ARTIFICIAL INTELLIGENCE

Logical Agents

NOTICE

Presentation slides will be available for download along with the recording of this review session.

You are free to take pictures regardless if you'd like.

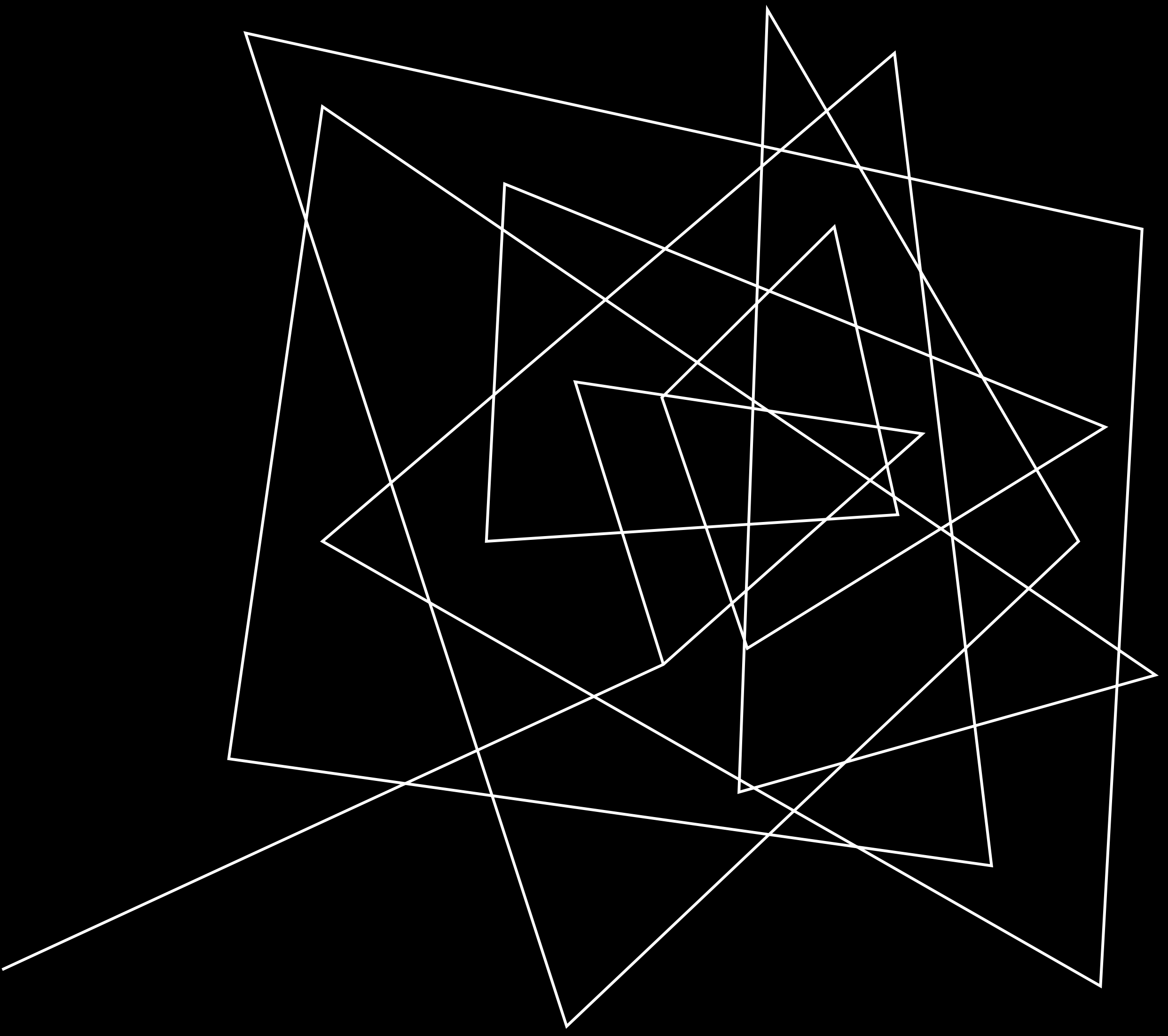
DISCLAIMER

I have tried to pull all information from the slides, the textbook, and other authorized resources, but I cannot guarantee the veracity of any information in the slides hereafter.

See presentation notes for sources (chapter pages are for 4^{th} ed.)

OUTLINE

- Knowledge-based agents
- Logic
- Propositional logic
- Entailment concepts & inference rules
- Advanced inference algorithms



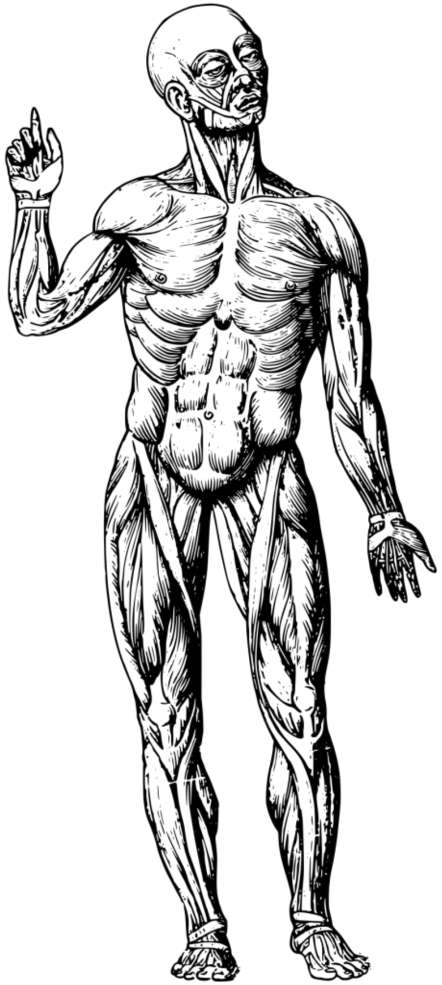
KB AGENTS

Unrelated to KGB agents, I promise

KNOWLEDGE-BASED AGENTS

- "**Knowledge-based agents (KB agents)** use a process of **reasoning** over an internal **representation** of knowledge to decide what actions to take."
- But what real thing can we use to demonstrate this oh so complex idea?

THIS GOOFY LAD



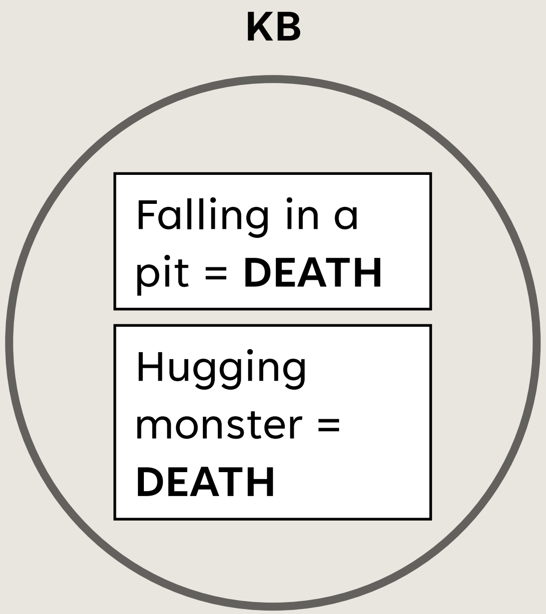
- Few things are more relatable than our own mortal existence!
- Also, humans and KB agents share a depressing number of similarities.

THE COMPONENTS OF THE KB AGENT

- Four categories of components:
  - Knowledge Base (KB) – Memory
  - Sensors – Eyes, ears, nose, skin, etc.
  - Actuators – Muscles, tear glands, etc.
  - Logic – Cognitive functions

THE KNOWLEDGE BASE

- The **knowledge base (KB)** is a set of **sentences**.
- A sentence states an **assertion** or **premise** about the agent's world.
- These sentences are expressed in a **formal** language, called a **knowledge representation language (KRL)**.



ON SENTENCES

- **Axioms** are sentences that are not derived.
- Sometimes, the agent is given **background knowledge**.
  - Axioms (typically) ∈ Background knowledge
    - Depends on nature of agent
- Sentence operations:
  - You can $TELL$ an agent a sentence (insert it into the KB)
  - You can ASK an agent a query (derive action from KB)
- Both may involve **inference**, deriving new sentences from old.
  - Inference should “follow” from KB (i.e. not make things up)

HUMAN SENTENCES OPERATIONS

- Human axioms and background knowledge are questionable, but it’s easy to $TELL$ something to or ASK something of a human!
- $TELL$ – human perception, experiences, trauma
- ASK – reaction to stimuli, predetermined action
  - Human ASK is not 100% rational
  - KB ASK may be misguided by sensors
- Human beings are largely **declarative**.

DECLARATIVE VS. PROCEDURAL

- **Declarative approach** – Building an agent by $TELL$ing it everything it needs to know.
- **Procedural approach** – Building an agent by encoding behavior as program code.
- Successful agents often combine **both** elements in their design.
- Declarative knowledge can be reduced into more efficient procedural code.

Declarative

- To face east from north, turn right by 90 degrees
- To face east from west, turn right by 180 degrees
- To face east from south, turn left by 90 degrees

Procedural

```
function faceEast() {
    if (facingNorth()) {
        turnRightBy90();
    } else if (facingEast()) {
…
```

PERSPECTIVES ON KNOWLEDGE

- **Knowledge level** – What the agent knows, regardless of implementation.
  - Ex. Examinations, quizzes
- **Implementation level** – The data structures representing the KB and the algorithms that manipulate them.
  - Ex. Types of memory, psychology
- Parallel to **design and implementation** in software engineering.

ADDING TO THE PILE

- You $TELL$ the KB agent information from its **sensors**.
- Sensors intake info from the KB's world.
- The KB receives this information expressed through its KRL.
- Note that sensors can be **deceived**!
  - Like humans! We are kept from reality as-is by conditions, biases, etc.

