

Lecture 4: Propositional Logic

September 14th, 2026



Gabriel Poesia

Assistant Professor, CSE

University of Michigan, Ann Arbor

`poesia@umich.edu`

`https://gpoesia.com`

- 0. Administrivia
- 1. Knowledge-based agents
- 2. A small Wumpus world
- 3. Propositional logic
- 4. Models and proofs
- 5. Resolution and refutations
- 6. DPLL

Homework 1 check-in

Project proposal will be out this Wednesday (16th), due 2 weeks after

Should have: group (2-3 members), problem setup, evaluation plan. Short, 1 page proposal + references.

Use Ed to find others with common interests (many already doing this!)

Scope: propose and explore anything new related to the course!

Can be an extension of an existing paper, or new idea entirely

Open-ended, but you need a clear problem and experimental evaluation!

This *can change* later as you work on the idea. But you need a well-scoped starting point.

Use our office hours and Ed for feedback if you'd like.

Also very useful: your favorite AI's Deep Research mode. Find papers, background, brainstorm, etc.

Read the primary sources! (e.g., the actual papers, rather than just AI explanations or summaries)

Be critical and careful with sycophancy ("*Yes, that's a brilliant idea!*")

Where we left off

Search problems: states, actions, transitions, and a goal test.

Last lecture, we assumed **factored states**: variables, domains, and constraints.

After assigning a variable, we could **infer consequences** before making more choices.

CSP was given to us up front (e.g., Sudoku grid and all constraints).

But an agent acting in the world might observe state incrementally.

It will need to keep track of *what it knows*, and update this knowledge as it acts and sees more.

How can we represent what we know, and use it to decide what to do next?

Knowledge-based agents

We will give the agent a language to represent and operate on knowledge.

The main idea is that we'll store a **knowledge base** by accumulating pieces of knowledge.

DEFINITION (KNOWLEDGE BASE)

A **knowledge base** (KB) is a collection of sentences describing what the agent knows about the world.

Some sentences might be known in advance (e.g., prior knowledge about the world).

Others might come from observations and actions dynamically.

Two basic operations we want:

TELL

Add a new sentence to the KB

ASK

Ask if some sentence follows from the KB

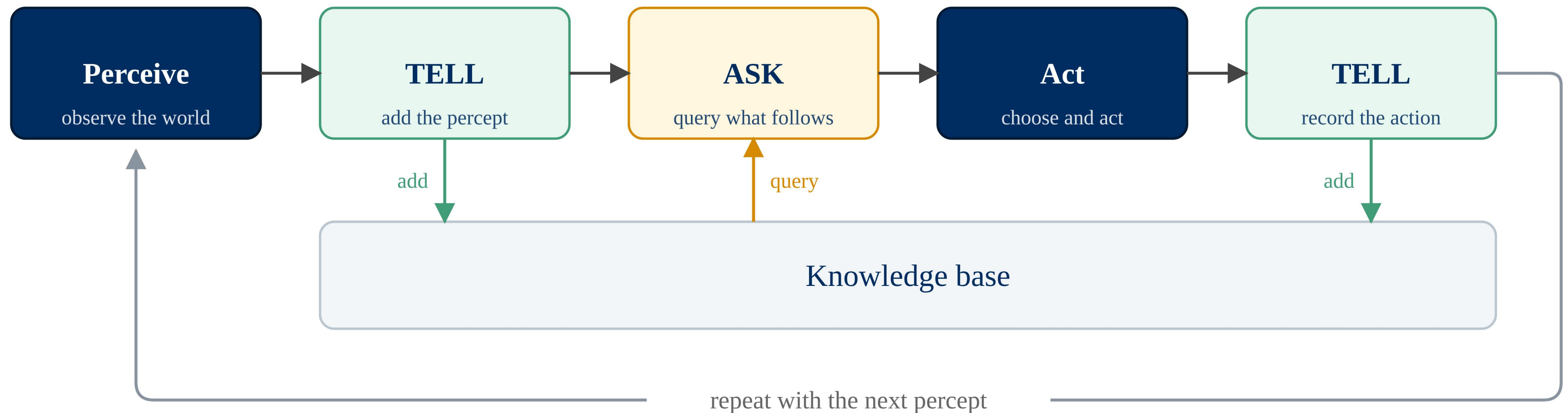
ASK should do more than look up sentences we already added!

The KB might leave many world states possible.

We want conclusions that hold in *all of them*

Knowledge-based agents

A knowledge-based agent repeatedly observes the world, updates its KB, and chooses an action:



Example: ASK where is it safe to go, choose one place using the agent's goal, and move.
Record the action in the KB, then repeat with the next observation.

