Logical Agents: Knowledge Representation II

CS3243: Introduction to Artificial Intelligence – Lecture 9a

20 March 2023

Contents

Reference: AIMA 4th Edition, Section 7.5

Administrative Matters

Upcoming…

- Deadlines
  - TA7 (released last week)
    - *Due in your Week 10 tutorial session*
    - *Submit the a physical copy (more instructions on the Tutorial Worksheet)*
  - Prepare for the tutorial!
    - Participation marks = 5%
  - Project 2 (released Week 6)
    - *Was due yesterday, Sunday (19 March), 2359 hrs*
    - *Late penalties now apply*
  - Project 3 (released last week)
    - *Due Week 12 Sunday (9 April), 2359 hrs*

- Week 11: Final Content Lecture
- Week 12: Lecture Slot = Project 3 Consultation (conducted over Zoom)

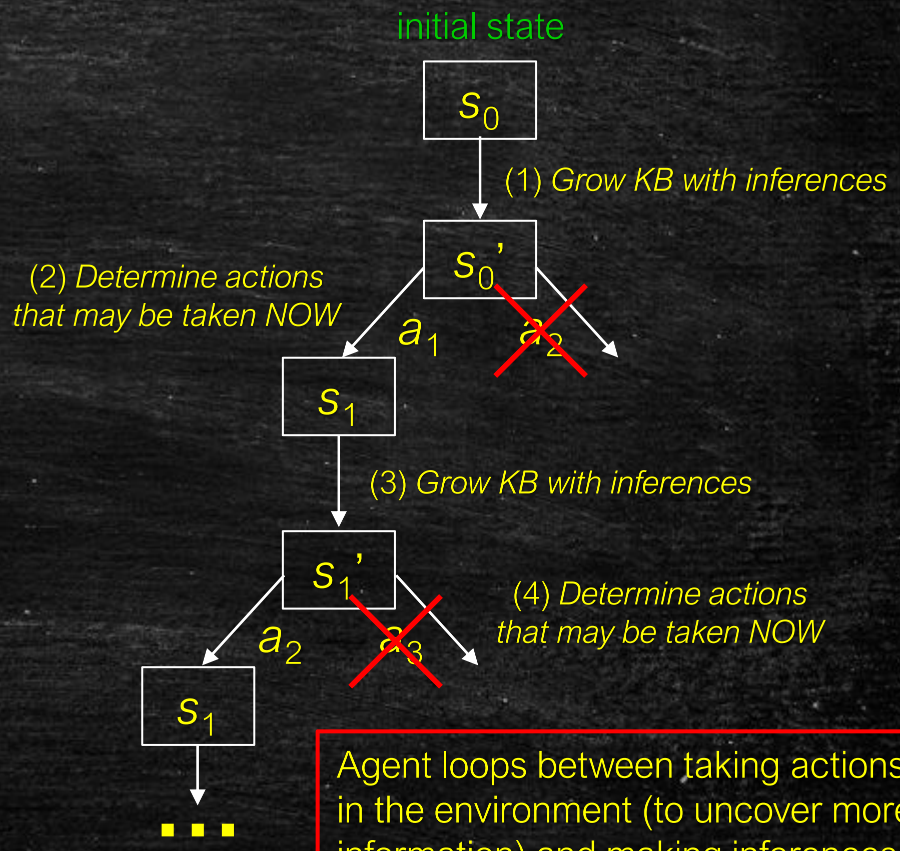
Recap on Logical Agents

Logical Agents

- Agent contains
  - Knowledge Base (KB)
    - Specified in some language (e.g., propositional logic)
  - Inference Engine (IE)
    - Determines sentences that will guide action choice, $\alpha_1, \alpha_2, \ldots, \alpha_k$
    - Uses an algorithm that infers α_i such the KB $\models \alpha_i$
- General algorithm
  - Pre-populate KB with domain knowledge
  - Each time step t:
    - Update KB with percepts
    - Use IE to make inferences
      - Update KB with inferences
      - Select action based on inferences
  - Take action and update KB with new state (current truth value assignments)

Logical Agents

- Logical Agent *cannot* plan an entire path to goal
  - Environment is only Partially Observable
- Assume operation as follows
  - Make inferences about environment
    - Assume query (α) to action (a_i) mappings
      - e.g., reflex agent with conditions based on KB
    - Developer designing the agent must formulate these mappings
  - Suppose $\alpha \Rightarrow a_i$
    - Agent uses the KB & IE to determine if a query (α) may be inferred, and if so, takes associated action (a_i)
      - Chosen action (a_i) is EXECUTED in environment (as we are no longer planning)
  - Loops...



Agent loops between taking actions in the environment (to uncover more information) and making inferences

Making Inferences

- Entailment (⊨)
  - KB ⊨ α means that M(KB) ⊆ M(α)
    - This says that all value assignments that satisfy the KB will also satisfy α
    - i.e., whenever KB is true, α is true
- Inference algorithm ($\mathcal{A}$)
  - Sound: (KB $\vdash_{\mathcal{A}}$ α) ⇒ (KB ⊨ α)
    - $\mathcal{A}$ only infers α that are valid
  - Complete: (KB ⊨ α) ⇒ (KB $\vdash_{\mathcal{A}}$ α)
    - $\mathcal{A}$ is able to infer all valid α
- Inference algorithm example: Truth Table Enumeration
  - Construct entire truth table for KB and α
  - Check (via DFS) that M(KB) ⊆ M(α)
  - i.e., every model of KB is a model of α

Truth Table Enumeration:
- Sound and Complete
- Time complexity $O(2^n)$
- Space complexity O(n)

