# CAP6671 Intelligent Systems

**Lecture 7:**

**Trading Agent Competition: Bidding under Uncertainty**

Instructor: Dr. Gita Sukthankar
Email: gitars@eecs.ucf.edu

Schedule: T & Th 9:00-10:15am

Location: HEC 302

Office Hours (in HEC 232):
T & Th 10:30am-12

# TAC Problem

- Acquire a certain number of items within a period of time
- Can we write a plan to handle item acquisition?
  - Input: domain information
  - State: current ask/sell prices
  - Output: list of bidding actions
    - Bid(Item, Price, Time, Number)

# Problems with Planning for TAC

- Continuous state, action space makes it difficult to consider all alternatives
- High probability of failure at every decision point
- Simultaneous decision on multiple items
- Irrecoverable errors
- Dependencies between goals

# 28 Simultaneous Auctions

- Flights: Inflight days 1-4, Outflight days 2-5 (8)
  - Separate auction for each type of plane ticket
  - Ask price set by server periodically increases/decreases randomly
  - Obtain ticket by bidding at or above ask price
- Hotels: Expensive hotel/cheap hotel for days 1-4 (8)
  - 16 rooms per auction; $16^{th}$ price ascending English auction; no resale
  - Ask price published by server is $16^{th}$ highest price; no information about other bids
  - No bid withdrawal; no resale
  - Hotel auctions can close after a specified period of inactivity
- Entertainment: 3 different types of tickets, 4 nights (12)
  - Agents start with a set of entertainment tickets
  - Server publishes bid-ask spreads (highest bid price, lowest ask price)
  - Continuous double auction (no trading phases, prices to buy and sell may be submitted at any time
  - Resale allowed

# Bidding Strategies

- Flights: delay commitment
- Entertainment tickets: resell sub-optimal decisions
- Hotels?
  - Irrevocable resource commitment
  - Simultaneous auctions
  - Combinatorial valuations

# Combinatorial Valuations

- Complements

$$v(X\bar{Y}) + v(\bar{X}Y) \leq v(XY)$$

  - Camera, flash, and tripod
- Substitutes

$$v(X\bar{Y}) + v(\bar{X}Y) \geq v(XY)$$

  - Canon AE-1 and A-1

# Marginal Utility

- Marginal utility is a mechanism for determining whether it's worth bidding on a new object
- Definition:
  - Difference between the utility of owning the set plus the new object vs. owning the set without the new object minus purchase costs
- Formally:

the *marginal utility* of good $x \notin X \cup Y$ is defined as follows: $\mu(x, X, Y, \vec{p}) = \alpha(X \cup \{x\}, Y, \vec{p}) - \alpha(X, Y, \vec{p})$.

- Example:
  - MU(camera+flash) vs MU(flash alone)

# Problems with Marginal Utility

- Simultaneous auctions
  - Marginal utility doesn't take into account the possibility of obtaining other substitute valuations through simultaneous auctions.
  - Amy's example:

**Example 1.1** Consider a set of $N > 1$ goods that are being auctioned off simultaneously. Assume the value of one or more of these goods is 2, while the auction price of each good is 1, deterministically.[2] In this setting, bidding marginal utilities amounts to bidding 1 on each good. In doing so, this strategy obtains utility $2 - N < 1$. In contrast, any strategy that bids 1 on exactly one good obtains utility $2 - 1 = 1$. Thus, bidding marginal utilities is suboptimal. $\square$

# Research Problem

- Comparison of marginal utility bidding (ATTac) and policy search (RoxyBOT)
- Results:
  - Marginal utility is not optimal in simultaneous auction (as shown by example)
  - Optimal in sequential option
  - Empirical results demonstrate that MU bidding is a reasonable heuristic for TAC Classic hotel auctions

# ATTac

- Bidding:
  - Calculate G* (most profitable allocation of goods to clients based on current holdings and predicted future prices) for use in bidding
  - Buy/sell bids for entertainment based on a sliding price strategy (dependent on time till end of game)
- Allocation:
  - Uses MILP to find optimal allocation
- Online adaptation to game conditions:
  - Passive/active bidding modes based on server latency
  - Allocation strategy based on time required for MILP
  - Hotel bidding based on closing prices in previous games