EXAMPLE: WUMPUS WORLD

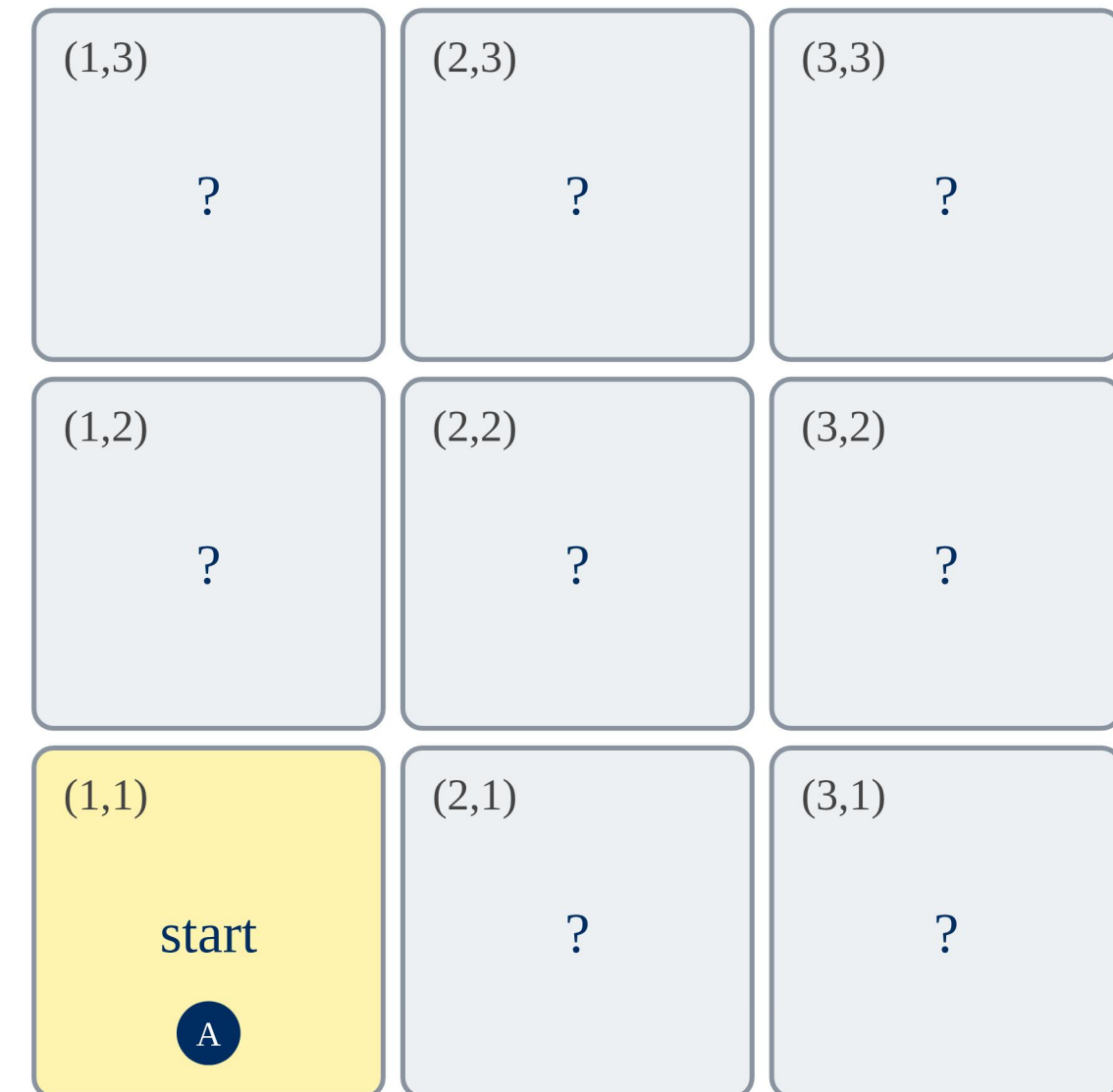
What can the agent see?

Imagine we're in a grid world. Goal: find cell with gold.

We know the following rules:

- We can only move to a horizontally or vertically adjacent cell
- Cells can be empty, have a pit, or have gold
- If we enter a pit, we die (so no gold!)
- We can only sense gold if we're on it
- We can sense a **breeze** when (and only when) at least one adjacent cell has a pit

Start cell is safe, pits do not move, and observations are reliable.



No breeze at the start

Suppose we are at (1, 1) and observe **no breeze**.