Theorem Proving Methods

Proof Methods

- Model checking (special case of CSPs where domains are T/F)
  - Truth Table Enumeration (time complexity exponential in n)
  - Resolution (inference via proof)
- Applying inference rules (i.e., theorem proving)
  - Generate new sentences from old
  - Proof = sequential application of inference rules
    - Inference rules help deduce valid actions
    - Proof facilitates efficiency – ignores irrelevant propositions

Validity & Satisfiability

- A sentence α is valid if it is *true for ALL possible truth value assignments*
  - e.g., True, $A \lor \neg A$, $A \Rightarrow A$, $(A \land (A \Rightarrow B)) \Rightarrow B$
  - i.e., tautologies
- Validity is connected to entailment via the Deduction Theorem:
  - $(KB \vDash \alpha) \Leftrightarrow ((KB \Rightarrow \alpha)$ is valid $)$
- A sentence is satisfiable if it is *true for SOME truth value assignment*
  - e.g., $A \lor B$, C
- A sentence if unsatisfiable if it is *true for NO truth value assignments*
  - e.g., $A \land \neg A$
  - i.e., contradictions
- Satisfiability is connected to entailment via the following:
  - $(KB \vDash \alpha) \Leftrightarrow ((KB \land \neg\alpha)$ is unsatisfiable $)$
  - i.e., definition of Proof by Contradiction

i.e., a model exists for that sentence

i.e., no model exists for that sentence

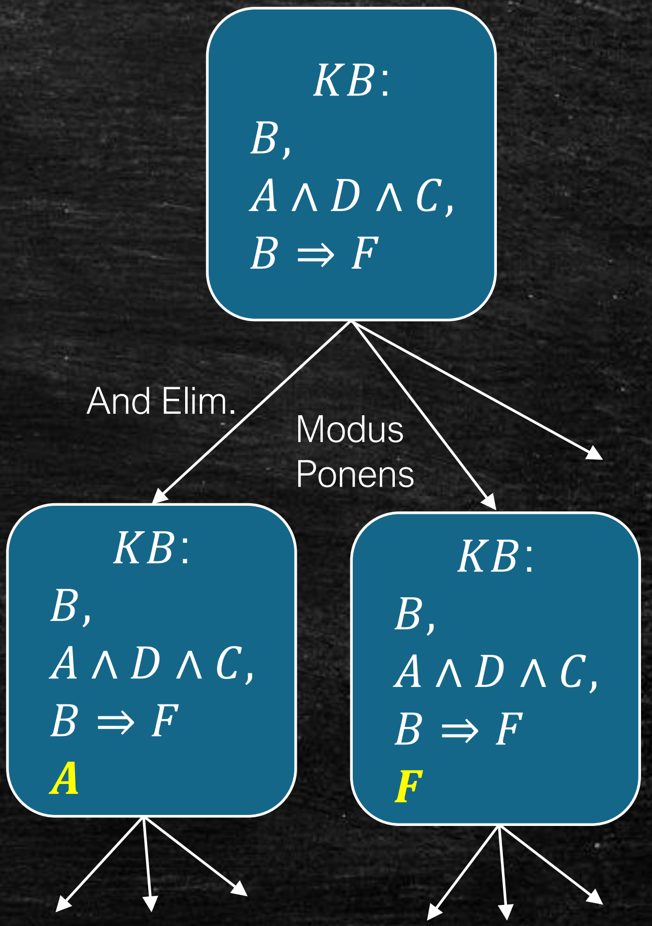
- $A \Rightarrow B \equiv \neg A \lor B$
- $\neg(\neg A \lor B) \equiv A \land \neg B$
- Showing that $(A \land \neg B)$ is unsatisfiable shows that the negation, $\neg A \lor B$ is valid!

Inference Algorithms: Application of Inference Rules

- Search for more knowledge (growing the KB)
  - Equivalent to a search problem
    - States: Versions of the KB (e.g., initial state is initial KB)
    - Actions: Application of inference rules
    - Transition: Update KB with an inferred sentence (may not be target query α)
    - Goal: KB contains sentence to (dis)prove (e.g., given query α)

- Examples of inference rules
  - And-Elimination (AE): e.g., $a \wedge b \vDash a$; $a \wedge b \vDash b$
  - Modus Ponens (MP): e.g., $a \wedge (a \Rightarrow b) \vDash b$
  - Logical Equivalences: e.g., $(a \vee b) \vDash \neg(\neg a \wedge \neg b)$

How does this relate to Truth Table Enumeration?



Resolution

Resolution for Conjunctive Normal Form (CNF)

- CNF = conjunction of disjunctive sentences
  - e.g., $(x_1 \vee \neg x_2) \wedge (x_2 \vee x_3 \vee \neg x_4)$
- Resolution
  - Method of simplifying KB to prove entailment of query α
  - Specifically
    - Given KB: $R_1 \wedge R_2 \wedge \ldots \wedge R_n$
    - If a literal, x, appears in R_i and its negation, $\neg x$, appears in R_j, where $R_i, R_j \in$ KB, it can be removed from both

resolvent

$$\frac{(x_1 \vee \cdots \vee x_m \vee x) \wedge (y_1 \vee \cdots \vee y_k \vee \neg x)}{(x_1 \vee \cdots \vee x_m \vee y_1 \vee \cdots \vee y_k)}$$

  - Resolution under propositional logic
    - Sound
    - Complete

KB: (P or x) and (Q or (not x))
α: (P or Q) must hold?

