Concise representations of games

Vincent Conitzer
conitzer@cs.duke.edu

Games with many agents

- How do we represent a (say, simultaneous-move) game with n agents?
- Even with only 2 actions (pure strategies) per player, there are 2^n possible outcomes
  - Impractical to list them all
- Real-world games often have structure that allows us to describe them concisely
- E.g., complete symmetry among players
  - How would we represent such a game?
  - How many numbers (utilities) do we need to specify?
  - For more, see e.g., Brandt et al. 2009, Jiang et al. 2009
- What other structure can we make use of?

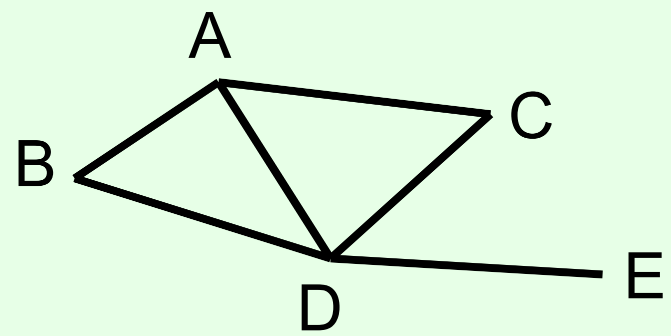
Congestion games [Rosenthal 73]

- There is a set of resources R
- Agent i's set of actions (pure strategies) A_i is a subset of 2^R, representing which subsets of resources would meet her needs
  - Note: different agents may need different resources
- There exist cost functions $c_r: \{1, 2, 3, \ldots\} \rightarrow \Re$ such that agent i's utility for $a = (a_i, a_{-i})$ is $-\Sigma_{r \text{ in } a_i} c_r(\#(r, a))$
  - #(r, a) is the number of agents that chose r as one of their resources

Example: writing paper/playing video game

- Player 1 needs to write a paper
- Player 2 (player 1's roommate) "needs" to play a video game
- Resources: 1's Laptop (L1), 2's Laptop (L2), Video game system (V), Internet connection (I), Common room (C), Bedroom (B)
  - Both rooms have (shared) internet connection, video game system cannot be moved out of common room
- Player 1's action set: {{L1, I, C}, {L1, I, B}}
- Player 2's action set: {{V, C}, {L2, I, C}, {L2, I, B}}
- For all resources r, $c_r(1) = 0$, $c_r(2) = 1$
- What are the pure-strategy equilibria?

Example: network routing



- Player 1 has source A and target E
- Player 2 has source B and target C
- Player 3 has source B and target E
- Resources = edges
- Player 1's action set: {{AD,DE}, {AB,BD,DE}, {AC,CD,DE}}
- Player 2's action set: {{AB,AC}, {BD,CD}, {AB,AD,CD}, {BD,AD,AC}}
- Player 3's action set: {{BD,DE}, {AB,AD,DE}, {AB,AC,CD,DE}}
- For all resources r, $c_r(1) = 0$, $c_r(2) = 1$, $c_r(3) = 2$
- Pure-strategy equilibria?

Potential games [Monderer & Shapley 96]

- A potential game is a game with a potential function p: $A \rightarrow \Re$ (A is set of all joint actions (pure strategy profiles, outcomes)) such that
- for all players i, all a_{-i}, all a_i, all a_i',
- $u_i(a_i, a_{-i}) - u_i(a_i', a_{-i}) = p(a_i, a_{-i}) - p(a_i', a_{-i})$

1, 3	6, 6
3, 5	3, 3

Potential function:

p(UL) = 0

p(UR) = 3

p(DL) = 2

p(DR) = 0

- Can you think of an algorithm for verifying whether a normal-form game is a potential game (i.e., for finding a potential function)?