POP QUIZ!

What is a KRL?

CHANGE THE WORLD
(MY FINAL MESSAGE, GOODBYE)

- A KB agent takes actions in the world through **actuators** (in response to ASK).
- Actions generally incur the passage of time.
- Actions may be associated with prerequisites or costs.
  - Ex. Writing takes pencil, paper, energy, etc.
- Actions are evaluated with a **performance measure**.
  - Ex. Punishment/reward system

A BRIEF ASIDE ON LOGIC

- Saving **logic** for further elaboration.
- For now, know that logic decides how a KB agent responds to an ASK based on its KB.
- Actions are executed with the agent's actuators.
- When an agent draws a conclusion from correct information, the conclusion is guaranteed to be correct.

A BASIC KB AGENT

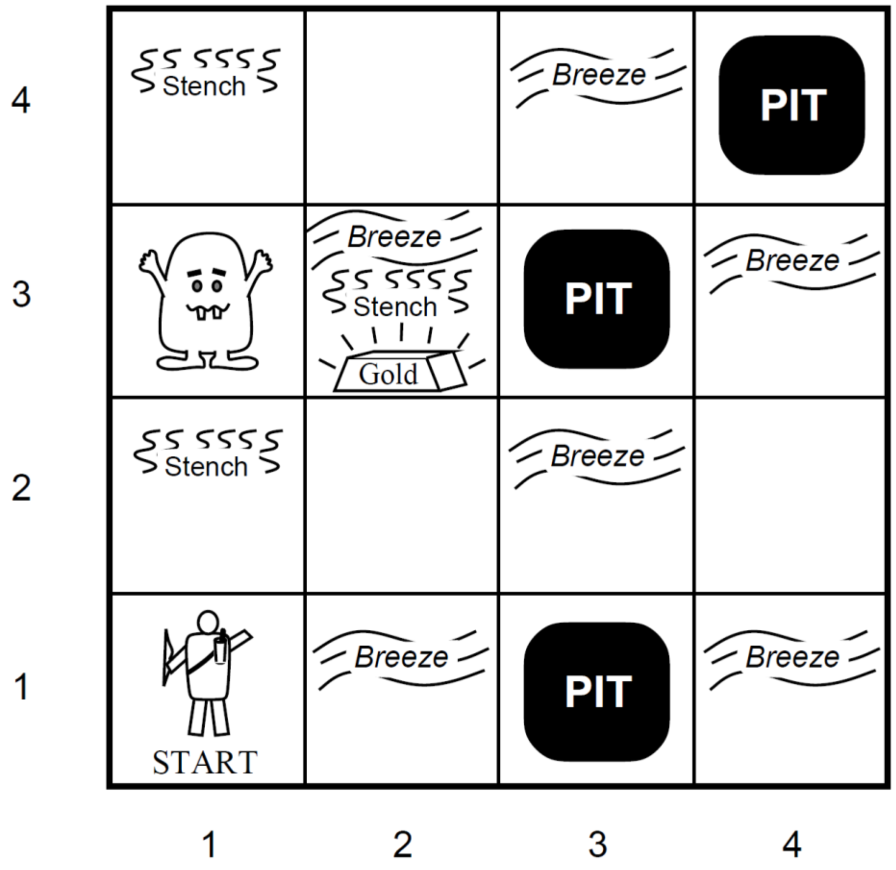
function KB-AGENT(*percept*) **returns** an *action*
    static: *KB*, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
    action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*))
    TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
    $t \leftarrow t + 1$
    return *action*

WUMPUS WORLD (LIKE THE ONE FROM DISCORD?)

- Performance Measure
  - Gold = +1,000
  - Step = -1
  - Arrow Use = -10
  - Death = -1,000
- Actuators
  - Left Turn
  - Right Turn
  - Forward
  - Grab
  - Release
  - Shoot
- Sensors
  - Breeze
  - Glitter
  - Smell

- Environment
  - Squares adjacent to Wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter **iff*** gold is in the same square
  - Shooting kills Wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
- ***iff** = if and only if
  - Will be used continually throughout this chapter!



CHARACTERISTICS OF WUMPUS WORLD

	Answer	Explanation
Observable?	No	Only **local** perception. The agent can only see within one space.
Deterministic?	Yes	Outcomes exactly specified. No random chance or unpredictability involved.
Episodic?	No	Sequential at the level of actions. Each perception and action affects the next.
Static?	Yes	Wumpus and Pits (environment) do not move.
Discrete?	Yes	There are a defined number of spaces, and a defined method of navigating them.
Single-agent?	Yes	Wumpus is essentially a natural feature. No other agents take turns or actions.

Questions on Wumpus World?

LOGIC

Thinking about thinking

INTRODUCTION TO LOGIC

- **Logics** are formal languages for representing information, such that conclusions can be drawn.
- **Syntax** defines how sentences in the language are expressed.
- **Semantics** define the "meaning" of sentences.
  - Semantics also define the **truth** of each sentence with respect to each **possible world**.
    - Every sentence must be either true or false (except in fuzzy logic)

ARITHMETIC AS A LOGIC LANGUAGE (FLUENCY: QUESTIONABLE)

- $x + y = 4$ is a sentence; $x4y+=$ is not.
- $x + 2 \geq y$ is true **iff** the number $x + 2$ is no less than the number y
- $x + y = 4$ is true in a world where x is 2 and y is 2, but false in a world where x is 1 and y is 1.

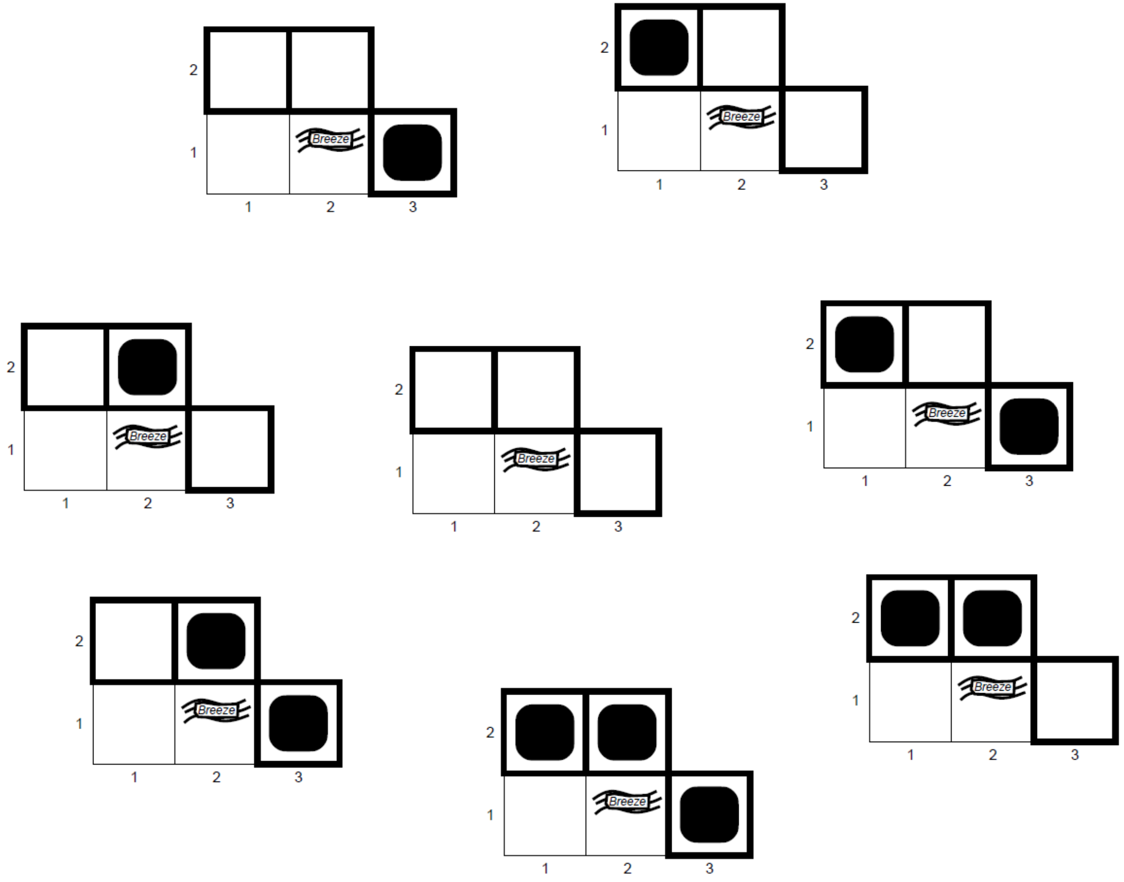
MODELS & SATISFACTION

- A possible world may or may not be real and/or accessible by the agent.
  - Possible world: assign x and y, true if $x + y = 4$
  - Model: Constrained to nonnegative integer values for x and y.
- A **model** is a mathematical abstraction where every relevant sentence has a fixed truth value.
- If a sentence α is true in model m, we say that m **satisfies** α or sometimes m **is a model of** α.
- $M(\alpha)$ means the set of all models of α.