x=T, Q must be True for KB to hold
x=F, P must be True for KB to hold

So (P or Q) must hold

Verify with truth table as an exercise
(note: we want M(KB) ⊆ M(α))

Some Rules for Conversion to CNF

1. Convert $\alpha \Leftrightarrow \beta$ to $\boxed{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$
2. Convert $\alpha \Rightarrow \beta$ to $\neg\alpha \vee \beta$
3. Expand $\neg$ using De Morgan and double negation
   a. Convert $\neg(\alpha \vee \beta)$ to $\boxed{\neg\alpha \wedge \neg\beta}$
   b. Convert $\neg(\alpha \wedge \beta)$ to $\neg\alpha \vee \neg\beta$
   c. Convert $\neg(\neg\alpha)$ to α
4. Convert $\big(\alpha \vee (\beta \wedge \gamma)\big)$ to $\boxed{(\alpha \vee \beta) \wedge (\alpha \vee \gamma)}$

Each of these conversions produces two rules in the KB (the others just one)

Resolution Algorithm

- Utilises proof by contradiction – tries to show that KB ∧ ¬α is unsatisfiable

```
function PL-RESOLUTION(KB, α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
   new ← { }
   while true do
      for each pair of clauses Ci, Cj in clauses do
         resolvents ← PL-RESOLVE(Ci, Cj)
         if resolvents contains the empty clause then return true
         new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```

If cannot be resolved further and not empty clause – cannot infer α

What does an empty clause imply??

Suppose we have a KB as follows:

$$\frac{(x_1 \vee \cdots \vee x_m \vee x) \wedge (y_1 \vee \cdots \vee y_k \vee \neg x)}{(x_1 \vee \cdots \vee x_m \vee y_1 \vee \cdots \vee y_k)}$$

And the algorithm slowly removes literals:

$$\frac{(\cancel{x_1 \vee \cdots \vee x_m} \vee x) \wedge (\cancel{y_1 \vee \cdots \vee y_k} \vee \neg x)}{(\cancel{x_1 \vee \cdots \vee x_m \vee y_1 \vee \cdots \vee y_k})}$$

Eventually, there is nothing in the KB.

KB indicates the disjunction of no literals holds. A disjunction is True only when at least one literal is true. So, whole KB is False here – i.e., the query ¬α is unsatisfiable.

We may infer α (via proof by contradiction).

Resolution Algorithm

- Summary
  - Make a clause list – i.e., copy of KB specified in CNF including negation of query, ¬α
    - Use conversion rules to convert KB to CNF
  - Repeatedly resolve two clauses from clause list
    - Add resolvent to clause list
  - Keep doing this till empty clause found or no more resolutions possible
    - If empty clause then can infer α
    - If no more resolutions and not empty clause then cannot infer α

Why is Resolution under Propositional Logic Sound and Complete?

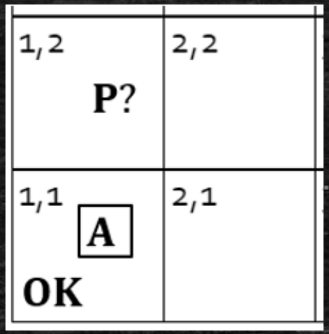
Soundness:
- Each resolvent is implied by generating clauses
- If ∅ is found, then (KB ∧ ¬α) is unsatisfiable

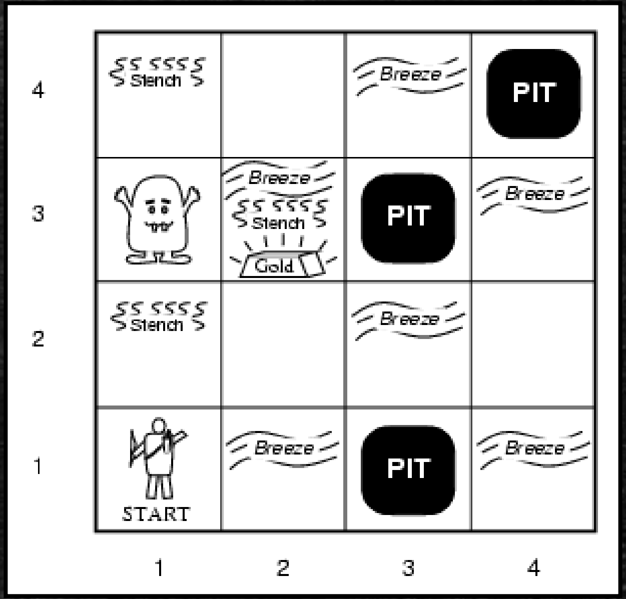
Completeness:
- Based on the idea of resolution closure – set of all clauses derivable