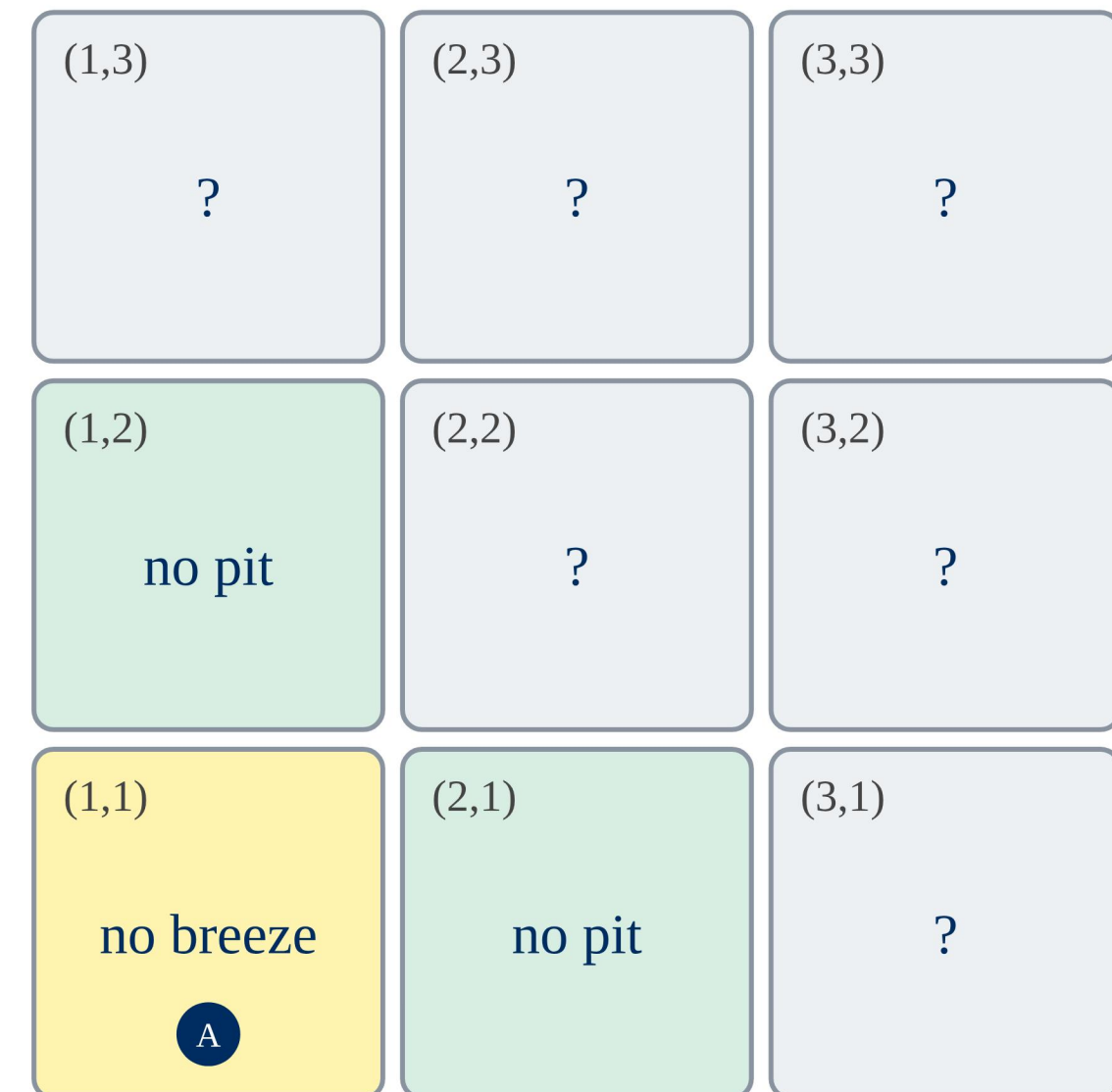
Which cells can we safely enter?

(1, 2) and (2, 1).

Why? If either cell had a pit, there would be a breeze at (1, 1).

So, an observation in one cell lets us infer something about two other cells.

This is what we want ASK to do.



What does a breeze tell us?

Say we move to (2, 1). Now we observe a breeze.

Neighbors: (1, 1), (2, 2), and (3, 1).

We know (1, 1) is safe, so there must be a pit in (2, 2) **or** (3, 1) (possibly both).

Can we conclude (2, 2) has a pit? No, we don't know!

Can we conclude (2, 2) does not have a pit? Also no!

What if: we go to (1, 2) and observe **no breeze**?

Then (2, 2) cannot have a pit. But some neighbor of (2, 1) does. So must be (3, 1)

We need a language that can express this kind of partial knowledge and inferences.



PROPOSITIONAL LOGIC

What counts as a proposition?

DEFINITION (PROPOSITION)

A **proposition** is a statement that can be true or false.

Which of these are propositions?

Statement	Proposition?
There is a pit at (2, 2)	Yes (even if we don't know whether it is true)
$2 + 2 = 5$	Yes, a false one
Move to the next cell!	No, this is a command
Is there a pit here?	No, this is a question
All cells have gold in them	Yes

We'll use symbols for propositions with defined meaning (e.g., $P_{2,2}$ for “pit at (2, 2)”).

Propositional Logic Syntax

Let's build up the *syntax* for our language. What are all the valid sentences?

Start with **atomic propositions** (atoms): $P, Q, R, P_{2,2}, \dots$

We can combine them using logical connectives:

DEFINITION (SENTENCE)

Every atom is a sentence. If α and β are sentences, so are:

$$\neg\alpha, \quad (\alpha \wedge \beta), \quad (\alpha \vee \beta), \quad (\alpha \rightarrow \beta), \quad (\alpha \leftrightarrow \beta).$$

Read: NOT, AND, OR, implies, if and only if.

Example: $B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$.

Sentences can be nested: $\neg(P \wedge (Q \vee R))$.

Parentheses matter! $(P \vee Q) \wedge R$ and $P \vee (Q \wedge R)$ say different things.

OR is inclusive: either or both.

Writing the knowledge base

We now have a language to write down our Wumpus knowledge base.

$P_{x,y}$: pit at (x, y) . $B_{x,y}$: breeze at (x, y) .

What we know	Sentence
Start is safe	$\neg P_{1,1}$
Breeze rule at $(1, 1)$	$B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$
No breeze at $(1, 1)$	$\neg B_{1,1}$
Breeze rule at $(2, 1)$	$B_{2,1} \leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
Breeze at $(2, 1)$	$B_{2,1}$

For the next few examples, write B, P, Q for $B_{1,1}, P_{1,2}, P_{2,1}$.

Our knowledge base (KB) will be a set of sentences like these

Initially, we know (among other things):

$$KB = \{B \leftrightarrow (P \vee Q), \neg B\}$$

In fact, we start knowing all the breeze *rules*, but not where we get breezes

What does it mean for one of these sentences to be true?

Truth will be defined once we have an *interpretation* for the atomic propositions.

