L01 – Introduction and Set-based Inference

AIMA4e: Ch 6.1; Optional: Ch 1, 2

What you should know after this lecture

- Basic logistics of 6.4110 Spring 2026
- The Rational-Agent view of AI
- Agent domains and where to find them
- Inference with set-based uncertainty
    - Constraint-satisfaction problem formulation
    - Backtracking search

6.4110 Course staff

- Leslie Kaelbling (lpk@mit.edu)
- Anushka Aggarwal (UTA) (anushka6@mit.edu)
- Sunshine Jiang (GTA) (sunsh16e@mit.edu)
- David Koplow (GTA) (dkoplow@mit.edu)
- Alicia Lin (UTA) (ayl27@mit.edu)
- Divya Shyamal (GTA) (dshyamal@mit.edu)
- Ellery Stahler (GTA) (ellerys@mit.edu)
- Tiffany Wang (UTA) (twangst@mit.edu)
- Ephraim Wu (UTA) (filbertw@mit.edu)

6.4110 Course logistics

- Lecture: M, W 11 – 12:30
- Problem session: F 11 – 12 : detailed worked examples, questions answered
- Calendar on cat-soop: `https://airr.mit.edu`
- Piazza: Ask all questions here. Can be private to staff.
- Exams:
  - 3 in-class quizzes: Wed Feb 25, Wed March 18, Mon April 27
  - Final exam scheduled by registrar
- Homework: Weekly via cat-soop due Mondays at midnight A score of 75% on HW will give you 100% of the HW grade component; no further credit for doing more; you decide what is most helpful to your understanding
- 5 mini-projects: More open-ended, submit written answers on Gradescope
- Collaboration: Homework and MPs are for learning; fine to study together, use AI, but be sure to answer independently
- More info (grading, extensions, office hours) on cat-soop

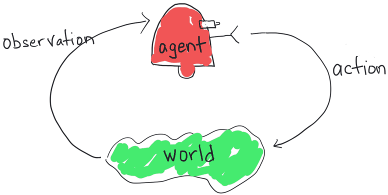
Course content

- Prereqs:
  - Probability (discrete and continuous)
  - Algorithms and Python (you can confidently go from pseudo-code to implementation and debug systematically)
- Focus on knowledge representation and reasoning methods of AI
- Complementary to machine learning

We like books!

- *Artificial Intelligence: A Modern Approach* (AIMA4e), Russell and Norvig `aima.cs.berkeley.edu`
- *Algorithms for Decision Making*, (KAlg), Kochenderfer `algorithmsbook.com`

Agents



"Agentic AI" can mean

- This
- Systems with subcomponents that are themselves "agents"

- **agent**: program with an ongoing feedback connection to an external environment (e.g., robot, smart house, Siri, banking advisor)
- generally $\pi : (O, A)^* \rightarrow A$ is a program with "state" or "memory" that maps the history of actions and observations to the next action

Policies for agents

How can we (engineers or nature) find a good policy for an agent? A policy is allowed to have memory—it could perform learning or search or solve differential equations or be made out of rubber bands. Anything (physically realizable) is fair as long as it maps histories of observations into the next action.

- Assume we have a specification (or a lot of examples) of a distribution of worlds where our agent is supposed to operate
- Assume also an objective function that measures how well the policy is performing
- Our job as engineers is to find the best policy, in expectation over possible environments, for this agent. But, how?

How to design policies for agents?

1. Be really smart, think hard, and write a program.
2. Systems-level “offline” rational design
   - Write down space of solutions
   - Write down objective function
   - Use search algorithm to find good solution
     Most applications of robot/agent learning fit here
3. Design of agent that is, itself, rational “online”
   - Be somewhat smart and construct a program for a rational agent that:
   - represents its knowledge about the environment and
   - reasons about what actions to take

Rational agents

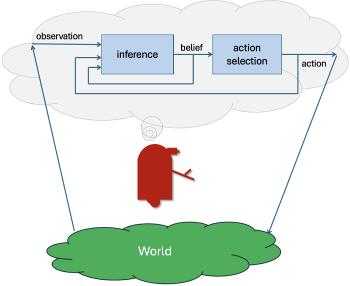
Principle of rationality: Select actions that will maximize expected future utility

- **expected**: take expectations over uncertainty about current environment state and/or outcomes of actions
- **future**: will consider one-step, finite, and infinite future horizons
- **utility**: Assume the agent has a scalar measure of utility or desirability of states of the external environment

Rationality is not the same as:

- omniscience or clairvoyance
- success

Rational agent architecture



Decompose into two main components, connected by “belief”:

- **belief**: the agent’s knowledge about the state of the external world (can be a point estimate, a set, a distribution)
- **inference**: algorithm for taking the history of actions and observations and computing a belief
- **action selection**: algorithm for selecting the next action, given current belief