ENTAILMENT

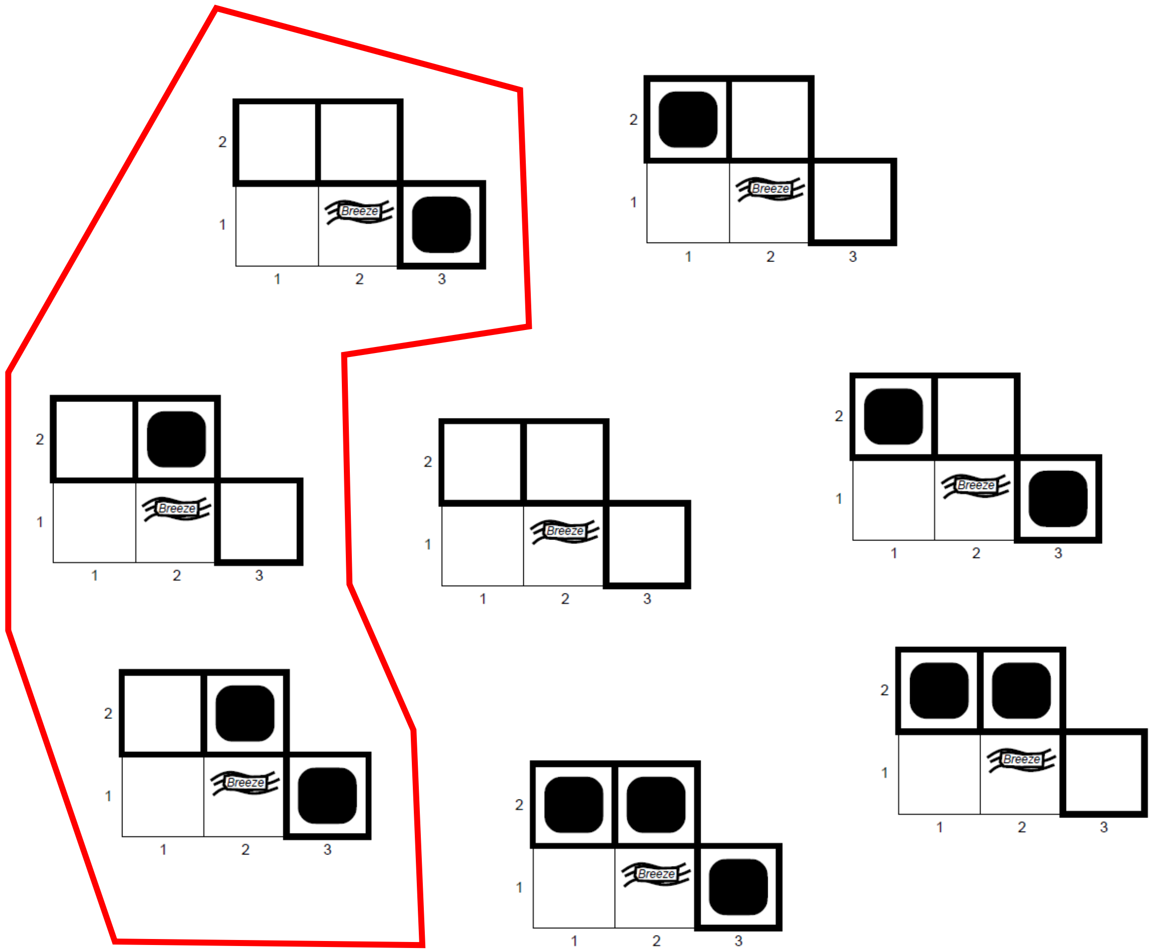
- **Entailment** – the idea that a sentence *follows logically* from another sentence.
  - In mathematical notation: $\alpha \models \beta$ means sentence α entails sentence β.
- The formal definition is that $\alpha \models \beta$ **iff** every model in which α is true, β is also true.
  - In mathematical notation: $\alpha \models \beta$, **iff** $M(\alpha) \subseteq M(\beta)$
- This means α is a **stronger** assertion than β
  - $M(\alpha)$ typically smaller than $M(\beta)$
- Example: $\alpha = x = 0$, $\beta = xy = 0$, $\alpha \models \beta$

ENTAILMENT, VISUALIZED



ENTAILMENT, VISUALIZED

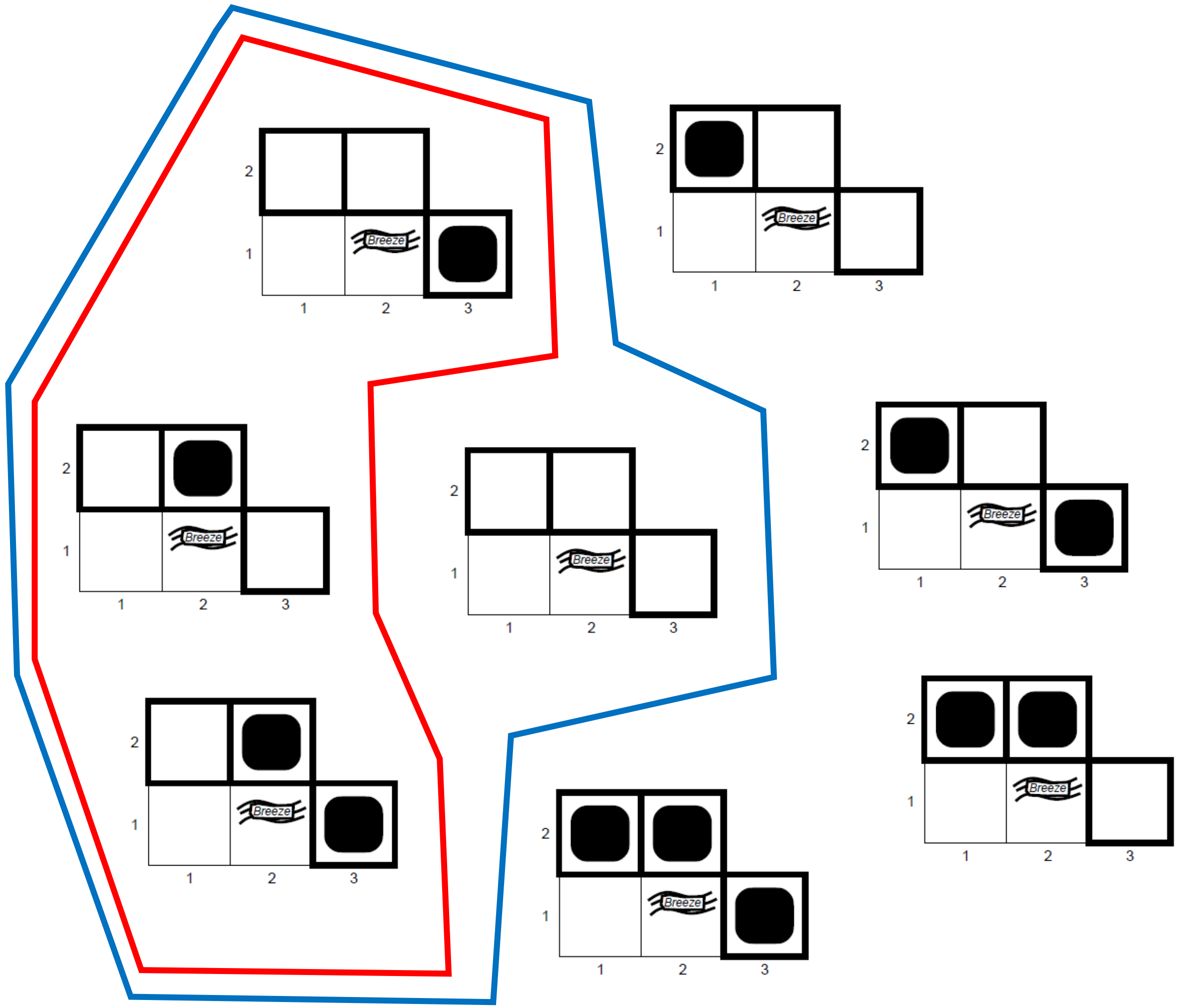
The red set is $M(KB)$, the set of all states that are compatible with the agent's KB, which is the agent's current understanding of reality/collection of perceptions, combined with the Wumpus World rules it was given.



ENTAILMENT, VISUALIZED

The blue set is $M(A)$, where A = [1, 2] is safe.

As you can see, $\mathrm{KB} \models A$. In every state consistent with reality ($M(KB)$, all models where KB = true), A is true, meaning that A is true, even as more knowledge is gained and KB shrinks.