Potential games always have a pure-strategy equilibrium

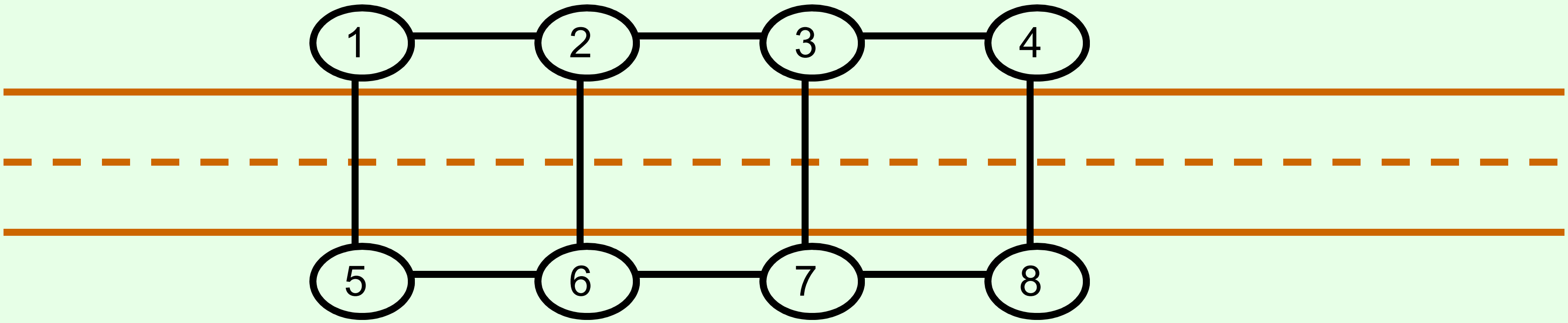
- Recall that $u_i(a_i, a_{-i}) - u_i(a_i', a_{-i}) = p(a_i, a_{-i}) - p(a_i', a_{-i})$
- Hence, let us simply choose $\text{argmax}_a\{p(a)\}$
- For any alternative a_i', $u_i(a_i', a_{-i}) - u_i(a) = p(a_i', a_{-i}) - p(a) \leq 0$, hence it is an equilibrium!
- More generally, the set of pure-strategy Nash equilibria is exactly the set of local maxima of the potential function
  - Local maximum = no player can improve the potential function by herself
- Easy algorithm for finding a pure-strategy equilibrium:
  - Start with any strategy profile
  - If a player is not best-responding, switch that player's strategy to a better response (must increase potential)
  - Terminate when no player can improve

Every congestion game is a potential game!

- Use potential $p(a) = -\Sigma_r \Sigma_{1 \leq i \leq \#(r, a)} c_r(i)$
  - One interpretation: the sum of the utilities that the agents would have received if each agent were unaffected by all later agents
- Why is this a correct potential function?
- Suppose you change actions
  - You stop using some resources (R-), start using others (R+)
- Change in your utility equals $\Sigma_{r \text{ in } R-} c_r(\#(r, a)) - \Sigma_{r \text{ in } R+} c_r(\#(r, a) + 1)$
- This is also the change in the potential function above
- Conversely, every potential game can be modeled as a congestion game
  - Proof omitted

Graphical games [Kearns et al. 01]

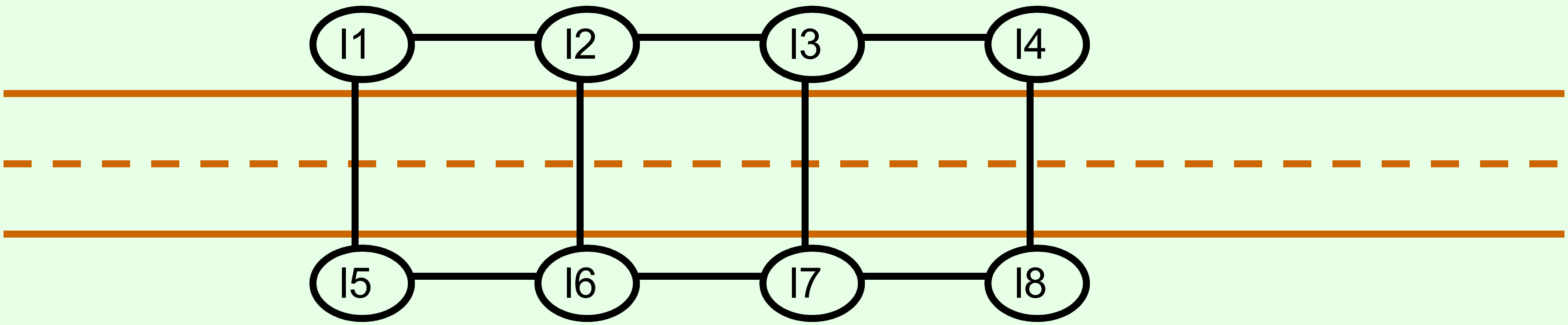
- Note: all games in this lecture have something to do with graphs, but only the following are considered "graphical games"
- Suppose players are vertices of a graph, and each agent is affected only by its neighbors' actions (and own action)
- E.g., physical neighbors on a road:



- Can write $u_2(a_1, a_2, a_3, a_6)$ rather than $u_2(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)$
  - If each agent has two actions, a table requiring $2^4 = 16$ numbers rather than $2^8 = 256$ numbers
- Graphical games can model any normal-form game (by using a fully connected graph)

Action-graph games [Bhat & Leyton-Brown 04]

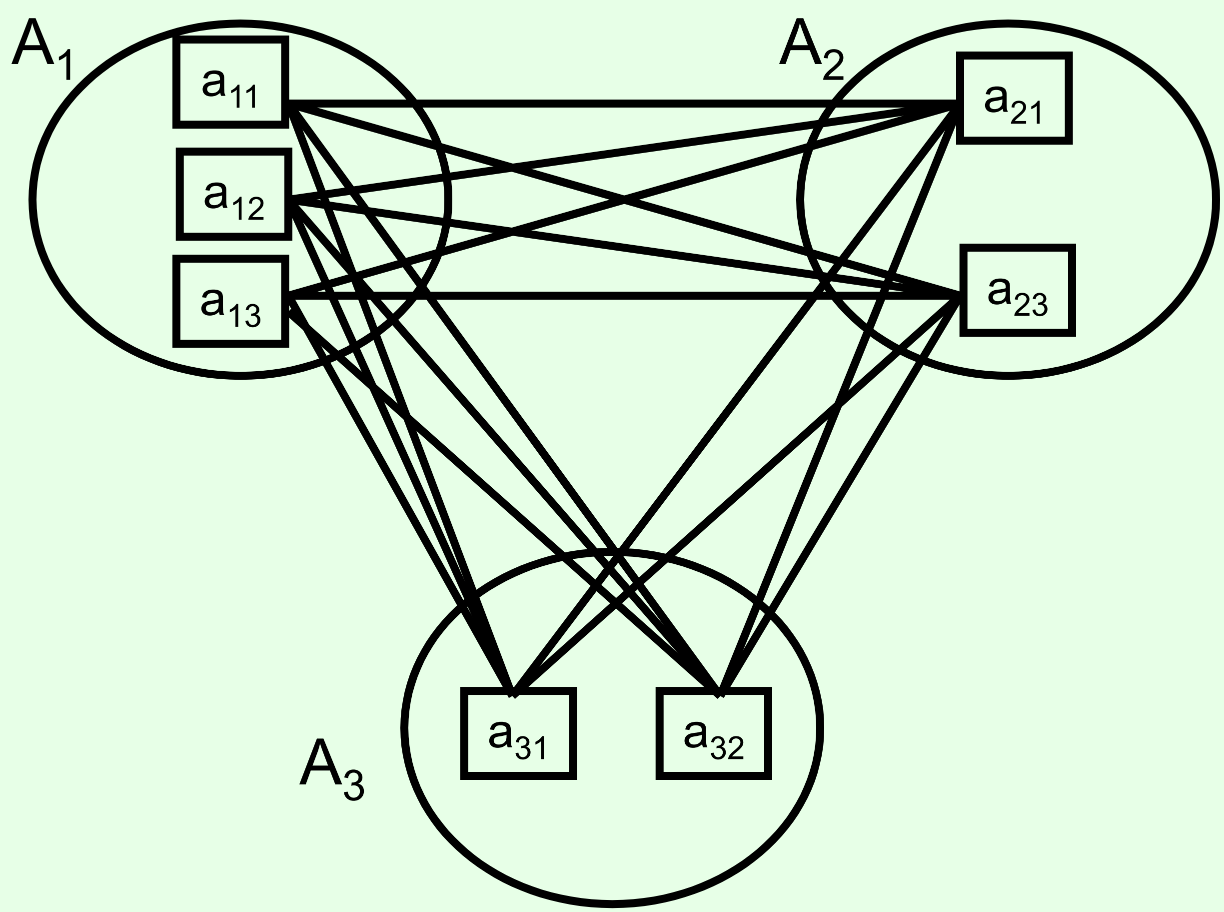
- Now there is a vertex for every action that a player may take
- E.g., action = where to sell hot dogs