DEFINITION (INTERPRETATION)

An **interpretation** m assigns each atomic proposition a truth value in $\{\perp, \top\}$ (false or true).

Think of m as one possible world. For example, $m(B) = \perp$, $m(P) = \top$, $m(Q) = \perp$.

Once the atoms are assigned, the connectives determine the truth of every sentence.

DEFINITION (MODEL)

Write $m \models \alpha$ when α is true in m . We say m is a **model of** α .

A model of a KB makes every sentence in KB true.

An interpretation might disagree with what we know. In that case, it is not a model of our KB.

Propositional semantics

Given a **particular interpretation** m and some arbitrary sentence P , how do we tell $m \models P$?

For atoms: $m \models P$ iff $m(P) = \top$ (base case).

For negation: $m \models \neg\alpha$ iff $m \not\models \alpha$.

For the other connectives, just need to look at the truth values of the two parts:

α	β	$\alpha \wedge \beta$	$\alpha \vee \beta$	$\alpha \rightarrow \beta$	$\alpha \leftrightarrow \beta$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$
$\top$	$\perp$	$\perp$	$\top$	$\perp$	$\perp$
$\perp$	$\top$	$\perp$	$\top$	$\top$	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$	$\top$	$\top$

α and β can themselves be compound sentences, so this defines truth **recursively**.

What does implication say?

$\alpha \rightarrow \beta$ is false in exactly one case: α is true and β is false.

Example: “If there is a pit at (1, 2), there is a breeze at (1, 1).”

What if there is no pit at (1, 2)? The rule says nothing about the breeze.

There could still be a breeze caused by a pit in another neighboring cell.

CAREFUL

$\alpha \rightarrow \beta$ does not imply $\beta \rightarrow \alpha$. It also does not, by itself, say that α causes β .

Example: think about "you are enrolled in CSE592" $\rightarrow$ "you are a U-M student".

Our breeze rule uses $\leftrightarrow$, which requires **both directions**:

$$B \rightarrow (P \vee Q) \quad \text{and} \quad (P \vee Q) \rightarrow B$$

Evaluating a sentence

Let's evaluate $B \leftrightarrow (P \vee Q)$ in two interpretations:

$$B = \perp, P = \top, Q = \perp$$

$$P \vee Q = \top$$

$$\perp \leftrightarrow \top = \perp$$

Violates the breeze rule

$$B = \perp, P = \perp, Q = \perp$$

$$P \vee Q = \perp$$

$$\perp \leftrightarrow \perp = \top$$

Satisfies the rule (and $\neg B$)

Remember: an interpretation is a "possible world"

All this is is: the first interpretation disagrees with what we know (our KB)

Given an interpretation, evaluating a sentence is straightforward.

But ASK needs to say what follows in **every** interpretation consistent with our knowledge.

This idea of "What is true in every possible world", is captured by entailment:

DEFINITION (ENTAILMENT)

KB **entails** β , written $KB \models \beta$, if every model of KB is also a model of β :

$$\forall m : \quad \text{if } m \models KB \text{ then } m \models \beta.$$

If everything in the KB holds, the conclusion must hold too.

Example: $\{B \leftrightarrow (P \vee Q), \neg B\} \models \neg P$.

Note that $\models$ and $\rightarrow$ have different roles:

- $\alpha \rightarrow \beta$: a sentence **in the logic**, true or false in a particular world
- $KB \models \beta$: a statement **about the logic** (meta-logic), comparing all models of the KB

How do we show that an entailment does *not* hold?

Find one **countermodel**: a world where the KB is true and the conclusion is false.

Entailment as a set of worlds

Let $M(\alpha) = \{m : m \models \alpha\}$ be the models of α .