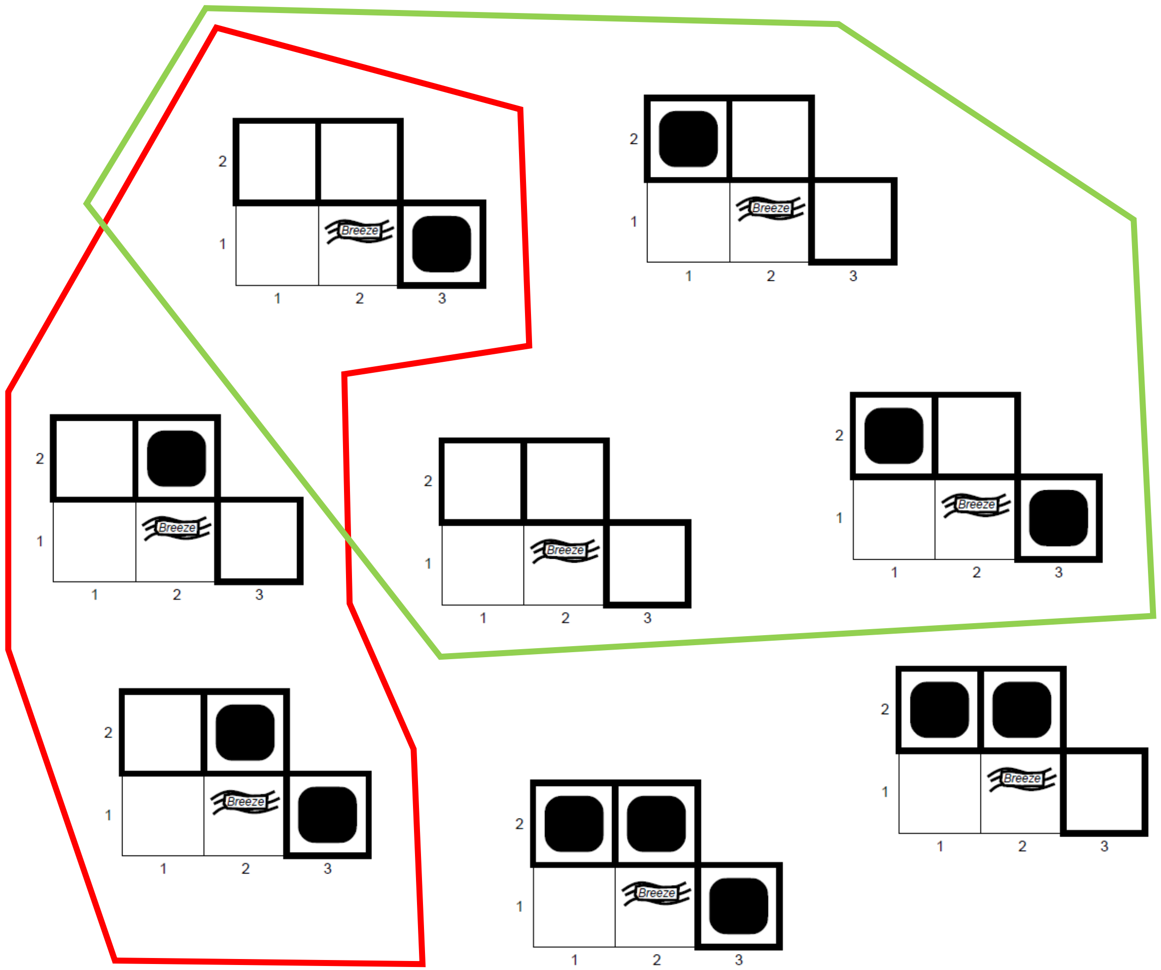


ENTAILMENT, VISUALIZED

The green set is $M(B)$, where B= [2, 2] is safe.

B is not true in all $M(KB)$, so we don't know if B is true.

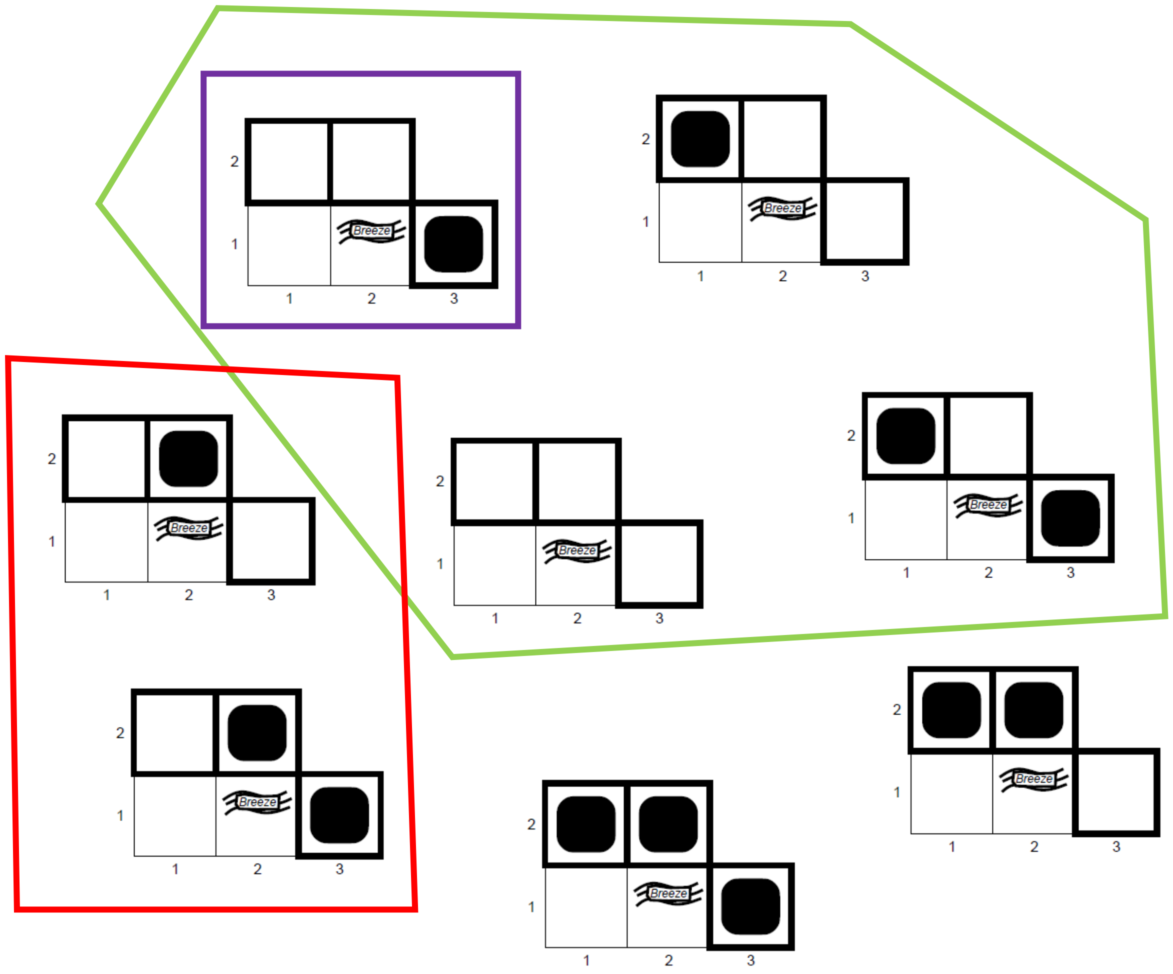
However, consider a situation where a new perception O is added to the KB (the agent moves)



ENTAILMENT, VISUALIZED

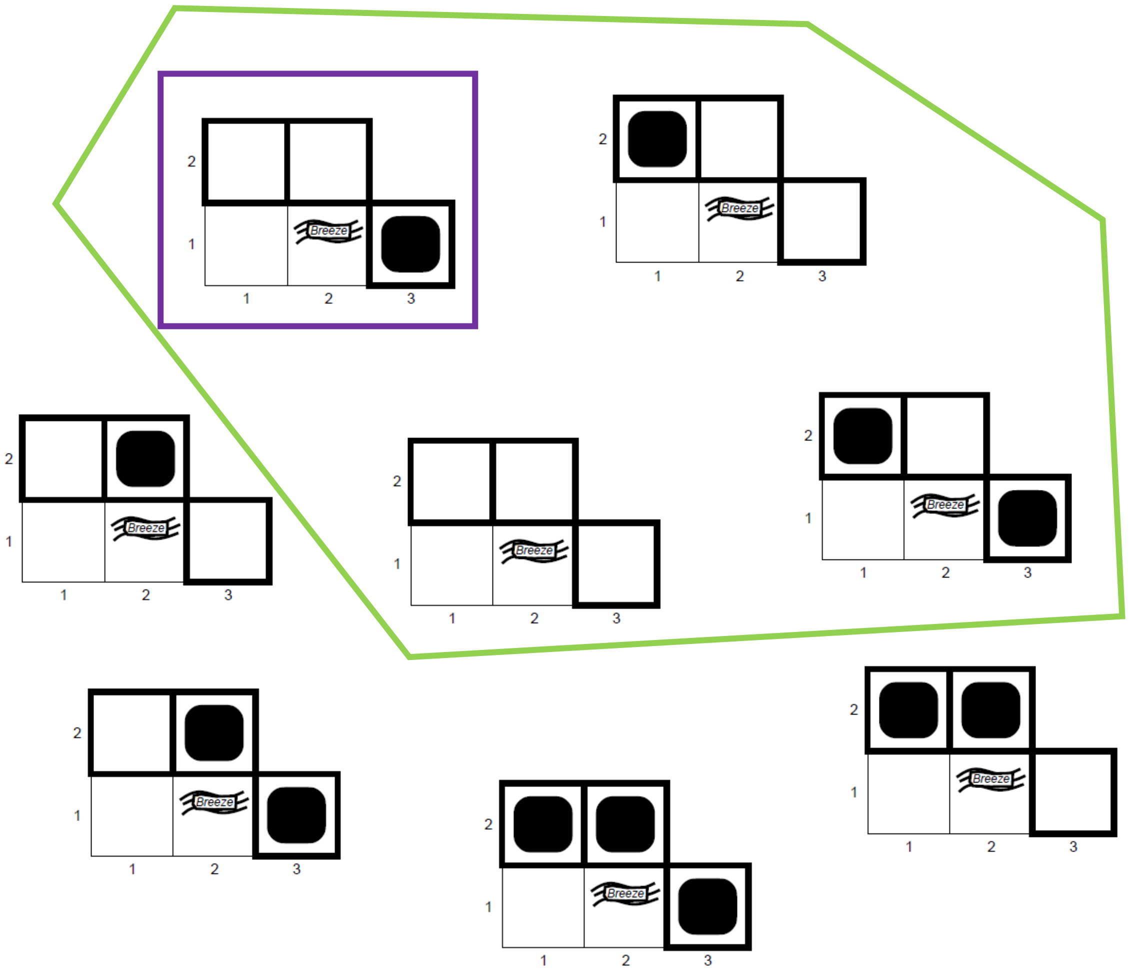
Let's call the model (state, to avoid confusion) boxed in purple S.

If O removes S from $M(KB)$, making the new $M(KB)$ in red, then we know B is not true, because no reality is compatible with it.



ENTAILMENT, VISUALIZED

However, if O removes every other state but S from $M(KB)$, making the new $M(KB)$ equal to the purple box, then we know B is true, because every reality is compatible with it ($\text{KB} \models B$).



ENTAILMENT, APPLIED

- Entailment can be applied to derive conclusions- that is, to carry out **logical inference**.
- The prior example is called **model checking**, because it enumerates through all possible models to verify entailment.