Connections to learning-based systems

- Chain-of-thought and tree-of-thought, etc., encourage LLMs to do some informal test-time "inference"
- Combining search with learned policies is key to state-of-the-art game-playing systems (A*, MCTS, game trees, which we will study)
- Alpha-Geometry nearly IMO Gold standard by using a formal reasoning engine as a component (Horn-clause proof, which we will study)
- "Agentic" AI systems often use LLMs to select actions in an external world (e.g., buy/sell transactions) based on history of previous inputs and actions

An agent's model of its environment

This is the most general case—we'll consider some simplifications

- $\mathcal{S}$: set of world states
- $\mathcal{A}$: set of actions of the agent
- $\mathcal{O}$: set of possible observations the agent can make
- $P(\mathcal{S}_{t+1} \mid \mathcal{S}_t, \mathcal{A}_t)$: transition model : distribution over the world state at time $t+1$ given the world state at time t and the action taken by the agent at time t
- $P(O_t \mid \mathcal{S}_t)$: observation model : distribution over observations the agent might make at time t given the world state at time t

"All models are wrong, but some are useful." – George Box

Some properties of world models

- fully versus partially observable: agent is not certain about current state s_t
- discrete versus continuous: states, actions, observations, time
- static versus dynamic: can world state change while agent is "thinking?"
- episodic versus sequential: important for framing some learning problems
- deterministic versus stochastic: transitions and/or observations
- single versus multiple agents: connections to game theory

Representation and reasoning

- Representation: information stored in a computer—specific to the problem at hand
  - **syntax**: a “language” for encoding information in a computer
  - **semantics**: an intended relationship between a piece of syntax inside the machine and the state of the world outside the machine
- Reasoning: (inference) semantics-preserving manipulation of a syntactic representation—general-purpose, domain-independent
  - Given observations and prior knowledge, make conclusions about the world state and, from there, about which action to select.
  - *Planning*, specifically, is reasoning about action sequences.

“All our knowledge begins with the senses, proceeds then to the understanding, and ends with reason. There is nothing higher than reason.” –Kant

Compositional representation

A compositional representation has:

- syntax that is formed from a small set of primitive components and a set of rules for putting them together
- semantics: meaning of a the whole representation can be formed by combining meanings of the components

"The infinite use of finite means." –Alexander von Humboldt, on language

Representation and reasoning

Knowing some information about:

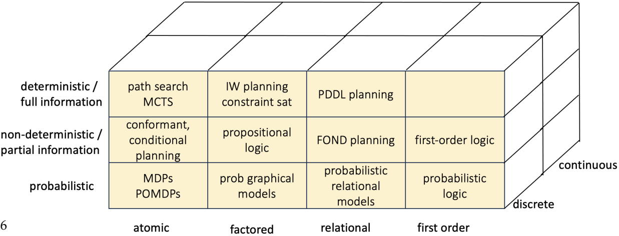
- Current world state
- Transition model
- Objective

Infer one or more of:

- Some aspect of world state
- What action to take next

Methods depend on

- How much information is known about current and future states
- Representational method



	atomic	factored	relational	first order
deterministic / full information	path search MCTS	IW planning constraint sat	PDDL planning	
non-deterministic / partial information	conformant, conditional planning	propositional logic	FOND planning	first-order logic
probabilistic	MDPs POMDPs	prob graphical models	probabilistic relational models	probabilistic logic

Representation styles

How do we describe a state?

- **atomic**: s_{44} (element of some enumeration)
- **factored**: $\langle 4, 5.2, '\mathtt{hippo}' \rangle$ (vector of attributes); embedding vector
- **relational**: $\{\mathit{bigger}(o_1, o_2), \mathit{resting}(o_2, o_2), \mathit{size}(o1) = 3.2\}$ (set of properties and relations on "objects"; also, interpretable as a graph)
- **first-order**: $\underline{\text{all the bacteria in the jar are dead}}$ $\forall x. \mathit{bacterium}(x) \wedge \mathit{in}(x, \mathit{jar}) \rightarrow \mathit{dead}(x)$

Different representations offer different opportunities for

- Expressing partial information
- Generalization

Connections to machine learning

Machine learning is everywhere! Probably most of you have studied it in some form. Many connections to what we will be studying:

- Learning the models / “knowledge” that our search or inference algorithms operate on
- Learn heuristics or other search control information to speed up inference and planning
- Learn problem-specific functions for quickly computing solutions to a space of common queries

Trade-offs between model-based and model-free methods

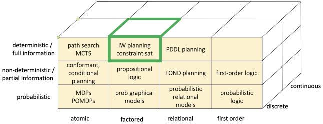
- Model-based often generalizes much more effectively
- Model-free methods not hampered by poor choices of model

Course outline

1. Inference
   - 1.1 Enumerative, factored, set-based belief
   - 1.2 Enumerative, factored, probabilistic belief
     (Propositional and first-order logic belief) $\rightarrow$ later
2. Action selection
   - 2.1 Exact belief
     - 2.1.1 Deterministic dynamics (path-search problems)
     - 2.1.2 Stochastic dynamics (MDPs)
   - 2.2 Uncertain belief
     - 2.2.1 Deterministic or set-based dynamics and observations
     - 2.2.2 Stochastic dynamics and observations (POMDPs)
   - 2.3 Other agents: game theory
3. Formal proof in propositional and first-order logic

Factored states and information

Factored, discrete states
Factored "observations" as constraints



	atomic	factored	relational	first order
deterministic / full information	path search MCTS	IW planning constraint sat	PDDL planning	
non-deterministic / partial information	conformant, conditional planning	propositional logic	FOND planning	first-order logic
probabilistic	MDPs POMDPs	prob graphical models	probabilistic relational models	probabilistic logic

Inference about state based on certain observations

- State space is factored into a set of state variables
- Observations are constraints on (pieces of information about) the values of those state variables
- Our objective is to figure out one or more possible states that are consistent with the observations.