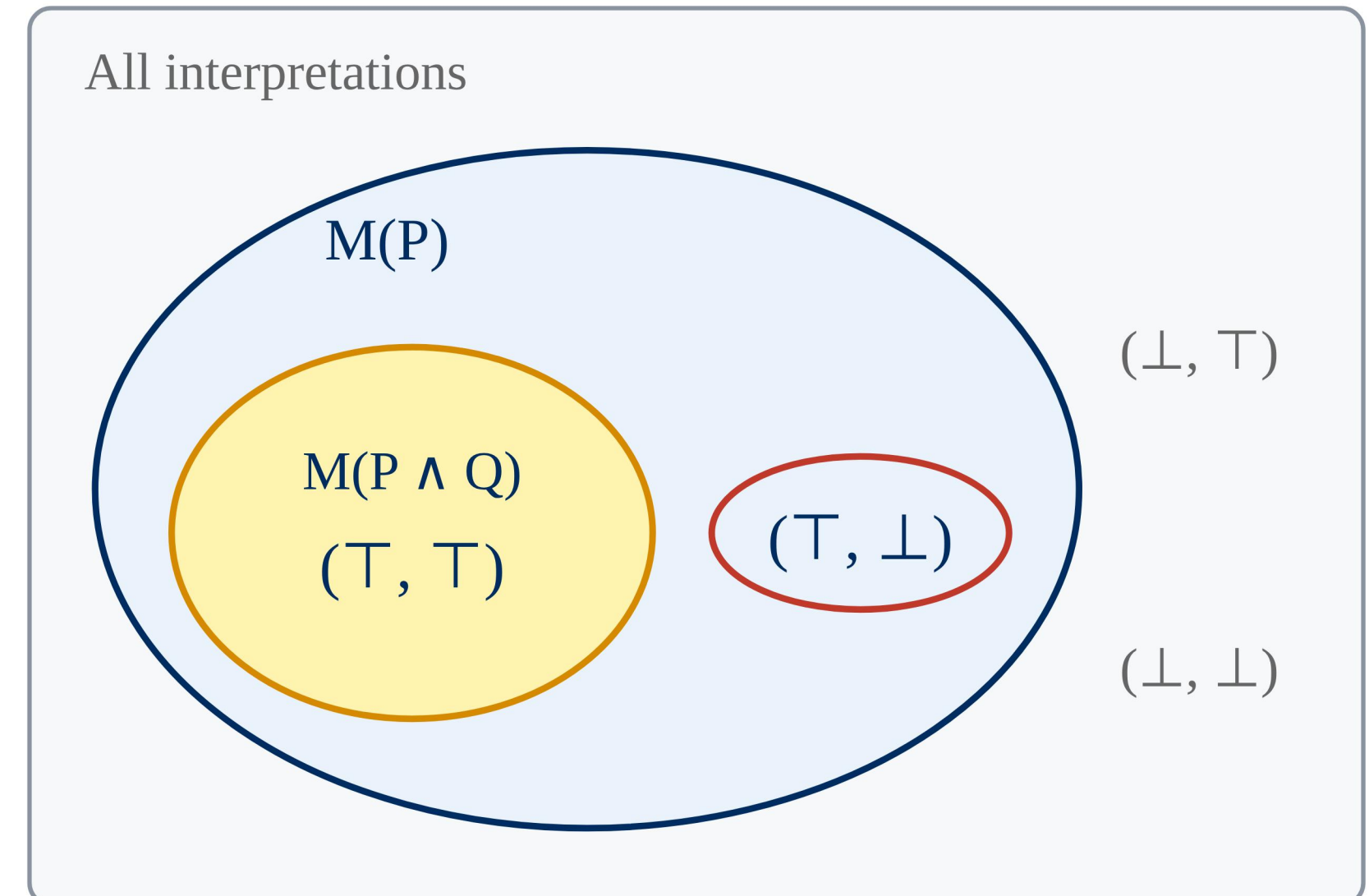
$$\alpha \models \beta \iff M(\alpha) \subseteq M(\beta)$$

Example: $P \wedge Q \models P$.

Does $P \models P \wedge Q$?

No: $P = \top, Q = \perp$ is a countermodel.

Adding information to a KB can only
remove possible worlds



Worlds written as truth values of (P, Q)

What follows?

$$KB = \{P \vee Q, \neg P\}$$

Which sentence is entailed? (R is another atom.)

A. Q

B. $\neg Q$

C. P

D. $Q \wedge R$

Every model of the KB has $P = \perp$ and $Q = \top$ (so B and C are false).

But R can be false. This gives a countermodel to D.

What if we do not know?

Suppose our KB contains only $P \vee Q$.

Does P follow? No. Does $\neg P$ follow? Also no!

$$P = \top, Q = \perp$$

Satisfies the KB, but falsifies $\neg P$

$$P = \perp, Q = \top$$

Satisfies the KB, but falsifies P

$$KB \not\models P \text{ and } KB \not\models \neg P$$

Each world gives P an ordinary true/false value. We just don't know which world we are in.

Not knowing whether P is true is different from knowing that P is false

MODELS AND PROOFS

Model checking by enumeration

$KB = \{\neg B, B \leftrightarrow (P \vee Q)\}$. Query: $\neg P$.

Can we give an algorithm?

Simplest idea: enumerate all models. We have 3 propositions. How many worlds? $2^3 = 8$

The four worlds with $B = \top$ already violate $\neg B$.

For $B = \perp$, we have four possibilities left:

P	Q	$B \leftrightarrow (P \vee Q)$	KB true?	$\neg P$
$\perp$	$\perp$	$\top$	Yes	$\top$
$\perp$	$\top$	$\perp$	No	$\top$
$\top$	$\perp$	$\perp$	No	$\perp$
$\top$	$\top$	$\perp$	No	$\perp$

$\neg P$ is true in every model of the KB. Thus, $KB \models \neg P$.

Rows where the KB is false are irrelevant to the query.

The cost of enumerating worlds

With n atoms in the KB and query, there are 2^n possible assignments.
For each one, check: if the KB holds, does the query hold too?

- Find a countermodel: return “not entailed”
- Finish without finding one: return “entailed”

This is a finite procedure, so propositional entailment is **decidable**.
But it might do a lot of unnecessary work!

EXAMPLE: UNRELATED INFORMATION

From P and $P \rightarrow Q$, we can infer Q . Suppose we also have 30 unrelated rules $R_i \vee S_i$: a naive truth table now enumerates 60 additional atoms.

Can we derive the conclusion using just the relevant sentences?

Equivalence, validity, satisfiability

Some terminology for how sentences behave across interpretations:

DEFINITION (LOGICAL EQUIVALENCE)

$\alpha \equiv \beta$ if they have exactly the same models. Equivalently, $\alpha \models \beta$ and $\beta \models \alpha$.

Example: $(P \rightarrow Q) \equiv (\neg P \vee Q)$.

A sentence is...	If it is true in...	Example
Valid (a tautology)	Every world	$P \vee \neg P$
Satisfiable	At least one world	$P \wedge Q$
Unsatisfiable	No world	$P \wedge \neg P$

Every valid sentence is satisfiable. Is the converse true? No (e.g., $P \wedge Q$).