ABOUT THAT INFERENCE THING

- $KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure (inference algorithm) i
- An inference algorithm (i) is searching for a consequence or additional truth implied by KB, and α is the result.

SIMPLE INFERENCE EXAMPLE

KB

- `x = 2`
- `y = 2`
- `2 + 2 = 4`

$\xrightarrow[\text{into i}]{\text{KB fed}}$

i

- `Model Checking`
- `Resolution`
- `Forward or Backward Chaining`

$\xrightarrow[\text{Or proves}]{\text{Infers}}$

Alpha

- `x + y = 4`

CHARACTERISTICS OF INFERENCE ALGORITHMS

- **Soundness** or **Truth-preserving** – i is **sound** if
  - Whenever $KB \vdash_i \alpha$, it is also true that $\mathrm{KB} \models \alpha$
  - Basically, i doesn't make anything up.
- **Completeness** – i is **complete** if
  - Whenever $\mathrm{KB} \models \alpha$, it is also true that $KB \vdash_i \alpha$
  - **NOTE THE DIFFERENCE BETWEEN** $\vdash_i$ **AND** $\models$
- Both properties are highly desirable!

TYPES OF INFERENCE ALGORITHMS

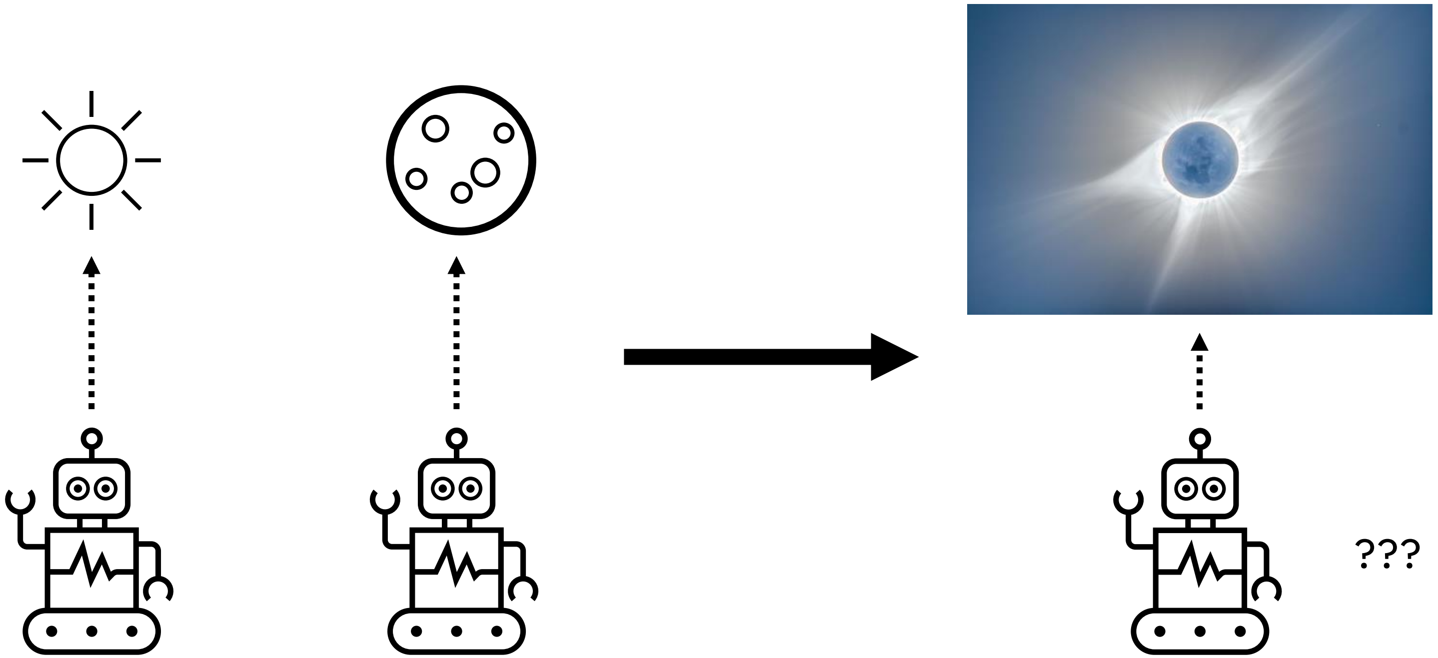
	Unsound	Sound
Incomplete	The algorithm may infer false sentences, and can’t infer all sentences KB entails.	The algorithm only infers true sentences, but it can’t infer all sentences KB entails.
Complete	The algorithm can infer all sentences KB entails, but it may additionally infer false sentences.	The algorithm infers only true sentences, and can infer all sentences KB entails.

A FINAL NOTE ON KB AGENTS

- An issues to consider is **grounding**.
- **Grounding** is the connection between logical reasoning processes and the real environment in which the agent exists.
- Simply: How do we know that KB is true in the real world?
- The answer depends on the validity of the sensors and the process of translating perception the KB agent’s KRL.

A FINAL NOTE ON KB AGENTS

- Even with perfect sensors and translation, a problem arises when a KB is **learning**.



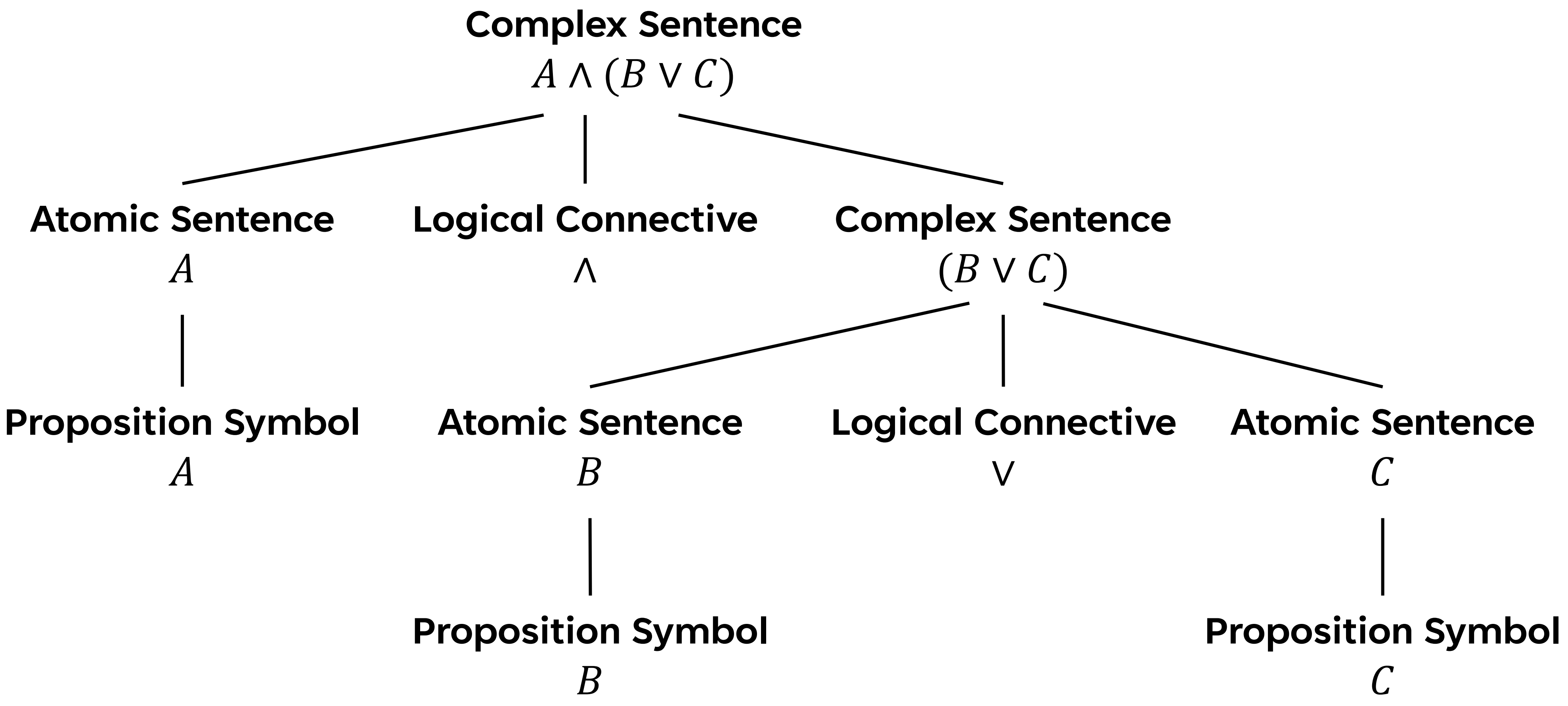
PROPOSITIONAL LOGIC

Thinking with Applied Discrete Mathematics

INTRODUCTION TO PROPOSITIONAL LOGIC

- **Propositional logic** is the simplest logic.
- Its core unit are **proposition symbols**.
- An **atomic sentence** is a sentence that contains of a single proposition symbol.
- **Complex sentences** are constructed from simpler sentences, using parentheses and operates called **logical connectives**.