- Players' can have distinct action sets, but they can overlap
  - E.g., perhaps some players cannot stand in every location
  - E.g., maybe player 1 lives on the right and can only make it to A_1 = {l3, l4, l7, l8}; but player 2 has a car and can make it to any location
- Your utility is a function of
  - Which vertex you choose,
  - How many other players choose your vertex,
  - For each neighbor of your vertex, how many players choose that neighbor

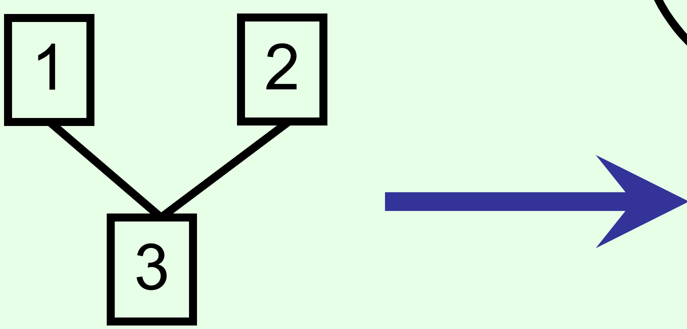
Any normal-form game can be expressed as an action-graph game

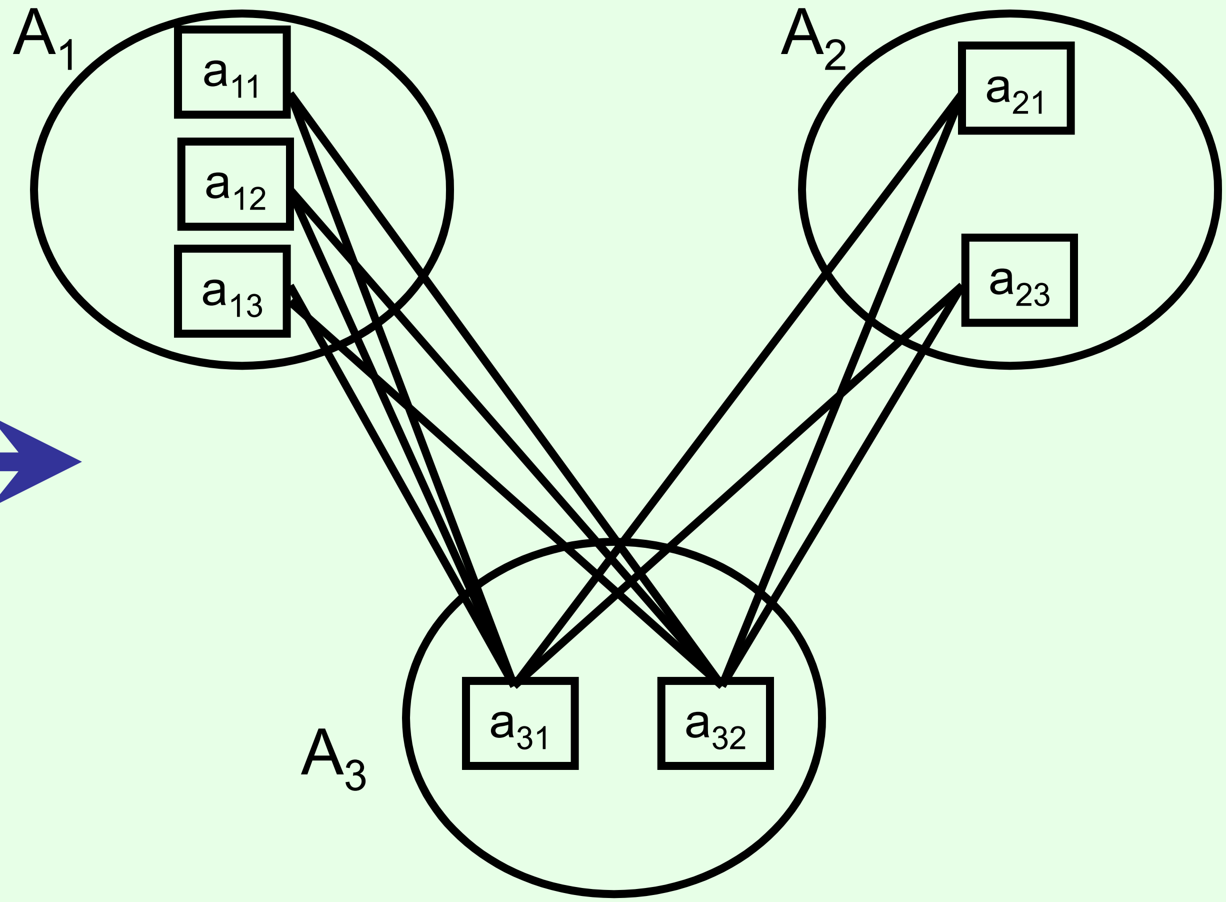
- Make action sets disjoint,
- Connect every action to every action by another player