Turning entailment into validity

Write K for the conjunction of the sentences in our finite KB.

We can turn an entailment question into a validity question:

THEOREM

$$KB \models \beta \iff (K \rightarrow \beta) \text{ is valid.}$$

PROOF

When is $K \rightarrow \beta$ false? Exactly when K is true and β is false.

This is precisely a countermodel to $KB \models \beta$.

So, $K \rightarrow \beta$ is true in every world iff there are no countermodels to the entailment.



For now this just relates two semantic questions (questions about models).

We can avoid thinking about "every interpretation" by reasoning about **proofs** instead.

Instead of enumerating worlds, we can perform inferences using sentences we already know.

DEFINITION (INFERENCE RULE)

An **inference rule** gives a conclusion from some premises. We write the premises above a line, and the conclusion below it.

Modus ponens

$$\frac{\alpha \quad \alpha \rightarrow \beta}{\beta}$$

AND elimination

$$\frac{\alpha \wedge \beta}{\alpha} \quad \frac{\alpha \wedge \beta}{\beta}$$

Another example: **double-negation elimination**, $\frac{\neg\neg\alpha}{\alpha}$.

α and β can be arbitrary sentences (not just atoms).

What is a proof?

Suppose $KB = \{P, P \rightarrow (Q \wedge R)\}$. How can we show that Q holds?

A (formal) proof starts from axioms (assumptions) and derives one conclusion from an inference rule in each line.

Line	Sentence	Reason
1	P	Given in KB
2	$P \rightarrow (Q \wedge R)$	Given in KB
3	$Q \wedge R$	Modus ponens on 1 and 2
4	Q	AND elimination on 3

DEFINITION (PROOF)

A **proof** is a sequence of sentences, each of which is a premise or follows from earlier sentences by an inference rule.

Can also be represented as a proof tree (duplicating possible shared subproofs)

Soundness and completeness

A set of inference rules I gives us a *proof system*, that allows us to make deductions

Write $KB \vdash_I \beta$ when proof system I can derive β from the KB.

What should we require from I ?

DEFINITION (SOUNDNESS)

A proof system is sound if everything we can derive is true when the premises are true:

$$KB \vdash_I \beta \implies KB \models \beta.$$

DEFINITION (COMPLETENESS)

A proof system is complete if we can derive any true sentence in it:

$$KB \models \beta \implies KB \vdash_I \beta.$$

In both cases we assume the premises hold. Wrong hypotheses can still lead to wrong conclusions.

Ideally we'd have both! This give two equally powerful ways to reason from the KB

We also need a search algorithm to search for proofs in $\vdash_I$.

Checking soundness

Given an inference rule, how do we know it is sound?

We just need to check the models where the premises are true.

THEOREM

Double-negation elimination is sound: $\neg\neg\alpha \models \alpha$.

PROOF

Suppose m satisfies $\neg\neg\alpha$.

By the meaning of $\neg$, m does not satisfy $\neg\alpha$.

Again by the meaning of $\neg$, m satisfies α .

This holds for every model of the premise, so the rule is sound. ■

What about $\frac{\alpha \rightarrow \beta}{\alpha} \beta$? **Not sound!**

Set $\alpha = \perp$, $\beta = \top$: both premises hold, but the conclusion is false.

If every rule in $\vdash_I$ is sound, the proof system is sound. (by induction over steps of a proof)

Are these rules enough?

So given rules in $\vdash_I$, we can tell whether they are sound.

How do we know we have enough rules?



RECALL

$$\{P \vee Q, \neg P\} \models Q$$

Can our three rules derive this conclusion?

Modus ponens, AND elimination, and double-negation elimination cannot even make a first step here.

So, these rules are sound, but together they are **incomplete**.

We could add more rules. How would we know when we have enough?

Turns out a single rule, **resolution**, is enough to find a contradiction whenever propositional clauses are unsatisfiable.

First, we'll put our sentences in a common form.

RESOLUTION AND REFUTATIONS

Conjunctive normal form

Resolution operates on sentences in a standard form:

DEFINITION (CONJUNCTIVE NORMAL FORM)

A **literal** is an atom or its negation.

A **clause** is a disjunction of literals.

A sentence is in **conjunctive normal form** (CNF) if it is a conjunction of clauses.

Example: $(P \vee \neg Q) \wedge (Q \vee R) \wedge \neg R$.

Literals: $P, \neg Q, \dots$ Clauses: $P \vee \neg Q, Q \vee R, \neg R$.

We can store this as a set of sets: $\{\{P, \neg Q\}, \{Q, R\}, \{\neg R\}\}$.

- One literal: a **unit clause**
- Zero literals: the **empty clause**, written $\square$ (always false)

⚠ CAREFUL

An empty *clause* is false. An empty *set of clauses* is true (no requirements left).

Getting to CNF



Every propositional sentence has an equivalent CNF.
We get there by applying logical equivalences. For example:

Sentence	Rewrite	General rule
$P \rightarrow (Q \wedge R)$	Original sentence	
$\neg P \vee (Q \wedge R)$	Eliminate implication	$\alpha \rightarrow \beta \equiv \neg \alpha \vee \beta$
$(\neg P \vee Q) \wedge (\neg P \vee R)$	Distribute OR over AND	$\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

Other useful rewrites: $\neg(A \wedge B) \equiv \neg A \vee \neg B$, $\neg\neg A \equiv A$, and some more (book chapter 7 has all relevant ones).
These can take any propositional sentence into one in CNF

 RECALL

Like the CSP $\rightarrow$ binary CSP encoding from last lecture, we can reason about a special case without losing completeness

 CAREFUL

An equivalent CNF can be exponentially larger than the original sentence.