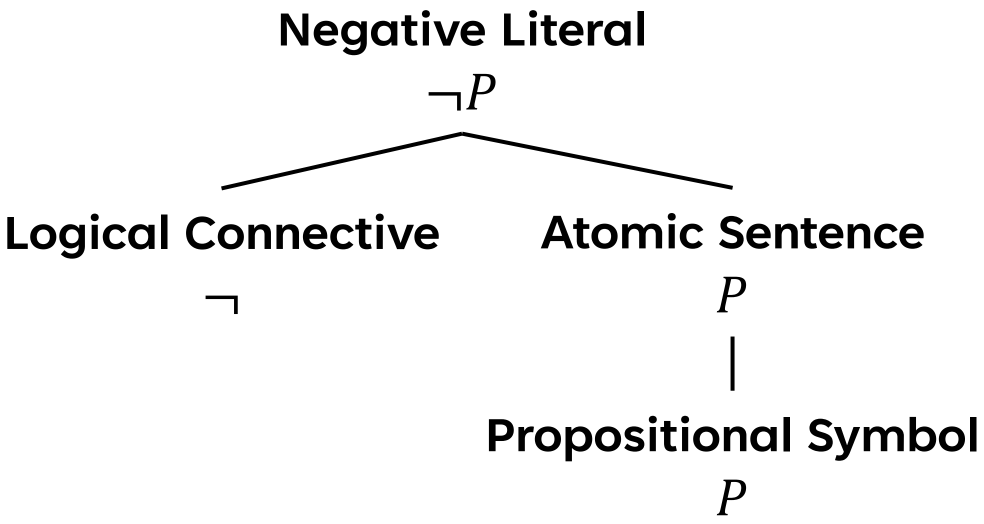
PROPOSITIONAL LOGIC SYNTAX



MEET THE CONNECTIVES - NOT

- $\neg$ (not)
- A sentence $\neg P$ is called the **negation** of P.
- A **literal** is either an atomic sentence (a **positive literal**) or a negated atomic sentence (a **negative literal**).

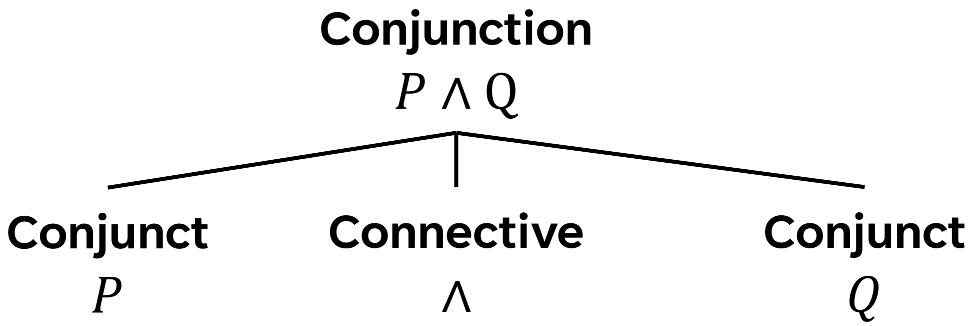
P	$\neg P$
`true`	`false`
`false`	`true`



MEET THE CONNECTIVES - AND

- $\wedge$ (and)
- A sentence whose main connective is $\wedge$, such as $\mathrm{P} \wedge \mathrm{Q}$, is called a **conjunction**.
- Its parts are the **conjuncts**.
- Note the similarity between $\wedge$ and the A in And.

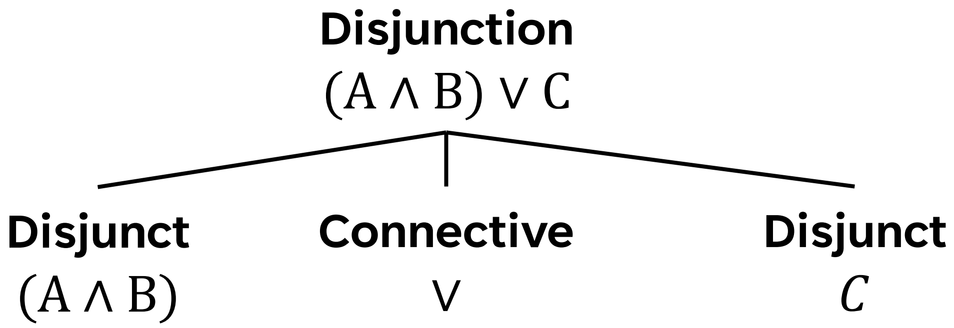
P	Q	$P \wedge \mathrm{Q}$
`false`	`false`	`false`
`false`	`true`	`false`
`true`	`false`	`false`
`true`	`true`	`true`



MEET THE CONNECTIVES - OR

- $\lor$ (or)
- A sentence whose main connective is $\lor$, such as $(A \land B) \lor C$, is called a **disjunction**.
- Its parts are **disjuncts**.

P	Q	$P \lor Q$
`false`	`false`	`false`
`false`	`true`	`true`
`true`	`false`	`true`
`true`	`true`	`true`



MEET THE CONNECTIVES - IMPLIES

- $\Rightarrow$ (implies)
- A sentence such as $(A \land B) \Rightarrow \neg C$, is called an **implication** (or conditional).
- Its **premise** or **antecedent** is $(A \land B)$.
- Its **conclusion** or **consequent** is $\neg C$.
- Implications are also known as **rules** or **if-then** statements.

P	Q	$P \Rightarrow Q$
false	false	true
false	true	true
true	false	false
true	true	true

Implication
$(A \land B) \Rightarrow \neg C$

Antecedent
$(A \land B)$

Connective
$\Rightarrow$

Consequent
$\neg C$

MEET THE CONNECTIVES – IF AND ONLY IF

- $\Leftrightarrow$ (if and only if)
- The sentence $\mathrm{A} \Leftrightarrow \mathrm{B}$ is a **biconditional**.
- Another framing: true when $\mathrm{P} \Rightarrow \mathrm{Q}$ and $\mathrm{Q} \Rightarrow \mathrm{P}$

P	Q	$P \Leftrightarrow \mathrm{Q}$
false	false	true
false	true	false
true	false	false
true	true	true

SUMMARY OF SEMANTICS

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

ANATOMY OF MODELS IN PROPOSITIONAL LOGIC

- In propositional logic, a model simply sets the **truth value** (*true* or *false*) for every proposition symbol:
  - Ex. $m_1 = \{P_{1,2} = \text{false}, P_{2,2} = \text{false}, P_{3,1} = \text{true}\}$
- Sentences can be applied to our models.
  - Ex. $s_1 = P_{1,2} \vee P_{3,1}$, s_1 is true in m_1
  - Ex. $s_2 = P_{1,2} \vee P_{2,2}$, s_2 is false in m_1
- With this knowledge, we can construct our first inference algorithm.

A SIMPLE INFERENCE PROCEDURE

- Consider a set of proposition symbols $P_i, i = [1,7]$, each representing a proposition/variable.
- Consider a set of sentences $S_i, i = [1,5]$, all expressed in terms of logical connectives and P_i
- Note that KB contains all sentences S_i.
- Consider a sentence α in terms of logical connectives and P_i

STEPS OF TT-ENTAILS?()

Step 1 — Enumerate through all possible models.

Step 2 — Test all models against KB, keeping those where KB is true.

Step 3 — We check if α holds for all models where KB is true.

Step 4 — If α holds for all models where KB is true, KB |= α

TRUTH TABLE FOR INFERENCE

Proposition Symbols | **Sentences**

All Possible Models

P_1	P_2	P_3	P_4	P_5	P_6	P_7	S_1	S_2	S_3	S_4	S_5	KB
false	false	false	false	false	false	false	true	true	true	true	false	false
false	false	false	false	false	false	true	true	true	false	true	false	false
…	…	…	…	…	…	…	…	…	…	…	…	…
false	true	false	false	false	false	false	true	true	false	true	true	false
false	true	false	false	false	false	true	true	true	true	true	true	true
false	true	false	false	false	true	false	true	true	true	true	true	true
false	true	false	false	false	true	true	true	true	true	true	true	true
false	true	false	false	true	false	false	true	false	false	true	true	false
…	…	…	…	…	…	…	…	…	…	…	…	…
true	true	true	true	true	true	true	false	true	true	false	true	false

INFERENCE BY ENUMERATION

- This algorithm is both **sound** and **complete**.
- The massive issue is that the time complexity **SUCKS**! $O(2^n)$ for n propositional symbols!
- Propositional entailment is co-NP-complete, so this isn't surprising.
- However, inference by enumeration makes no attempt to reduce this.

ENTAILMENT CONCEPTS & INFERENCE RULES

Ramping up for advanced inference algorithms

TOOLS FOR THEOREM PROVING

- The next step is entailment by **theorem proving** – applying rules of inference directly to the sentences in our KB to prove a desired sentence without models.
- Before that, we need some tools:
  - Logical equivalence
  - Validity
  - Satisfiability

LOGICAL EQUIVALENCE

- Two sentences are logically equivalent **iff** true in the same models: $\alpha \equiv \beta$ **iff** $\alpha \models \beta$ and $\beta \models \alpha$
- This allows a variety of logical equivalences to be applied to sentences:

$$
\begin{array}{rcll}
(\alpha \land \beta) & \equiv & (\beta \land \alpha) & \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv & (\beta \lor \alpha) & \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv & (\alpha \land (\beta \land \gamma)) & \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv & (\alpha \lor (\beta \lor \gamma)) & \text{associativity of } \lor \\
\neg(\neg\alpha) & \equiv & \alpha & \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\beta \Rightarrow \neg\alpha) & \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\alpha \lor \beta) & \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv & ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) & \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv & (\neg\alpha \lor \neg\beta) & \text{De Morgan} \\
\neg(\alpha \lor \beta) & \equiv & (\neg\alpha \land \neg\beta) & \text{De Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv & ((\alpha \land \beta) \lor (\alpha \land \gamma)) & \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv & ((\alpha \lor \beta) \land (\alpha \lor \gamma)) & \text{distributivity of } \lor \text{ over } \land
\end{array}
$$