Action-graph games can capture the structure of graphical games

- Omit edges between actions of players not connected in the graphical game





- But, AGGs can capture more structure than that: e.g., things like:
- "Player 1's utility does not depend on what player 2 does given that player 1 plays a_{12} (context-specific independence)

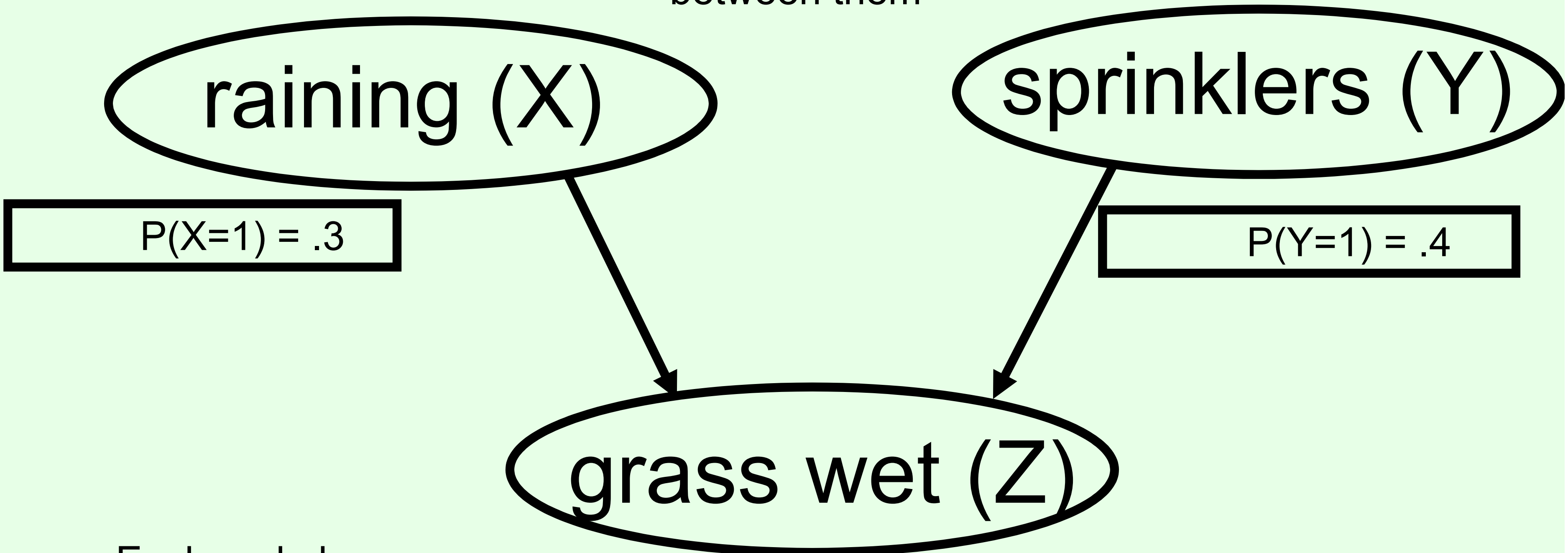
Computing expected utilities in AGGs

[Jiang & Leyton-Brown AAAI06]

- How do we compute the expected utility of playing a given action a_i, given the other players' mixed strategies σ_{-i}?
  - Key step in many algorithms for computing equilibria
- Observation: only care about neighboring actions of a_i
  - Lump other actions together into one big "irrelevant" action