Inference doesn't increase our information about the underlying state—just processes it into a more useful form

Constraint-satisfaction problem: formal definition

- $\mathcal{X}$ is a set of variables $\{X_1, \ldots, X_n\}$
- $\mathcal{D}$ is a set of domains $\{D_1, \ldots, D_n\}$, where $D_i = \{x_1, \ldots, x_k\}$ is the set of possible values of X_i
- $\mathcal{C}$ is a set of constraints:
  - *scope* : a tuple of variables
  - *relation* : a relation specifying tuples of values that this tuple of variables can legally take on

Define:

- *assignment* : mapping from variables to values
- *partial assignment*: only provides values for some variables
- *consistent assignment* : partial assignment that doesn't violate any constraints
- *solution* : complete assignment that doesn't violate any constraints

CSP Objectives

Three possible objectives:

1. Solution: find a satisfying assignment or prove one does not exist.
2. All solutions: find all satisfying assignments.
3. Inference: conclude what values a variable must have.

CSP is NP-Complete: time exponential in domain size in worst case

Constraint-satisfaction problems: context

Why study CSP?

- A CSP formulation exposes structure in the problem that enables efficient inference
- There are many professional, efficient CSP solvers
- There are lots of important problems that can be formulated as CSP
- So, if you can formulate as a CSP, it can often be solved efficiently

Factory problem

Objects:

- Machines: sander, painter, dryer
- Parts: A, B, C
- Times: $1, \dots, 5$

Constraints:

- Each part must be sanded before painted before dried.
- The sander and painter can each operate on at most one part at a time.
- The sander can't operate at the same time the dryer is operating.

How can we formulate this as a CSP?

Hey robot, where are my keys?

- Rooms: bedroom, bathroom, office, kitchen
- Objects: keys, laptop, wallet, purse

Constraints:

- My keys are in the same place as my wallet.
- My wallet is not in my purse.
- My purse is next to my laptop.
- My laptop is in my office.
- I would never put my wallet in the bathroom.

Where should the robot look?

Variations on the theme

Theme: fixed number of variables with finite discrete domains

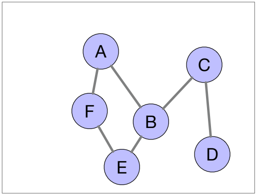
Variations:

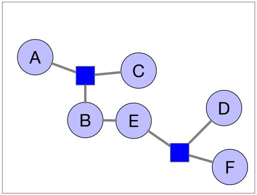
- Infinite discrete domains
- Continuous domains
  - With linear constraints == linear programming
    Polynomial time in number of variables!
  - Some other classes have good specialized solutions
    (e.g., quadratic programs)
- Getting all solutions
- Assigning costs and finding least cost solution
  == constrained optimization

Constraints

- **Unary** constraint only involves a single variable: use to reduce the domain of that variable
- **Binary** constraint involves two variables (domains can be any size). We will focus on binary constraints. Discussion in book and HW problem on reducing higher-order constraints to binary, and other ways of handling them.

Constraint (hyper)graph: useful to visualize constraint structure





If it does not contain any loops, then there's a cool, efficient message passing algorithm.

Stupidest possible algorithm

```
STUPID
    for each possible assignment $A$:
        if $A$ does not violate any $C \in \mathcal{C}$:
            return $A$
    return 'failed'
```

- How many A are there?
- Can we do better in the worst case?
- Can we do better in many cases?

Backtracking (= depth-first search)

a is an assignment, initially { }

```
BACKTRACK($a$)
    if COMPLETE($a$): return $a$
    $X$ = UNASSIGNED-VAR($a$)
    for $x \in$ DOMAIN-VALUES($X$):
        if CONSISTENT($a, \{X = x\}$):
            EXTEND($a, \{X = x\}$)
            $r$ = BACKTRACK($a$)
            if $r \neq$ 'failed': return $r$
            REMOVE($a, \{X = x\}$)
    return 'failed'
```

Is this better than the stupidest possible algorithm?

Variable and value ordering

Dynamically, during search:

- **variables**: UNASSIGNED-VAR chooses the variable with the fewest values in its domain
- **values**: DOMAIN-VALUES orders values earlier that rule out the fewest choices for variables it's connected to in the constraint graph

These will be especially useful in combination with some inference methods.

Next time: more inference methods

- Forward checking, backjumping
- Local search
- Message passing