VALIDITY & DEDUCTION THEOREM

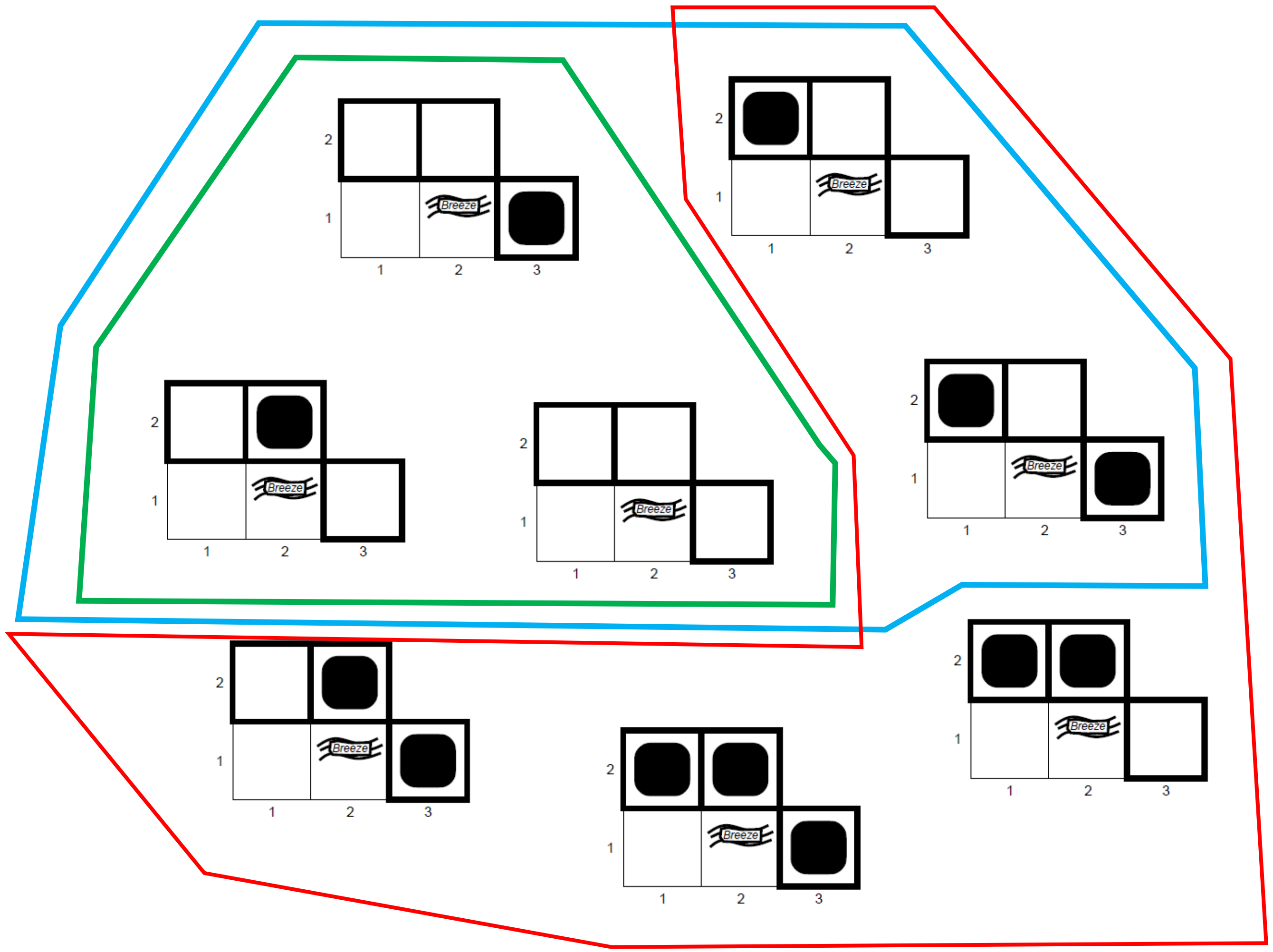
- A sentence is **valid** if it is true in **all** models. Valid sentences are also called **tautologies**- they are *necessarily* true.
  - Ex. $\text{True}, A \vee \neg A, \; A \Rightarrow A$
- Validity is connected to the inference via the **Deduction Theorem**: *For any sentences* α and β, $\alpha \models \beta$ *if and only if the sentence* $(\alpha \Rightarrow \beta)$ *is valid.*

PROVING THE DEDUCTION THEOREM

The green set is $M(\alpha)$. Every model in this set must be true in β (model must be in $M(\beta)$) for $\alpha \models \beta$. If any model in $M(\alpha)$ is not in $M(\beta)$, then α does not entail β. This logic is equivalent to $\alpha \Rightarrow \beta$ if α is true.

The red set are all models not in $M(\alpha)$. If any model is in this set, it doesn't change whether $\alpha \models \beta$. This logic is equivalent to $\alpha \Rightarrow \beta$ if α is false (remember, if the antecedent of an implication is false, the truth value is true).

Therefore, we have proved the equivalency of the truth tables of $\alpha \models \beta$ and $\alpha \Rightarrow \beta$, proving the Deduction Theorem.



SATISFIABILITY

- A sentence is **satisfiable** if it is true in **some** model.
  - Ex. $A \vee \mathrm{B}, \mathrm{C}$
- A sentence is **unsatisfiable** if it is true in **no** models.
  - Ex. $A \wedge \neg A$
- Satisfiability is connected to inference via the following:
  - $\alpha \models \beta$, **iff** $(\alpha \wedge \neg\beta)$ is unsatisfiable
- This is *reductio ad absurdum*, also called proof by **refutation** or proof by **contradiction**.

BUT WAIT, THERE'S MORE

- Now that we're done with **entailment concepts**, we must take a brief detour into **inference rules.**
- One applies inference rules to derive a **proof**- a chain of conclusions that leads to the desired goal.
- These rules are expressed in the following notation:

$$\frac{\text{sentenceOne}, \quad \text{sentenceTwo}}{\text{sentenceThree}}$$

If `sentenceOne` and `sentenceTwo` are true, then `sentenceThree` is true (inferred).

WELL-KNOWN INFERENCE RULES

Modus Ponens

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

And-Elimination

$$\frac{\alpha \land \beta}{\alpha}$$

ADDITIONAL PROOFS PRACTICE

Let us see how these inference rules and equivalences can be used in the wumpus world. We start with the knowledge base containing R_1 through R_5 and show how to prove $\neg P_{1,2}$, that is, there is no pit in [1,2]. First, we apply biconditional elimination to R_2 to obtain

$$R_6: \quad (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}) .$$

Then we apply And-Elimination to R_6 to obtain

$$R_7: \quad ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}) .$$

Logical equivalence for contrapositives gives

$$R_8: \quad (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})) .$$

Now we can apply Modus Ponens with R_8 and the percept R_4 (i.e., $\neg B_{1,1}$), to obtain

$$R_9: \quad \neg(P_{1,2} \vee P_{2,1}) .$$

Finally, we apply De Morgan's rule, giving the conclusion

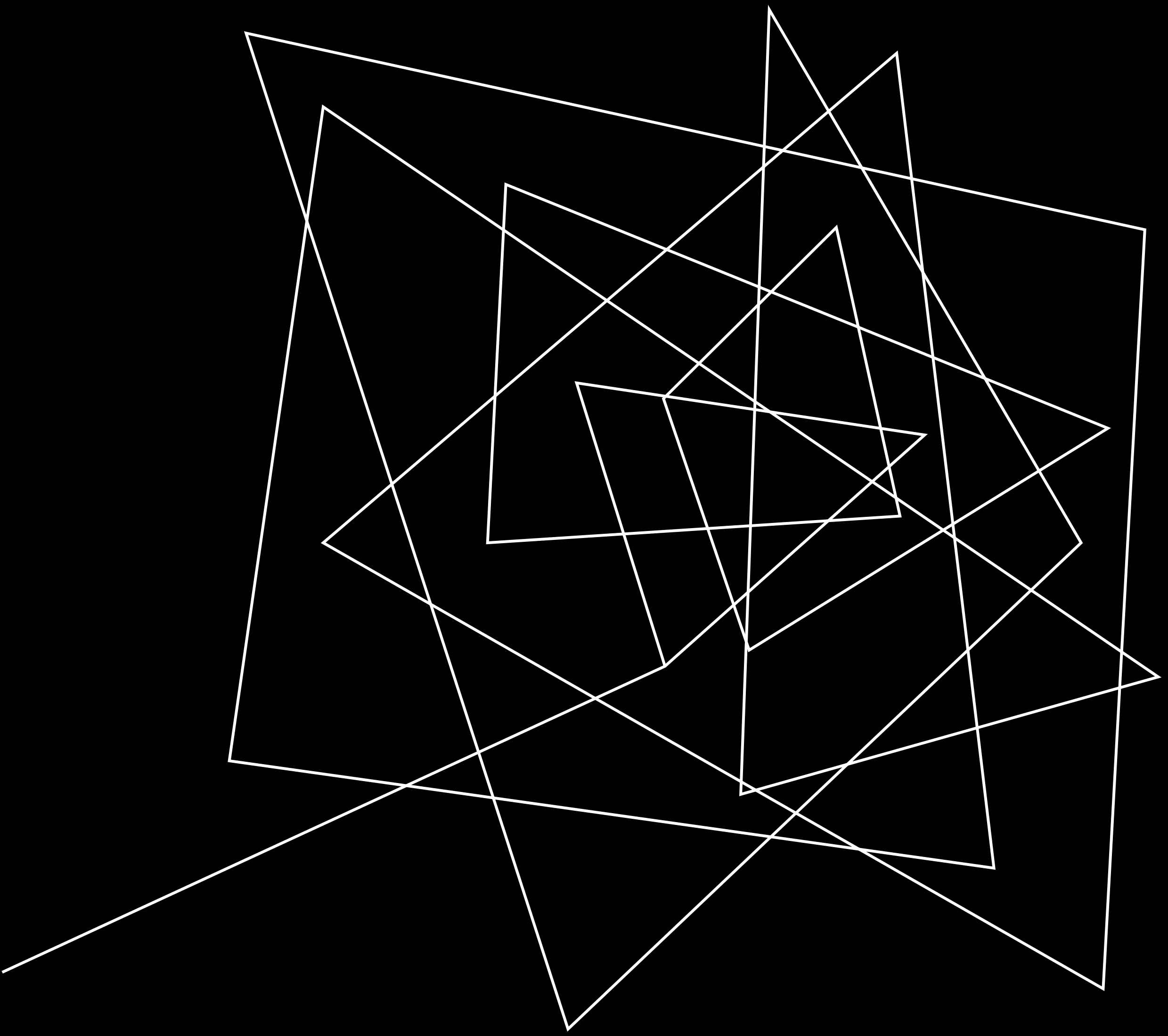
$$R_{10}: \quad \neg P_{1,2} \wedge \neg P_{2,1} .$$

That is, neither [1,2] nor [2,1] contains a pit.