- A configuration $D(a_i)$ specifies the number of (other) players that end up playing each action
  - Utility of playing a_i is a function of the configuration
  - Need to know the probability distribution over configurations
- Use dynamic programming
- Say $P_k(D(a_i))$ is the probability of $D(a_i)$ if only the first k other players play
- Say $D(D(a_i), a_j)$ is the configuration that results from adding one player playing a_j to $D(a_i)$
- Then $P_k(D(a_i)) = \Sigma_{D'(a_i),\, a_j:\, D(D'(a_i),\, a_j) = D(a_i)} P_{k-1}(D'(a_i))\, \sigma_k(a_j)$

Bayes nets: Rain and sprinklers example

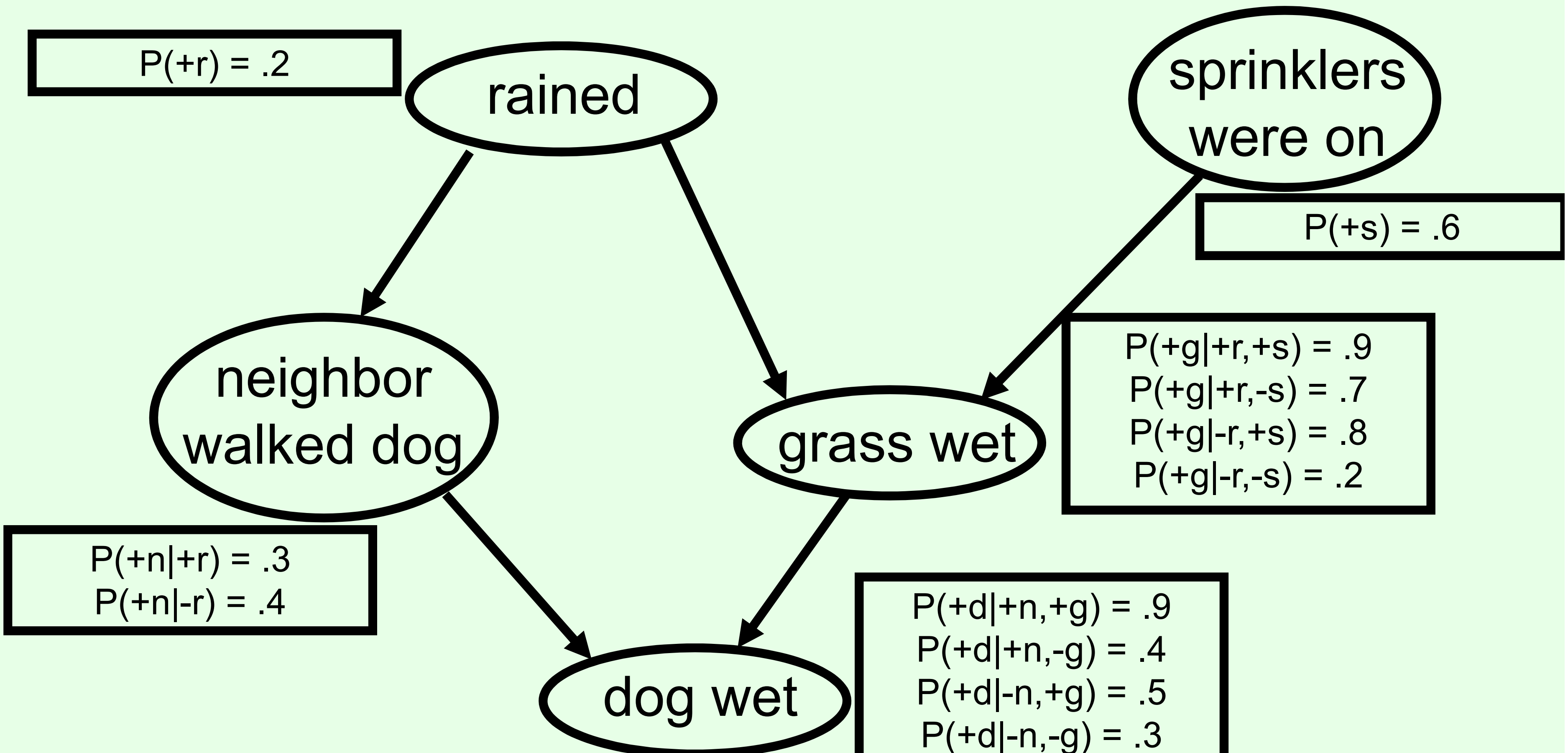
sprinklers is independent of raining, so no edge between them