- Not covered in CS3243
- Refer to AIMA 4th Edition pp. 228-229

Resolution Example

Resolution Example: Back to Wumpus World

- Assume that agent is at (1,1) in Wumpus World
  - And we wish to make inferences about a pit at (1,2)
- KB
  - $\left(B_{1,1} \Leftrightarrow \left(P_{1,2} \vee P_{2,1}\right)\right) \wedge \left(\neg B_{1,1}\right)$
  - i.e., we know
    - R_a: rule for breezes
    - R_b: no breeze at (1,1)
- α
  - $\neg P_{1,2}$
  - i.e., want to know if we can move to (1,2)
- Can we infer α?
  - Use the resolution algorithm to determine if (KB ⊨ α)
  - i.e., use (KB ⊨ α) ⇔ ((KB ∧ ¬α) is unsatisfiable)

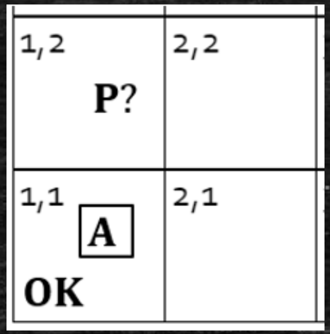


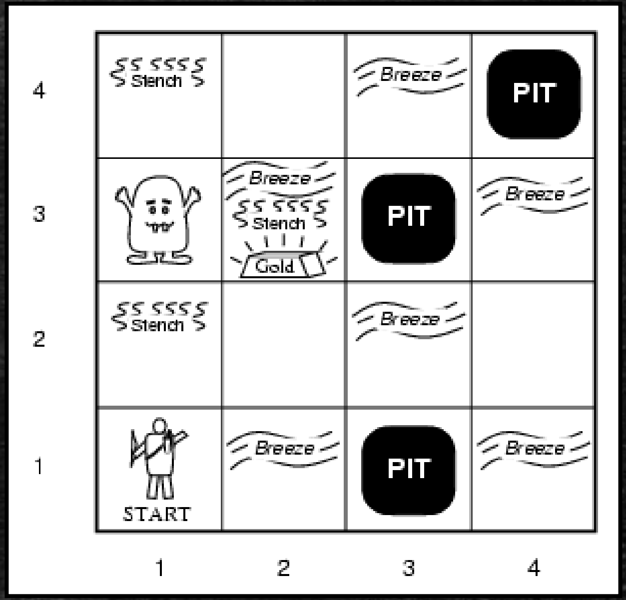


Resolution Example: Back to Wumpus World

- Given
  - $KB = \neg B_{1,1} \wedge B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  - $\alpha = \neg P_{1,2}$
- Step 1 – Form clause list (over $KB \wedge \neg\alpha$)
  - $(\neg B_{1,1}) \wedge (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge (P_{1,2})$
- Step 2 – Convert clause list to CNF
  - $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
    - $B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})$
    - $(P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$

$(\neg B_{1,1})$, $(P_{1,2})$ already in CNF (literals)





- Step 2a
  - $B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})$
    - $\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})$
    - $\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$

Now in CNF

- Step 2b
  - $(P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
    - $\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1}$
    - $(\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1}$
    - $(\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

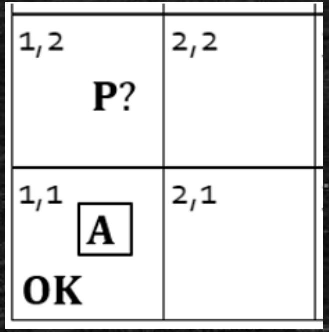
Now in CNF – as 2 rules

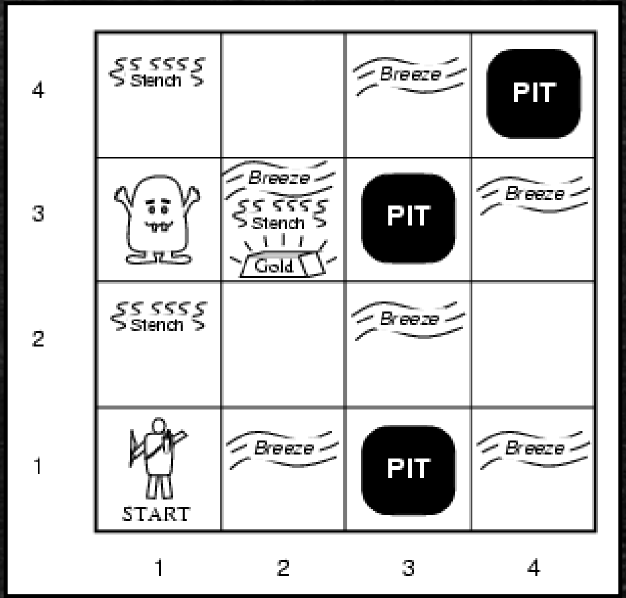
Resolution Example: Back to Wumpus World

- Clause list (in CNF)
  - R_1: $\neg B_{1,1}$
  - R_2: $P_{1,2}$
  - R_3: $\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}$
  - R_4: $\neg P_{1,2} \lor B_{1,1}$
  - R_5: $\neg P_{2,1} \lor B_{1,1}$
- Step 3 – Pick two rules and resolve via

$$\frac{(x_1 \lor \cdots \lor x_m \lor x) \land (y_1 \lor \cdots \lor y_k \lor \neg x)}{(x_1 \lor \cdots \lor x_m \lor y_1 \lor \cdots \lor y_k)}$$

- Step 3a – Reduce R_2 and R_4
  - R_6: $B_{1,1}$
- Step 3b – Reduce R_1 and R_6
  - $\emptyset$





Proof by contradiction that KB ⊨ α
i.e., α holds when KB holds; we can infer $\alpha = \neg P_{1,2}$

Where Does α Come From?

Regarding the Query α

- Inference algorithms show that we can infer α
- Where do we get α?
  - Recall that Logical Agent program

```
function KB-AGENT(percept) returns an action
   persistent: KB, a knowledge base
               t, a counter, initially 0, indicating time

   TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
   action ← ASK(KB, MAKE-ACTION-QUERY(t))
   TELL(KB, MAKE-ACTION-SENTENCE(action, t))
   t ← t + 1
   return action
```

reasoning on what should be done

construct a sentence
relating to an action to take

  - Inference algorithms ($\mathcal{A}$) assumes α is given and decides if $KB \vDash \alpha$
  - When discussing soundness and completeness of $\mathcal{A}$, we consider which among any given/input α that will satisfy $KB \vDash \alpha$

Questions about the Lecture?

- Was anything unclear?
- Do you need to clarify anything?

- Ask on Archipelago
  - Specify a question
  - Upvote someone else's question

Invitation Link (Use NUS Email --- starts with E)
https://archipelago.rocks/app/resend-invite/73861543030

Uncertainty

CS3243: Introduction to Artificial Intelligence – Lecture 9b

20 March 2023

Logical Agents & Uncertainty

Dealing with Uncertainty

- Example – Let A_t denote an autonomous taxi agent's action
  - A_t : leave for airport t minutes before a flight
  - Will A_t get me to the airport on time?
- Sources of uncertainty
  - Partial observability (e.g., road state, other drivers' plans)
  - Noisy sensors (e.g., traffic reports, fuel sensor)
  - Uncertainty in action outcomes (e.g., flat tire, accident)
