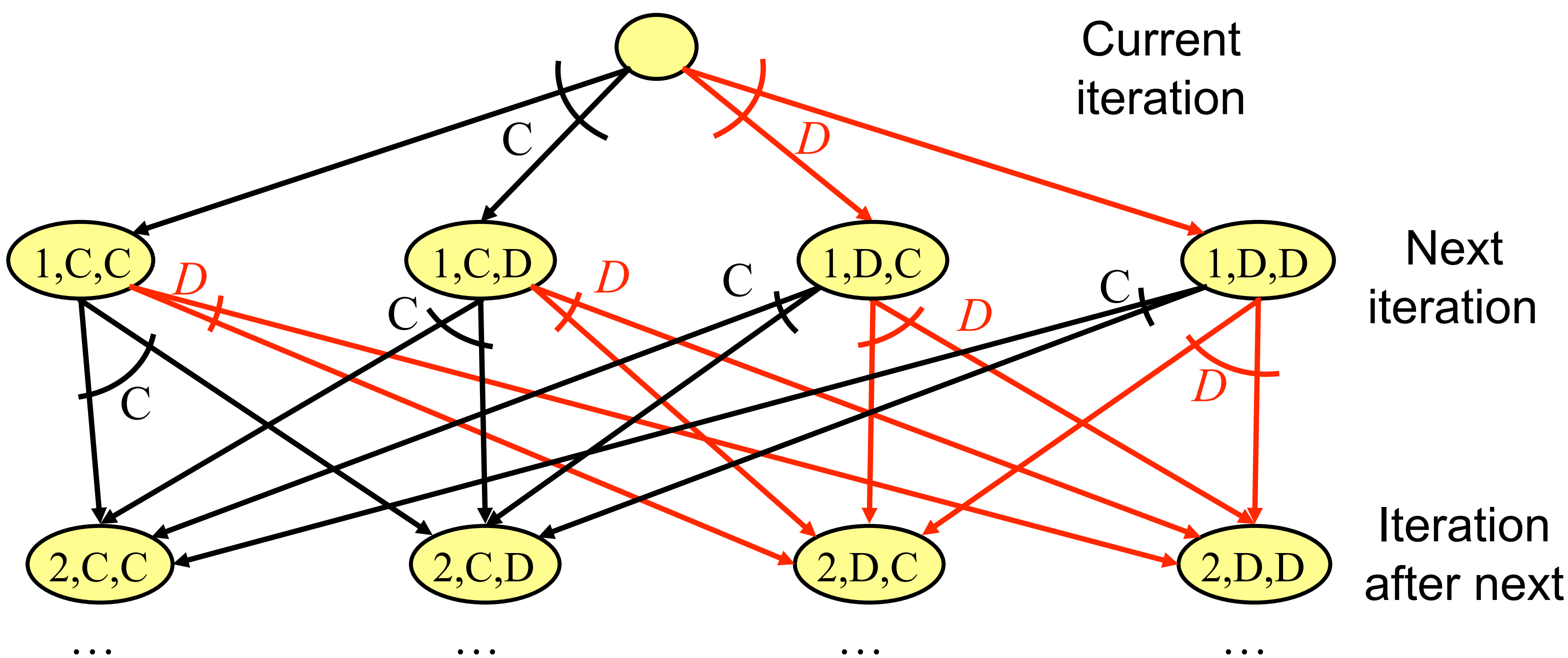
- Histories ending in (C,C)
- Histories ending in (C,D)
- Histories ending in (D,C)
- Histories ending in (D,D)

Au & Nau. Accident or intention: That is the question (in the iterated prisoner's dilemma). *AAMAS*, 2006.

Au & Nau. Is it accidental or intentional? A symbolic approach to the noisy iterated prisoner's dilemma. In G. Kendall (ed.), *The Iterated Prisoners Dilemma: 20 Years On*. World Scientific, 2007.

Move Generation

- At each iteration:
  - » Generate an acyclic MDP
    - Each state s is a triple [*iteration, our last move, their last move*]
    - Probabilities are given by the behavioral model
    - Depth = 60 (arbitrary)
  - » Solve it using dynamic programming
  - » At the root node, choose the move (C or D) with higher expected utility



20th-Anniversary IPD Competition

http://www.prisoners-dilemma.com

- Category 2: IPD with noise
  - » 165 programs participated
- DBS dominated the top 10 places
- Two programs scored higher than DBS
  - » Each of them used *master-and-slaves* strategies