Fresh variables can avoid this blowup in SAT encodings, but preserve satisfiability rather than equivalence over all variables.

Proof by refutation

To prove β , we can show that assuming $\neg\beta$ leads to a contradiction.

THEOREM

$$KB \models \beta \iff K \wedge \neg\beta \text{ is unsatisfiable.}$$

Why? A model of $K \wedge \neg\beta$ would be a countermodel: KB true, query false.

If no such world exists, every model of the KB satisfies β .

EXAMPLE: PROVING A CELL IS SAFE

To prove $\neg P$ from $\{\neg B, B \leftrightarrow (P \vee Q)\}$, add P and show that the sentences cannot all hold.

DEFINITION (REFUTATION)

A **refutation** is a proof of contradiction from a set of premises.

The resolution rule

Suppose we know $P \vee Q$ and $\neg P \vee R$. What can we infer?

$$Q \vee R$$

If P is false, the first clause requires Q . If P is true, the second requires R .

DEFINITION (RESOLUTION)

Resolution in propositional logic is the inference rule:

$$\frac{C \vee L \quad D \vee \neg L}{C \vee D}$$

L is the **pivot literal**; C, D are the remaining disjunctions (possibly empty).

Remove **one complementary pair**, keep the remaining literals, and remove duplicates.

With unit clauses: $\frac{P \quad \neg P}{\square}$.

THEOREM

Resolution is sound: $\{C \vee L, D \vee \neg L\} \models C \vee D$.

PROOF

Suppose m satisfies both premises. There are two cases:

If L is true, $\neg L$ is false, so D must hold.

If L is false, C must hold.

In either case, m satisfies $C \vee D$. This holds for every model of the premises.



Can we remove two complementary pairs at once?

Consider $P \vee Q$ and $\neg P \vee \neg Q$. Both hold when $P = \top, Q = \perp$.

Removing both pairs would derive $\square$ from satisfiable premises!

A Wumpus refutation

Query: $\neg P$. KB: $\neg B, B \leftrightarrow (P \vee Q)$.

Converting the breeze rule to CNF gives:

$$\begin{aligned} &(\neg B \vee P \vee Q) \\ &\wedge (B \vee \neg P) \wedge (B \vee \neg Q) \end{aligned}$$

Add the negated query: P .

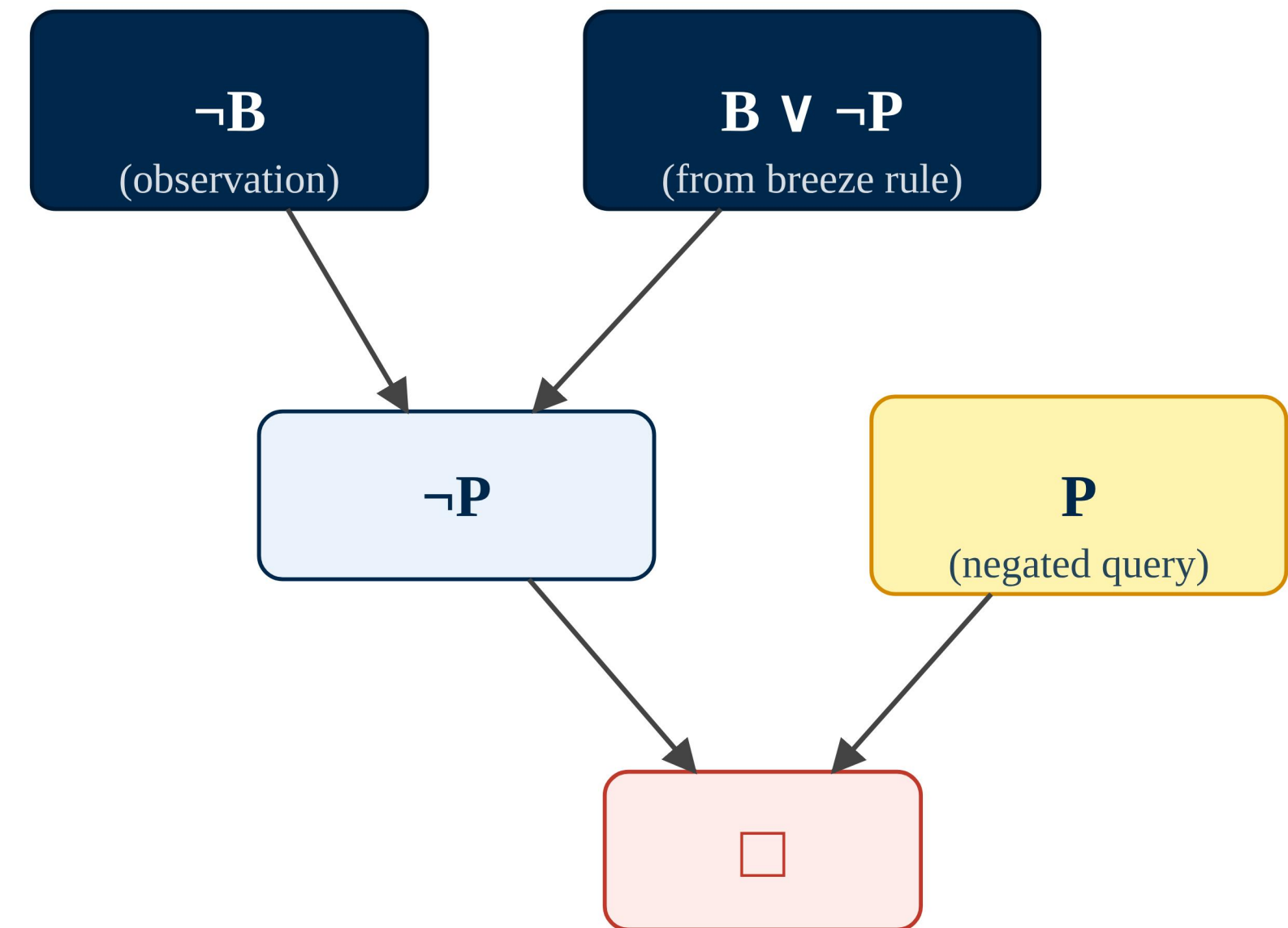
Resolve $\neg B$ with $B \vee \neg P$ on B .