  - Complexity in modelling and predicting traffic (e.g., congestion)
- Logical agent will either
  1. Risk Falsehood – e.g., A_{25} *will* get me there on time
  2. Reaches weaker conclusion – e.g., A_{25} ***will*** get me there on time if
     a. There is no accident on the bridge
     b. It does not rain
     c. I do not get a flat tire

Under logic (certainty), you may require A_{1440} to reach the airport on time (i.e., stay overnight)

Better to consider the probability of being on time with more reasonable t ...

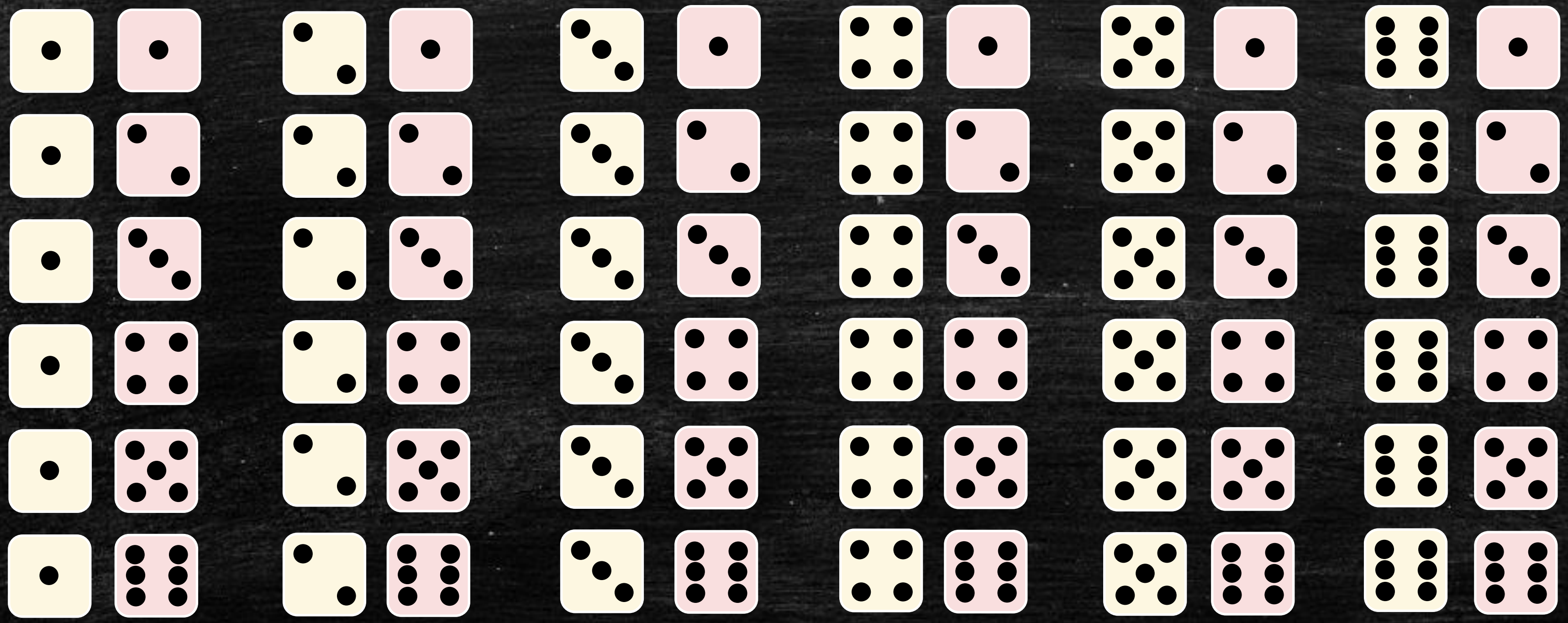
Probability

Random Variables

- Random variable (X)
  - Quantifies an outcome of a random occurrence
    - e.g., outcome of a coin toss, die roll, or COVID-19 ART
- Domains (D_X)
  - Boolean: coin is either heads of tails (i.e., True or False)
  - Discrete: a die can have values {1, ..., 6}
- Events
  - Subsets of domains
    - e.g., Heads(X): coin flipped to heads
    - e.g., Even(X): die has value ∈ {2,4,6}

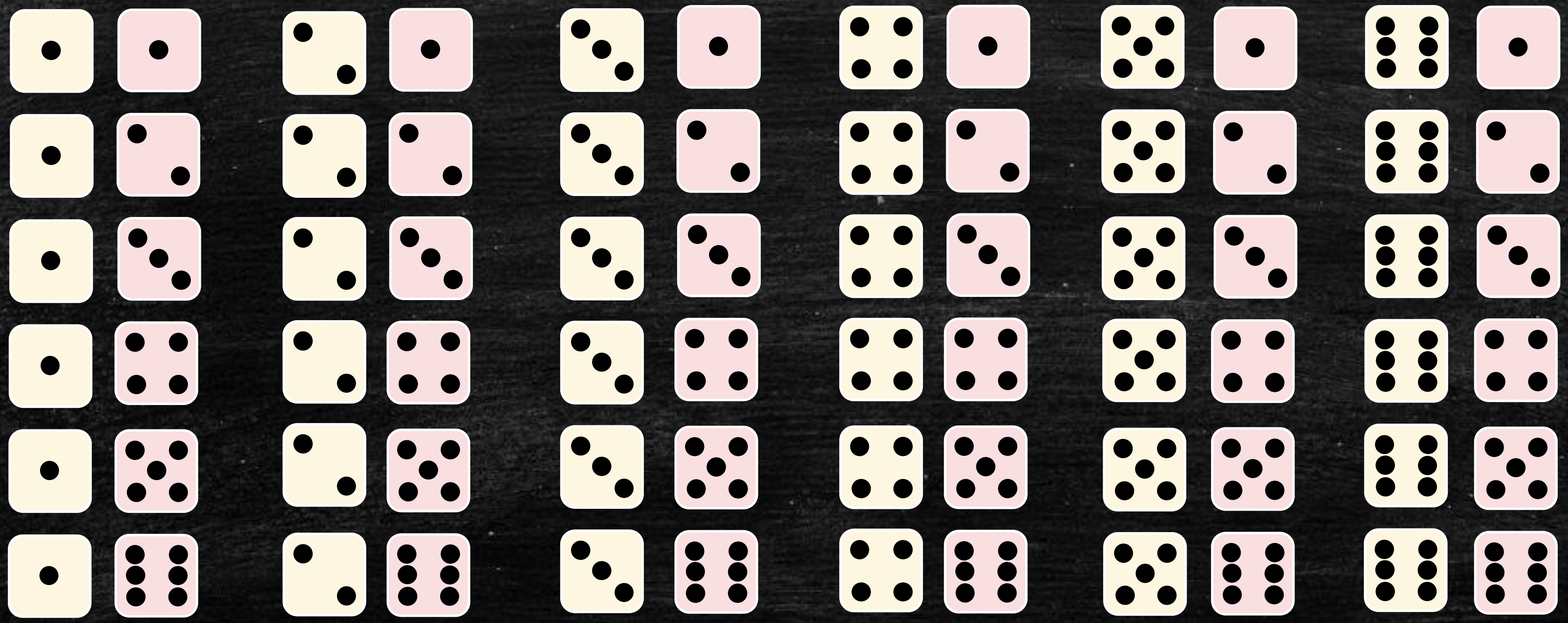
Events

- Atomic / singleton event (possible world)
  - An assignment of a value to each random variable
- Example – we roll two different dice



Events

- Yellow die = X_1
- Pink die = X_2
- Event: $X_1 + X_2 = 8$



Axioms of Probability