Rank	Program	Avg. score
1	BWIN	433.8
2	IMM01	414.1
3	DBSz	408.0
4	DBSy	408.0
5	DBSpl	407.5
6	DBSx	406.6
7	DBSf	402.0
8	DBStft	401.8
9	DBSd	400.9
10	lowESTFT_classic	397.2
11	TFTIm	397.0
12	Mod	396.9
13	TFTIz	395.5
14	TFTIc	393.7
15	DBSe	393.7
16	TTFT	393.4
17	TFTIa	393.3
18	TFTIb	393.1
19	TFTIx	393.0
20	mediumESTFT_classic	392.9

Master & Slaves Strategy

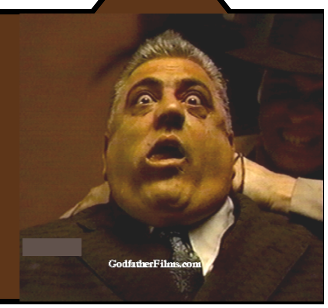
- Each participant could submit up to 20 programs
- Some submitted programs that could recognize each other
  - by communicating pre-arranged sequences of Cs and Ds
- The 20 programs worked as a team: 1 ***master***, 19 ***slaves***

- When a slave plays with its master
  - Slave cooperates, master defects
  - The master gets 5 points,
    the slave gets nothing

My slaves give
me
all their
money …

- When a slave plays with an agent not in its team
  - The slave defects, to minimize the other agent's payoff

and they
beat up
everyone
else



Comparison

- Analysis
  - » Average score of each master-and-slaves team was lower than DBS's
  - » If BWIN and IMM01 each had ≤ 10 slaves, DBS would have placed 1st
  - » Without any slaves, BWIN and IMM01 would have done badly

- In contrast, DBS had no slaves
  - » It established cooperation with many other agents
  - » It did this despite the noise
    - Its predictive model of the other agents' behavior enabled it to filter out the noise

Summary

- Reducing multi-agent planning problems to single-agent planning problems
  - Model the other agents' actions as nondeterministic outcomes of ours
- Capability model
  - Encode other agents' possible actions as nondeterministic outcomes of ours
  - BDDs, HTNs
  - Example: Hunter-Prey
- Predictive model
  - Encode information about probabilities of other agents' behaviors under various conditions
  - Building and using a predictive model
  - Example: IPD with Noise

- Next, some examples of open problems and future trends

Domain-Independent Focusing?

- In the Hunter-Prey domain, we wrote HTN methods to enable the planner to focus on one subproblem at a time
  - These methods were domain-specific
- A restricted version can be implemented without using HTNs
  - Add new preconditions and effects to the planning operators
  - Still domain-specific

R. Alford, *et al*. Maintaining focus: Overcoming attention deficit disorder in contingent planning. *FLAIRS-2009.*

- Focusing is a general idea that is useful in many different domains
  - poker
  - driving a car
  - Ph.D. research
  - giving this speech
- Can it be implemented in a domain-independent way?
  - I suspect there are restricted versions for which the answer is yes

Agent Modeling

- DBS's agent model was specific to the IPD
- For more complex environments, it is much harder to build agent models and use them effectively

Carmel & Markovich. Learning models of intelligent agents. *AAAI-95*, 1995.

- Several research efforts are focused on specific domains
  - e.g., games such as Poker

Billings *et al*. Game tree search with adaptation in stochastic imperfect information games. *Computers and Games* 1, 21–34, 2004.

Schweizer *et al*. An exploitive Monte-Carlo poker agent. *KI-2009*.

- Can we develop more general approaches?
  - Important for effective multi-agent planning

Khuller, *et al*. Computing most probable worlds of action probabilistic logic programs: scalable estimation for 1030,000 worlds. *AMAI* 51(2-4):295–331, 2007.

Subrahmanian, *et al*. CARA: A cultural-reasoning architecture. *IEEE Intelligent Systems*, Mar./Apr. 2007.

Planning and Game Theory

- Planning in multi-agent environments overlaps with game theory
- Good potential for combining planning and game theory
  - The last part of my talk (the Noisy IPD) is an example
  - Much more can be done

Objectives	Execution	Observability	Agent model	Planning technique
goals	offline	full	capabilities	planning as model checking
utilities	offline	full	predictive	planning on MDPs
utilities	off/online	full	predictive	game-tree search
utilities	off/online	partial	predictive	information-set search

A. Parker, *et al.* Overconfidence or paranoia? Search in imperfect-information games. *AAAI-2006.*

Thank You!

Any questions?

- **Acknowledgments**:
  - » Ugur Kuter wrote ND-SHOP2 and Yoyo as part of his Ph.D. research. He is now an Assistant Research Scientist at the University of Maryland.
  - » Tsz-Chiu Au wrote DBS as part of his Ph.D. research. He is now a postdoctoral researcher at the University of Texas.
  - » This work was supported in part by AFOSR grant FA95500610405 and DARPA IPTO grant FA8650-06-C-7606. The opinions in this presentation are mine, and do not necessarily reflect the opinions of the funders.