# RoxyBot

- Allocation
  - Using an A* search with admissible heuristic or variable-width beam search
- Completer
  - Optimal quantity of resources to buy and sell using priceline structure to forecast future costs
  - Pricelines are learned using ML techniques (whereas ATTac uses heuristics to estimate future prices)

# Markov Decision Processes

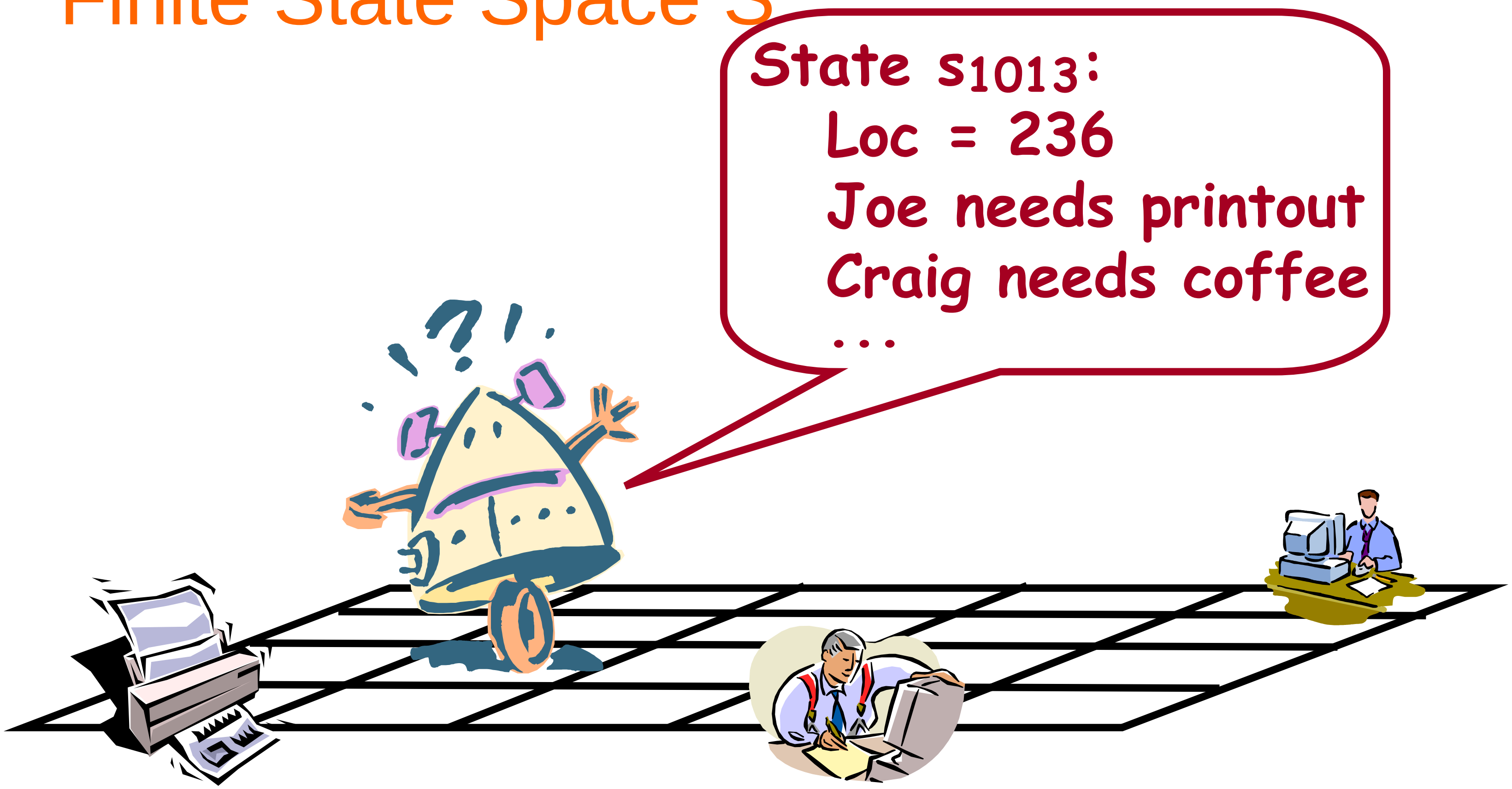
- Sequential auctions can be modeled as an MDP.
- Classical planning models:
  - logical representation of transition systems
  - goal-based objectives
  - plans as sequences
- Markov decision processes generalize this view
  - controllable, stochastic transition system
  - general objective functions (rewards) that allow tradeoffs with transition probabilities to be made
  - more general solution concepts (policies)