- Let X be a random variable with finite domain D_X
- A probability distribution over D_X assigns a value $p_X(v) \in [0,1]$ to every $v \in D_X$ such that

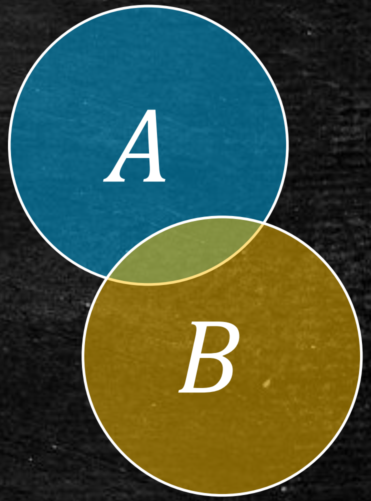
$$\sum_{v \in D_X} p_X(v) = 1$$

- For any event $A \subseteq D_X$, we have

$$\Pr[X \subseteq A] \equiv \Pr_X[A] = \sum_{v \in A} p_X(v)$$

- In particular

$$\Pr[A] + \Pr[B] = \Pr[A \cap B] + \Pr[A \cup B]$$



Joint Probability

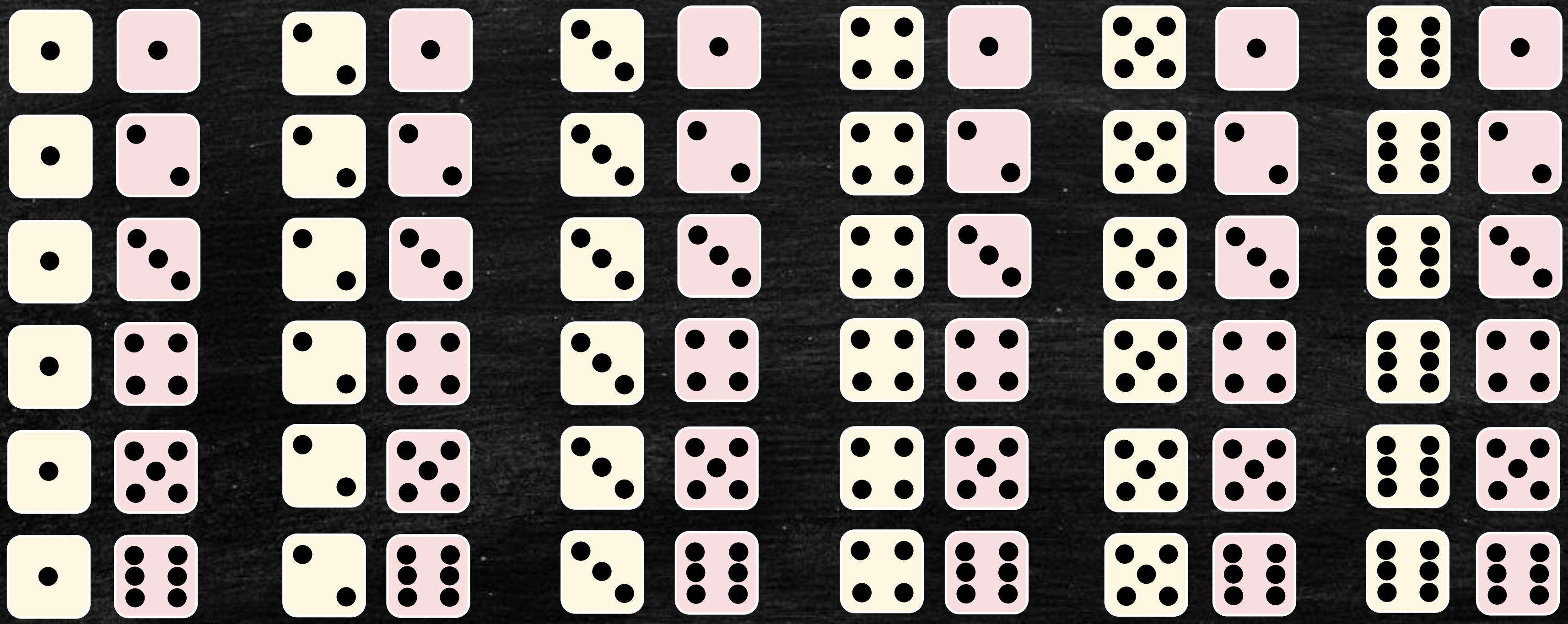
- Given two random variables X and Y
  - The joint probability of an atomic event $(x, y) \in D_X \times D_Y$ is $p_{X,Y}(x, y) = \Pr[X = x \wedge Y = y]$
- In particular $p_X(x) = \sum_{y \in D_Y} p_{X,Y}(x, y)$
- Example

Income (in SGD) / AGE	15-24	25-34	35-44	45-54	55-64	65+
$< S\$2500$	0.062	0.051	0.037	0.019	0.015	0.039
$S\$2500 - S\$5000$	0.078	0.068	0.061	0.057	0.031	0.053
$> S\$5000$	0.015	0.051	0.094	0.119	0.111	0.039

$\Pr[Age = (25 - 34)] = 0.051 + 0.068 + 0.051 = 0.17$

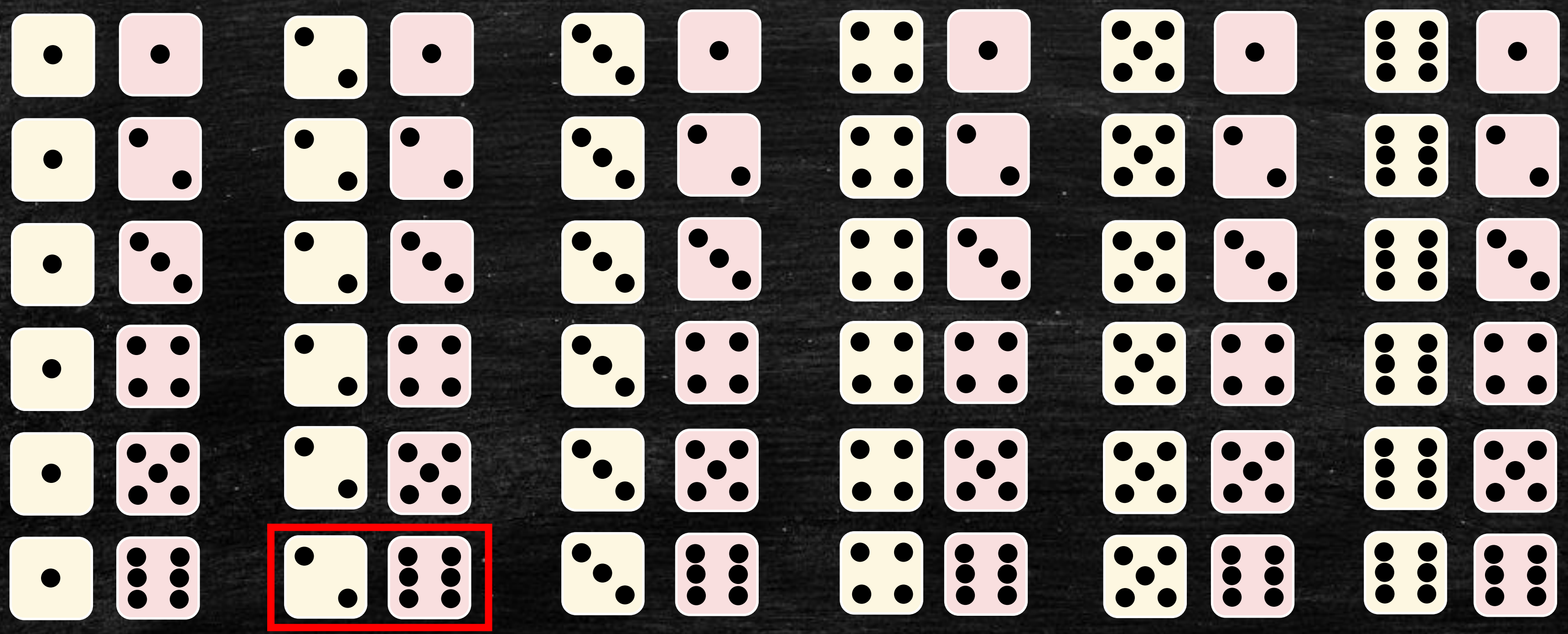
Conditional Probability

- The probability that an event occurs, given that some other event occurs
- Example – rolling 2 dice; $\Pr[X_1 = 2] = \frac{6}{36}$



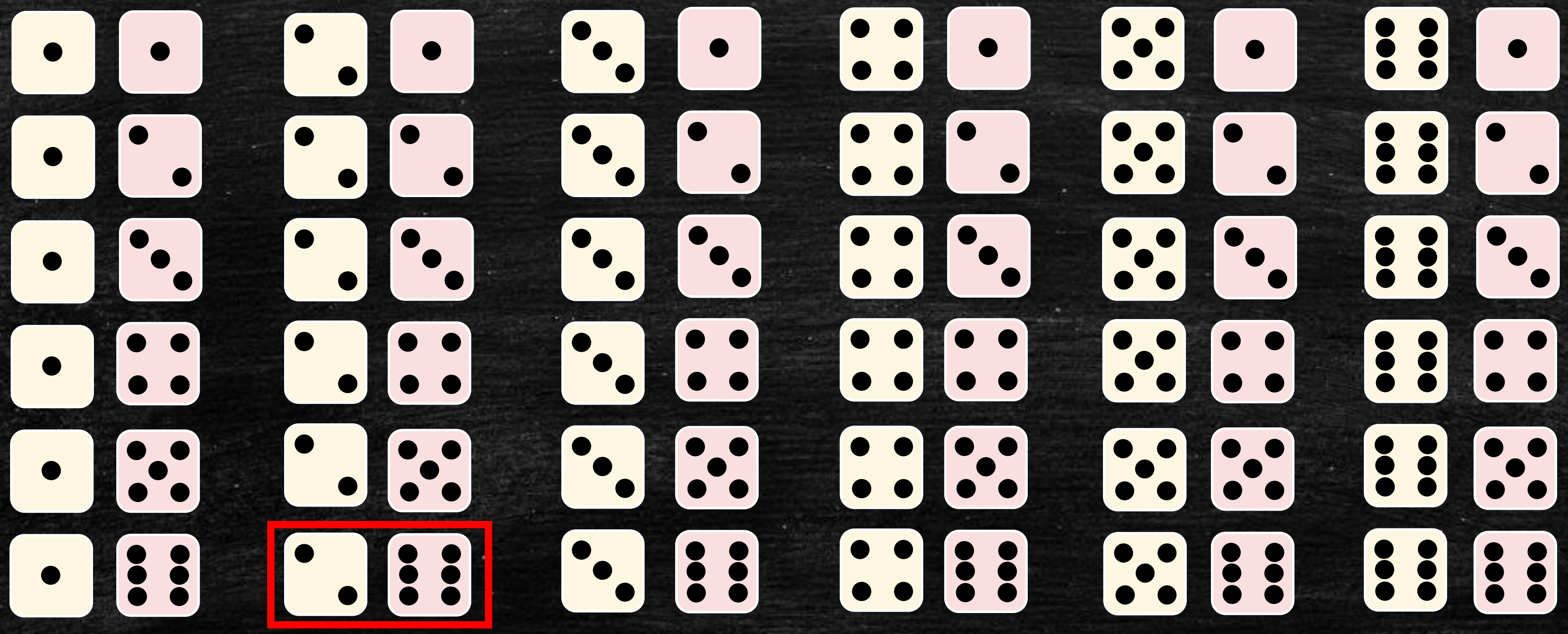
Conditional Probability

- The probability that an event occurs, given that some other event occurs