Each node has a conditional probability table (CPT)

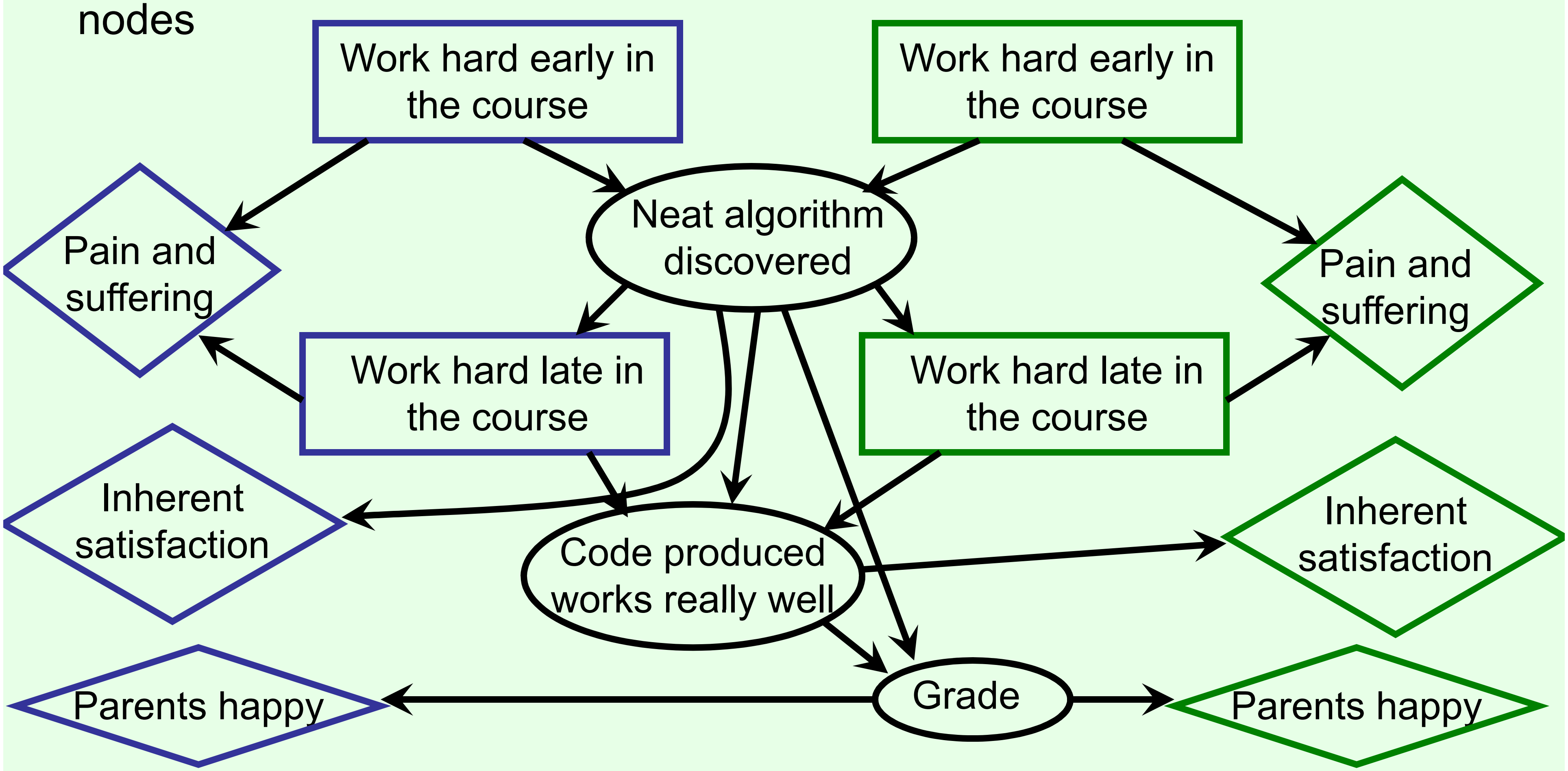
P(Z=1 | X=0, Y=0) = .1
P(Z=1 | X=0, Y=1) = .8
P(Z=1 | X=1, Y=0) = .7
P(Z=1 | X=1, Y=1) = .9