# Markov Decision Processes

- An MDP has four components, S, A, R, Pr:
  - (finite) state set S   (|S| = n)
  - (finite) action set A   (|A| = m)
  - transition function Pr(s,a,t)
    - each Pr(s,a,-) is a distribution over S
    - represented by set of n x n stochastic matrices
  - bounded, real-valued reward function R(s)
    - represented by an n-vector
    - can be generalized to include action costs: R(s,a)
    - can be stochastic (but replacable by expectation)
- Model easily generalizable to countable or continuous state and action spaces

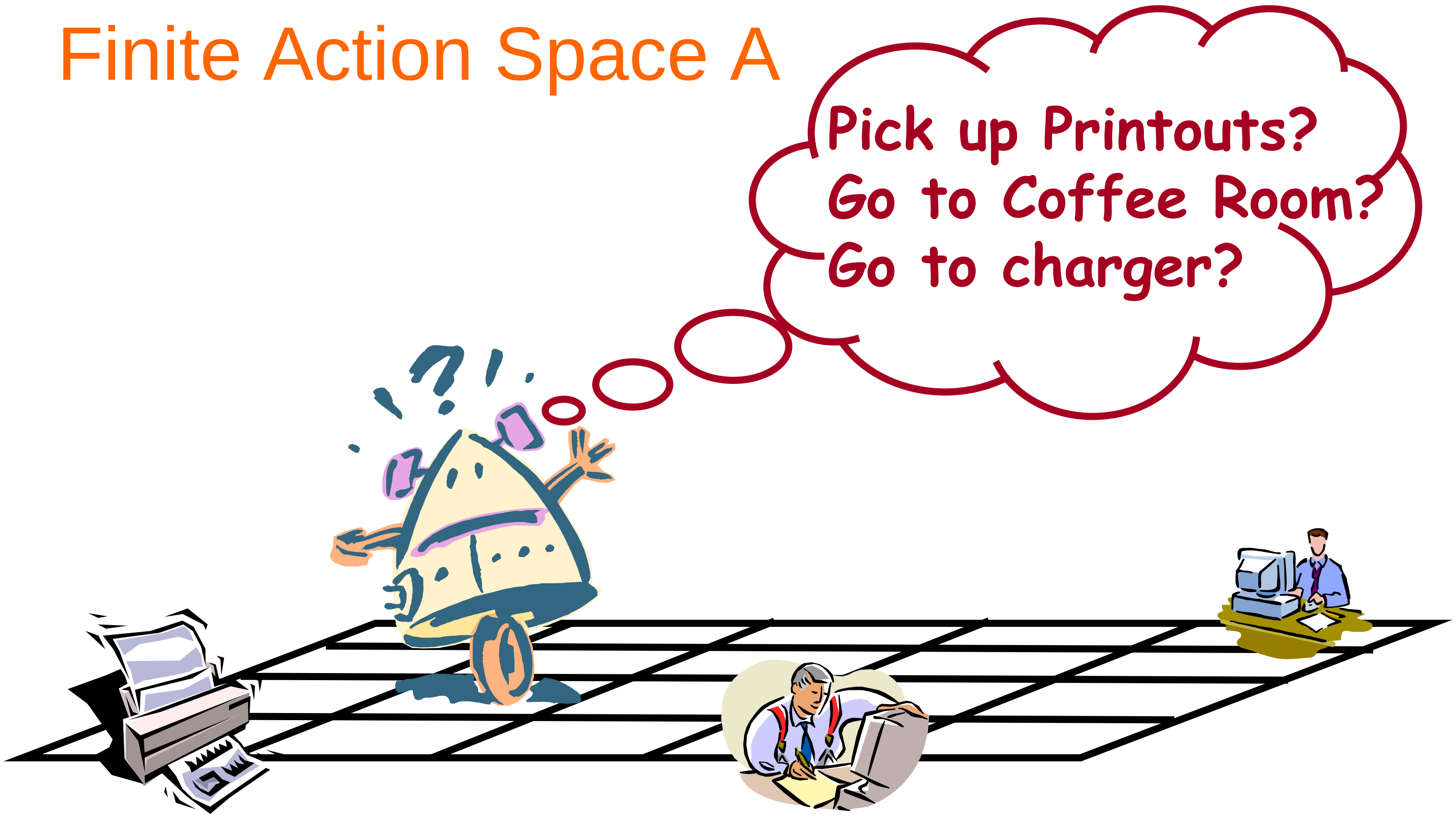
# System Dynamics

## Finite State Space S



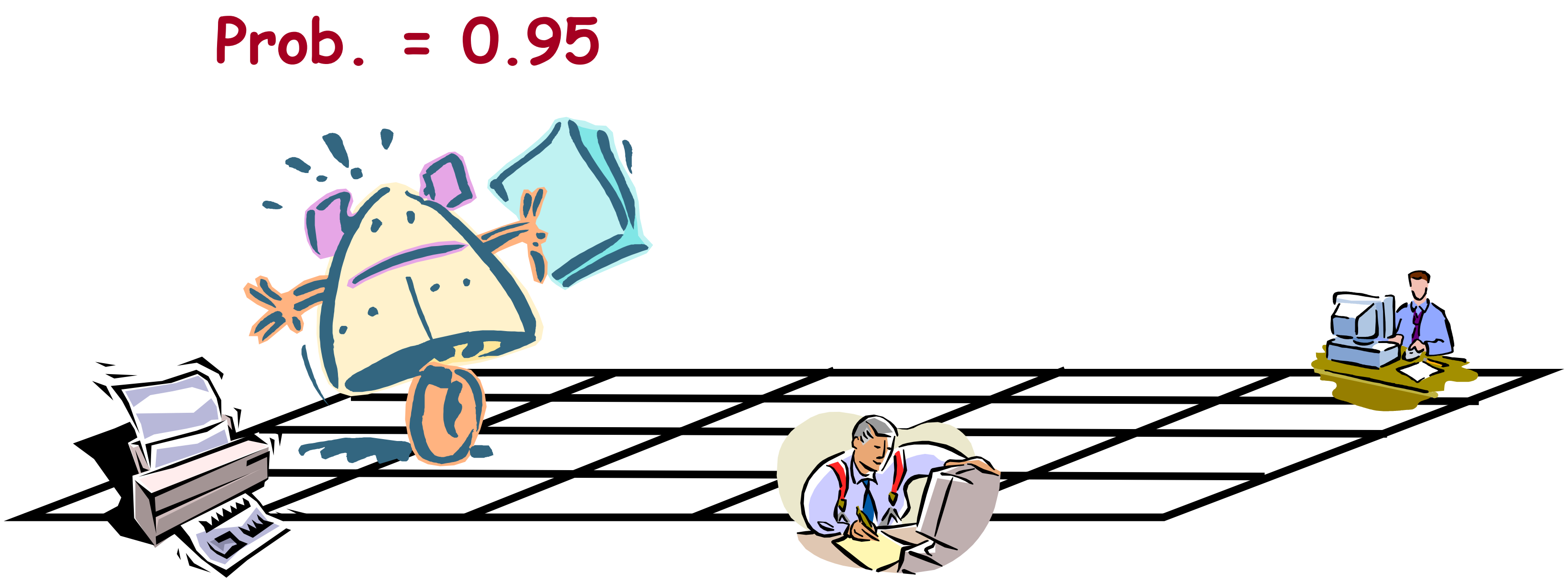
# System Dynamics

Finite Action Space A



# System Dynamics

Transition Probabilities: Pr($s_i$, a, $s_j$)