- Example – rolling 2 dice; $\Pr[X_1 = 2 \mid X_1 + X_2 = 8] = \frac{1}{5}$



Conditional Probability

- The probability that an event occurs, given that some other event occurs
- Example – rolling 2 dice; $\Pr[X_1 + X_2 = 8 \mid X_1 = 2] = ?\frac{1}{6}$



Conditional Probabilities & Bayes Rule

- $\Pr[A \mid B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$ assuming that $\Pr[B] > 0$

Note:
Pr[A | B] = Pr[A ∧ B] / Pr[B] --- (1)
Pr[B | A] = Pr[B ∧ A] / Pr[A] --- (2)

Also, we know:
Pr[A ∧ B] = Pr[B ∧ A] --- (3)

From (2) and (3), we have:
Pr[A ∧ B] = Pr[B | A] . Pr[A] --- (4)

And thus from (4) and the definition above, we have Bayes Rule:
P[A|B] = (P[B|A].P[A]) / P[B]

- Bayes rule: $\Pr[\,A \mid B\,] = \frac{\Pr[B|A]\,\Pr[A]}{\Pr[B]}$
- Example: $\Pr[X_1 = 2 \mid X_1 + X_2 = 8] = ?$

$$= \frac{\overset{1/6}{\Pr[X_1+X_2=8|X_1=2]} \,.\, \overset{1/6}{\Pr[X_1=2]}}{\underset{5/36}{\Pr[X_1+X_2=8]}} = \frac{1}{5}$$

Next week, we will look at various applications of Bayes Rule

Questions about the Lecture?

- Was anything unclear?
- Do you need to clarify anything?

- Ask on Archipelago
  - Specify a question
  - Upvote someone else's question

Invitation Link (Use NUS Email --- starts with E)
https://archipelago.rocks/app/resend-invite/73861543030