Resolve $\neg P$ with P on P .

Thus, $KB \models \neg P$: no pit at (1, 2).

The other clauses are available, but this proof doesn't need them.

Each clause we derive from resolution is called a **resolvent**



Searching for a refutation

Start with the KB and negated query. Repeatedly add all clauses we can derive:

```
1  function ENTAILS-BY-RESOLUTION(KB, beta):
2      clauses = CNF(KB AND (NOT beta))  # set of clauses
3      if empty clause in clauses: return true
4      while True:
5          new = all resolvents of pairs in clauses
6          if empty clause in new: return true
7          if new is a subset of clauses: return false
8          clauses = clauses union new
```

For each pair, consider every possible single-pivot resolution.

Treat clauses as sets; discard tautologies and repeated clauses.

If no new clause can be derived, we have reached a **fixed point**.

Resolution: guarantees and cost

THEOREM (REFUTATION COMPLETENESS)

Resolution is **refutation-complete**: Every unsatisfiable set of propositional clauses has a resolution derivation of $\square$.

⚠ CAREFUL

Not the same as completeness: even if $KB \models P$ holds, resolution from KB might not derive P

What is true: if $KB \models P$ holds, resolution from $KB \wedge \neg P$ will derive $\square$

So, together with soundness, this gives us an algorithm for entailment:

- Derive $\square$: the clauses have no model, so the query follows
- Reach a fixed point without $\square$: the clauses have a model, so the query does not follow

Does this terminate? With n atoms, there are at most 3^n non-tautological clauses.

Each atom can appear positively, negatively, or not at all.

So it terminates, but may store exponentially many clauses.

Heuristics can help, but can lose completeness if some pairs are never considered.

Back to the agent

Return through (1, 1) and enter (1, 2), which we know is safe.

Suppose we observe no breeze here either.

$$B_{1,2} \leftrightarrow (P_{1,1} \vee P_{1,3} \vee P_{2,2})$$

What follows? $\neg P_{1,3}$ and $\neg P_{2,2}$.

Earlier, the breeze at (2, 1) gave us $P_{2,2} \vee P_{3,1}$.

Now we know $\neg P_{2,2}$, so $P_{3,1}$ **must hold**.

We can safely enter (2, 2), and should avoid (3, 1).

(1,3) no pit	(2,3) ?	(3,3) ?
(1,2) no breeze A	(2,2) no pit	(3,2) ?
(1,1) no pit	(2,1) breeze	(3,1) PIT

- **Propositional logic:** atoms + connectives to represent knowledge
- **Semantics:** interpretations assign truth values; entailment asks what is true in every model of the KB
- **Entailment via truth tables:** enumerating models can decide propositional entailment (but exponential)
- **Proofs:** chain of applications of inference rules; sound rules preserve truth; resolution is sound and refutation-complete for CNF

Next lecture: a practical strategy to decide entailment via satisfiability (SAT): the DPLL algorithm (live-coding!)

Plus: a much more expressive language for knowledge: first-order logic.

Last lecture of the first module!

Bonus reading for today:

Karem Sakallah honored for 30 years of foundational SAT-solving research

September 2, 2026

Sakallah and his former doctoral student João Marques-Silva were recognized by the SAT Association for developing GRASP, which introduced an approach that transformed automated reasoning.

Karem Sakallah, professor of computer science and engineering (CSE) at the University of Michigan, has received an inaugural award from the SAT Association recognizing “Extraordinary Achievements for 30 Years of Clause Learning.”

Sakallah received the award alongside João Marques-Silva, his former PhD student at U-M and now a professor at the University of Toulouse and research director at France’s National Centre for Scientific Research. The award was presented at the [2026 International Conference on Theory and Applications of Satisfiability Testing \(SAT\)](#), held July 20–23 in Lisbon, Portugal.

The SAT conference is the premier annual meeting for researchers studying the Boolean satisfiability problem, commonly called SAT. This foundational problem asks whether values can be assigned to a collection of variables in a way that satisfies a given set of logical constraints. Solving it has important applications in areas spanning hardware and software verification, planning, test generation, and automated theorem proving.



Prof. Karem Sakallah

Clause Learning in SAT solvers, developed here at U-M, [was a major breakthrough in SAT!](#)