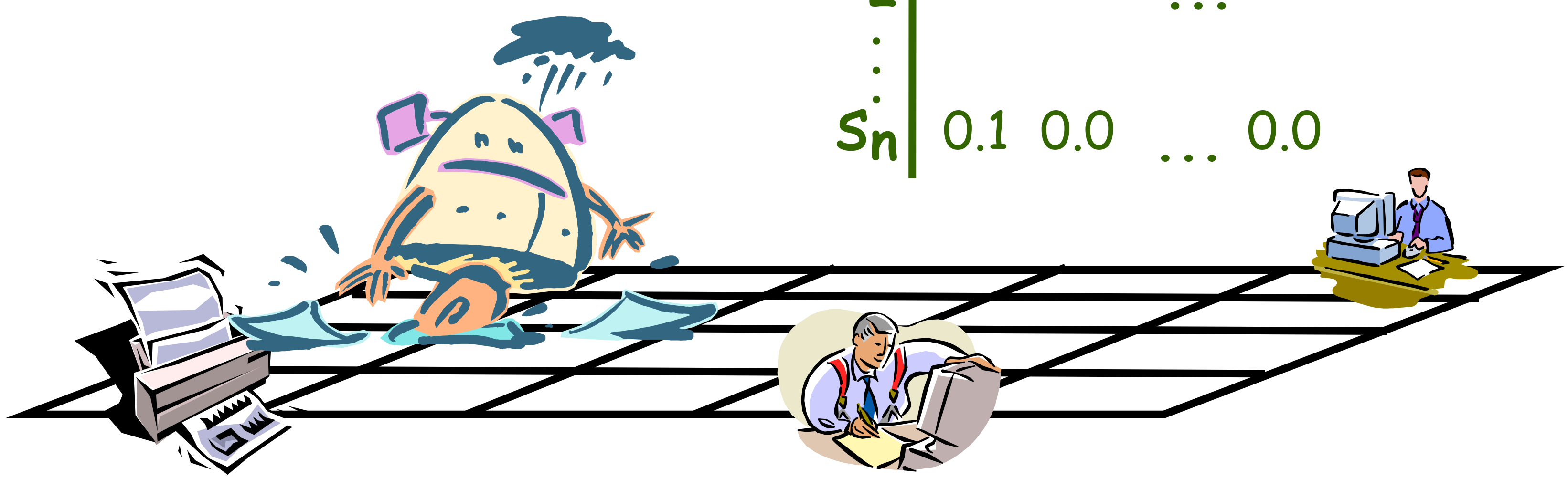


# System Dynamics

Transition Probabilities: Pr($s_i$, a, $s_k$)

**Prob. = 0.05**

| | $s_1$ | $s_2$ | ... | $s_n$ |
|---|---|---|---|---|
| $s_1$ | 0.9 | 0.05 | ... | 0.0 |
| $s_2$ | 0.0 | 0.20 | ... | 0.1 |
| ⋮ | | | | |
| $s_n$ | 0.1 | 0.0 | ... | 0.0 |