More elaborate rain and sprinklers example



MAIDs [Koller & Milch 03]

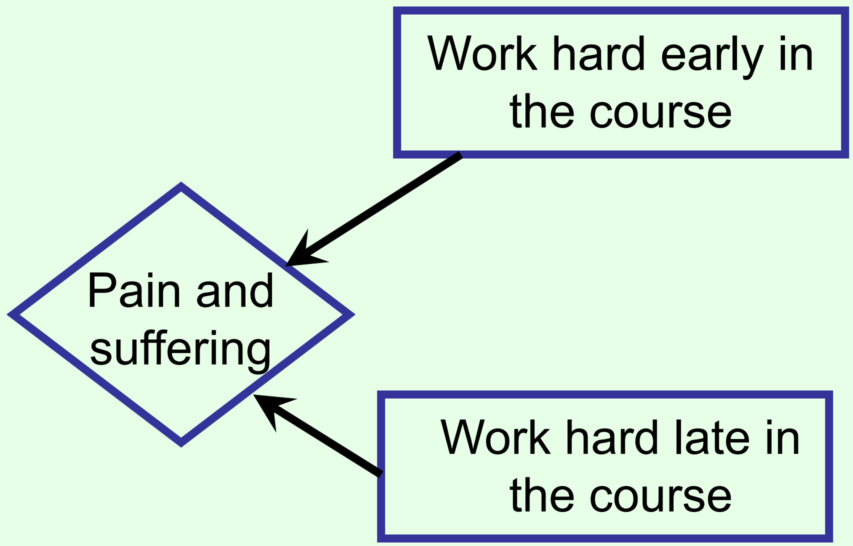
- MAID = Multi Agent Influence Diagram
- Example: two students working on a project together
- Blue student likes theory and algorithms, green student likes code that works well in practice
- Rectangles = decision nodes, ovals = random nodes, diamonds = utility nodes



MAIDs…

- The graph by itself is not enough:
- We need specifications of the utility functions…

$u_{p\&s}(WHE, WHL) = -30$
$u_{p\&s}(WHE, -WHL) = -20$
$u_{p\&s}(-WHE, WHL) = -20$
$u_{p\&s}(-WHE, -WHL) = 0$



- … and the conditional probabilities for random nodes

P(NAD | WHE, WHE) = .9
P(NAD | WHE, -WHE) = .8
P(NAD | -WHE, WHE) = .5
P(NAD | -WHE, -WHE) = 0