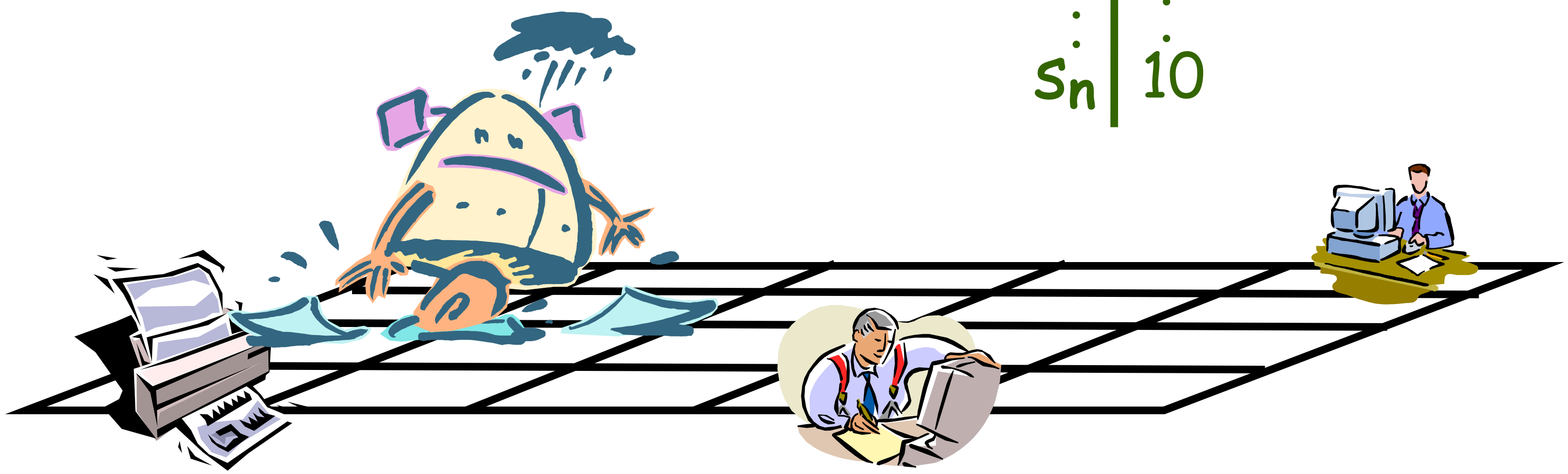
# Reward Process

Reward Function: $R(s_i)$

- action costs possible

| | R |
|---|---|
| $s_1$ | 12 |
| $s_2$ | 0.5 |
| ⋮ | ⋮ |
| $s_n$ | 10 |

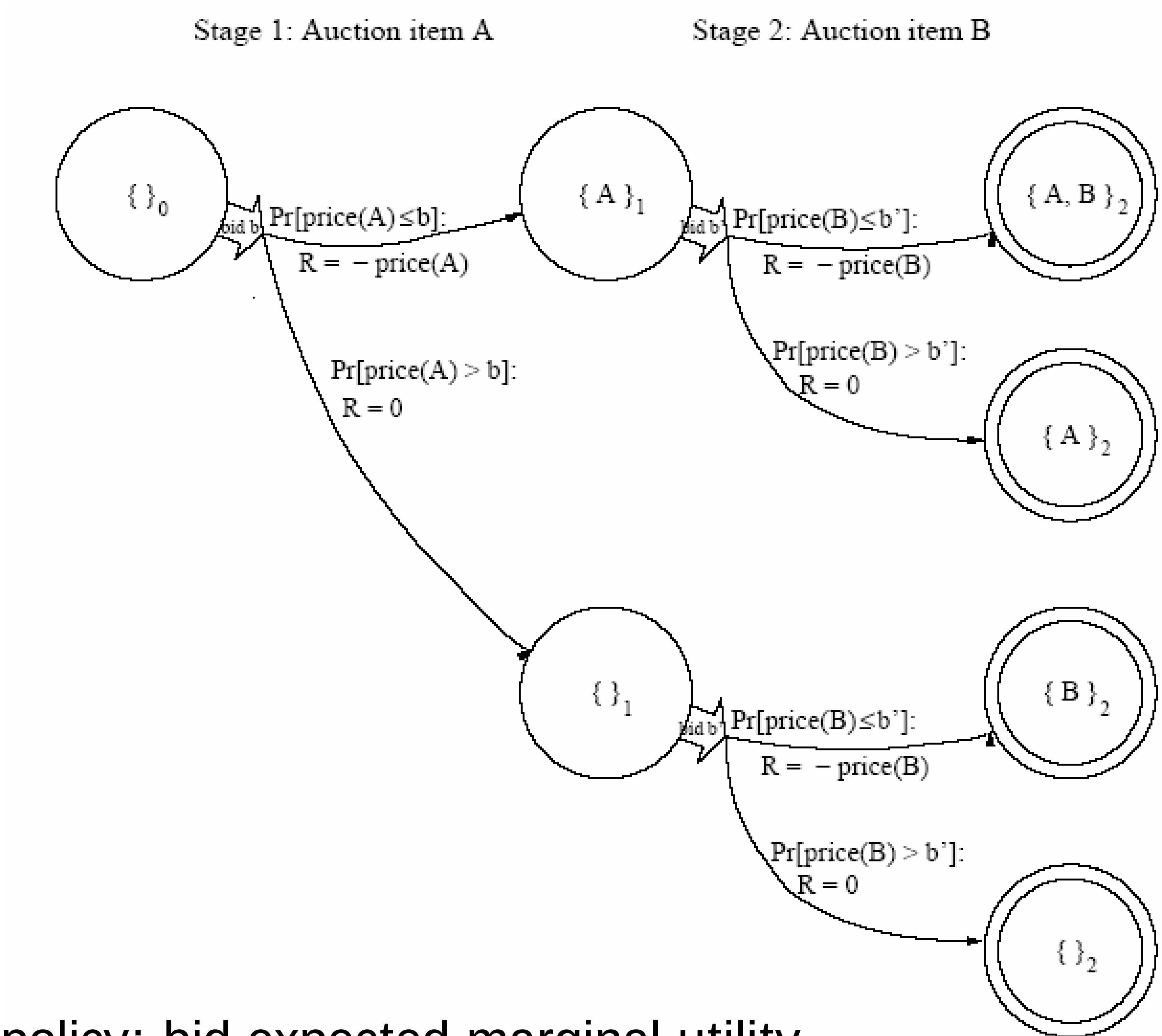
**Reward = -10**



# Assumptions

- Markovian dynamics (history independence)
  - $Pr(S^{t+1}|A^t,S^t,A^{t-1},S^{t-1},..., S^0) = Pr(S^{t+1}|A^t,S^t)$
- Markovian reward process
  - $Pr(R^t|A^t,S^t,A^{t-1},S^{t-1},..., S^0) = Pr(R^t|A^t,S^t)$
- Stationary dynamics and reward
  - $Pr(S^{t+1}|A^t,S^t) = Pr(S^{t'+1}|A^{t'},S^{t'})$ for all t, t'
- **Full observability**
  - though we can't predict what state we will reach when we execute an action, once it is realized, we know what it is

# MDP for Bidding




Optimal policy: bid expected marginal utility

# Simultaneous Auctions

- How are simultaneous auctions different?
- Camera and flash scenario:
  - U(camera, flash)=750
  - U(camera)=0
  - U(flash)=0
  - Price(flash)=50
  - Price(camera) either $500 (0.5) or $1000 (0.5)
  - What to bid?
    - Bid of (0,0) yields an expected utility of $9
    - Bid of (500,50) yields an expected utility of $150 ($200 half the time, -$50 half the time)

# Problem Formulation

- Compute expectation over all possible outcomes (win good 1, win good 2)
- Problem: exactly computing this expectation is exponential in number of goods
- Approaches:
  - Expected MU bidding
  - Expected value with MU bidding
  - Stochastic sampling technique

# Expected MU bidding (ATTac)

| $x$ | $y$ | $\mu(x)$ | $\mu(y)$ | Evaluation |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | -1 |
| 1 | 101 | 1 | 1 | 0 |
| 101 | 1 | 1 | 1 | 0 |
| 101 | 101 | 1 | 1 | 0 |
| Average | | 1 | 1 | $-\frac{1}{4}$ |

Method: calculate expectation on marginal utility
Result: expected marginal utility bidding (bid 1 on both goods) is suboptimal

# Expected Value Method

- Expected value: solve deterministic version of problem using prices calculated by expected values
  - U(camera, flash)=750
  - Price(flash)=50
  - Price(camera) either $500 (0.5) or $1000 (0.5)
  - Calculate policy using expected price $750+$50=$800
  - Since expected price is higher than utility, expected value recommends no bid
  - But that isn't quite right either...

# Expected Value Method/MU Bid

- Marginal utility bidding can do better than using the expected value:

**Example 3.2** Consider only one good $a$ of value \$100. Suppose $a$'s price is \$1 with probability .9, but that $a$'s price is \$1 million with probability .1. Thus, the expected price of good $a$ is roughly \$100,0010. The optimal policy using the expected value method is to bid \$0, which scores \$0. But now consider the bidding policy "bid \$100." This policy scores \$99 with probability .9, and \$0 with probability .1. Thus, on average, this policy scores roughly \$89. "Bid \$100" dominates the expected value method in this example. Indeed, "bid \$100," which corresponds to bidding expected marginal utility, is optimal, since this auction is sequential. □

- Combine approaches:
  - Compute optimal set of goods using expected value
  - Bid for the goods using marginal utility

# Sample Average Approximation

- Used by RoxyBot-02 (TAC-02)
- Solve stochastic program using a subset of scenarios
- Without heuristics just a form of generate and test
- Policies can be generated using MU, EVMU or expected MU

# Results

- MU outperforms expected MU
- RoxyBot-00 outperforms MU
- RoxyBot-02 (using SAA) outperforms-00

- Problems:
  - SAA is very slow if it just uses brute force search
  - Needs a good heuristic to direct the search
  - Computing policies to direct the search is in itself computationally expensive

# Reading

- Reading**:** David Pardoe and Peter Stone. An Autonomous Agent for Supply Chain Management. In Gedas Adomavicius and Alok Gupta, editors, Handbooks in Information Systems Series: Business Computing, Elsevier, 2007.