We found this proof by hand, but we can apply any of the search algorithms in Chapter 3 to find a sequence of steps that constitutes a proof. We just need to define a proof problem as follows:

- INITIAL STATE: the initial knowledge base.
- ACTIONS: the set of actions consists of all the inference rules applied to all the sentences that match the top half of the inference rule.
- RESULT: the result of an action is to add the sentence in the bottom half of the inference rule.
- GOAL: the goal is a state that contains the sentence we are trying to prove.

Thus, searching for proofs is an alternative to enumerating models. In many practical cases *finding a proof can be more efficient because the proof can ignore irrelevant propositions, no matter how many of them there are.* For example, the proof given earlier leading to $\neg P_{1,2} \wedge \neg P_{2,1}$ does not mention the propositions $B_{2,1}$, $P_{1,1}$, $P_{2,2}$, or $P_{3,1}$. They can be ignored because the goal proposition, $P_{1,2}$, appears only in sentence R_2; the other propositions in R_2 appear only in R_4 and R_2; so R_1, R_3, and R_5 have no bearing on the proof. The same would hold even if we added a million more sentences to the knowledge base; the simple truth-table algorithm, on the other hand, would be overwhelmed by the exponential explosion of models.



ADVANCED INFERENCE ALGORITHMS

Literally the Dark Souls of inference

CATEGORIZATION OF INFERENCE ALGORITHMS

- Inference algorithms divide into (roughly) two broad categories:

Application of Inference Rules

- Sound generation of new sentences from old
- Requires proofs = sequences of inference rule applications
- Typically requires translation of sentences into a normal form
- **New algorithms: Resolution, forward and backward chaining, etc.**

Model Checking

- Truth table enumeration (always exponential in n)
- **New algorithms: Improved backtracking, such as DPLL, etc.**

HORN FORM

- Before we get into forward and backward chaining, we have to establish **Horn Form**.
- Horn Form = KB = conjunction of **Horn clauses**
- Horn Clause =
  - Proposition symbol (called a **fact**) or
  - (conjunction of symbols (called the **body**)) ⇒ (symbol (called the **head**))
- **Not all logical sentences can be expressed as horn form!**

HORN FORM SYNTAX

Horn Form

$C \land (B \Rightarrow A) \land (C \land D \Rightarrow B)$

Horn Clause

C

Horn Clause

$(B \Rightarrow A)$

Horn Clause

$(C \land D \Rightarrow B)$

Conjunction of Symbols

B

Symbol

A

Conjunction of Symbols

$C \land D$

Symbol

B

POP QUIZ!

Can a Horn clause be represented with any other connectives?

ALTERNATE HORN CLAUSE EXPRESSION

Original Expression	1. $C \land (B \Rightarrow A) \land (C \land D \Rightarrow B)$
Implication Elimination	2. $C \land (\neg B \lor \mathrm{A}) \land (C \land D \Rightarrow B)$
Implication Elimination	3. $C \land (\neg B \lor \mathrm{A}) \land (\neg(C \land D) \lor B)$
De Morgan	4. $C \land (\neg B \lor \mathrm{A}) \land (\neg C \lor \neg D \lor B)$

Implication Elimination

$\alpha \Rightarrow \beta \equiv (\neg\alpha \lor \beta)$

De Morgan

$\neg(\alpha \land \beta) \equiv (\neg\alpha \lor \neg\beta)$

BREAKING OUT THE TOOLBOX

- Remember Modus Pones? Now, we have an application that is complete for KBs in Horn Form.
- Can be used with forward chaining or backward chaining.
- Runs in **linear** time.

Modus Ponens (for Horn Form)

$$\frac{\alpha_1, \ldots, \alpha_n, \quad \alpha_1 \land \cdots \land \alpha_n \Rightarrow \beta}{\beta}$$

Alternative Perspective…

$$KB = \{S_1 \land \cdots \land S_n\}$$

$$\frac{KB, \quad KB \Rightarrow \beta}{\beta}$$

FORWARD CHAINING

- **Forward chaining (FC)** determines if a single proposition symbol *q* is entailed by a knowledge base of **definite clauses***.
  - Query is ***not*** a sentence, it's a symbol!
- It starts from known facts (positive literals) in the knowledge base.
- **Premise:** Proving a **symbol** is true in the KB by starting with true **symbols** and traversing the **horn clauses** that use them!

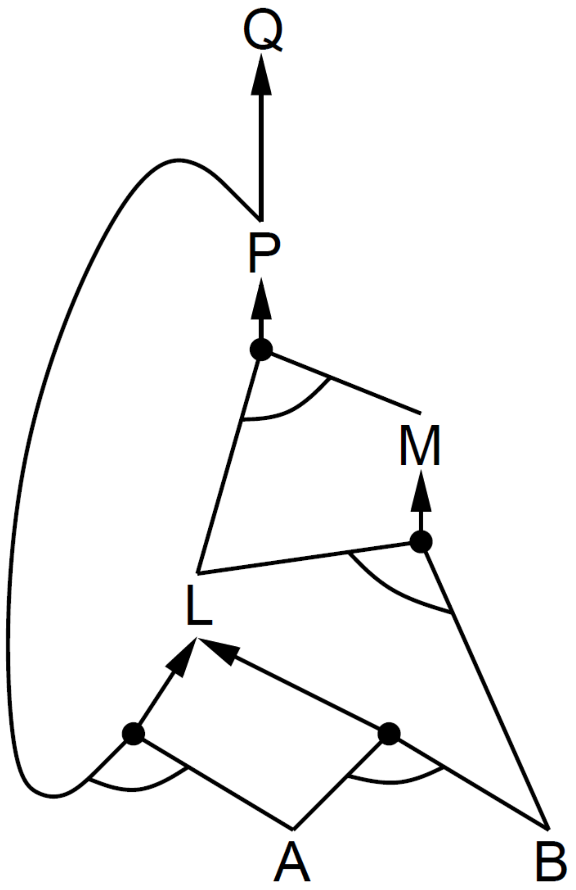
FORWARD CHAINING PSUEDOCODE

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional Horn clauses
          q, the query, a proposition symbol
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known in KB

  while agenda is not empty do
     p ← POP(agenda)
     unless inferred[p] do
        inferred[p] ← true
        for each Horn clause c in whose premise p appears do
           decrement count[c]
           if count[c] = 0 then do
              if HEAD[c] = q then return true
              PUSH(HEAD[c], agenda)
  return false
```

FORWARD CHAINING WALKTHROUGH

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



FORWARD CHAINING – PROOF OF COMPLETENESS

FC derives every atomic sentence entailed by KB.

1. FC reaches a **fixed point** where no new atomic sentences are derived.
2. Consider the final state as a model *m*, assigning truth values to symbols.
3. Every clause in the original KB is true in *m*.
4. Hence *m* is a model of KB.
5. If $KB \models q$, q is true in **every** model of KB, including *m*.

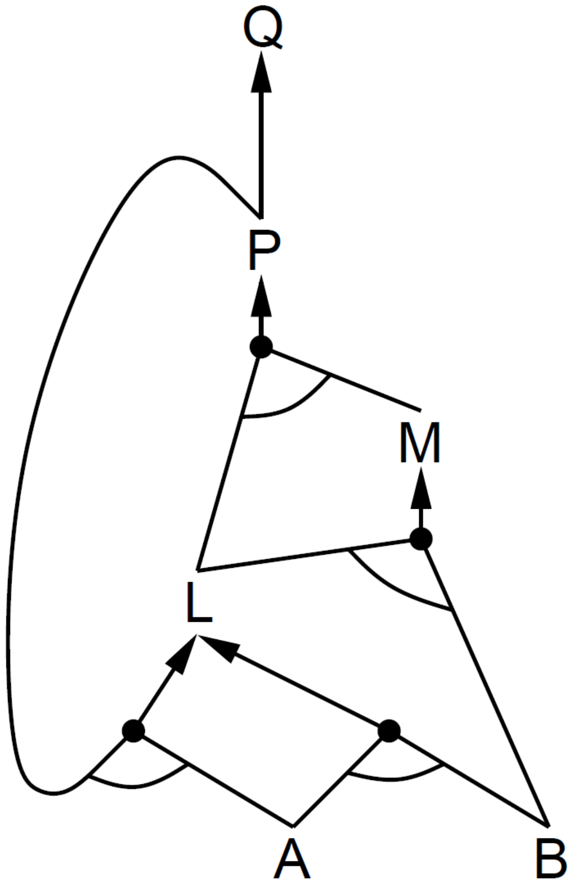
General idea: Construct any model of KB by sound inference, check α

BACKWARD CHAINING

- **Backward chaining (BC)** is like FC, but it works backwards from q instead of starting from positive literals.
- First, BC checks if q is known already. If so, it stops, as it has succeeded.
- If q is not known, prove all premises of some clause concluding with q.
- BC remembers clauses that have already been visited and will be visited to avoid repetition and loops.

BACKWARD CHAINING WALKTHROUGH

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



FORWARD VS. BACKWARD CHAINING

- Both are linear time complexity, though BC can be **much less** than linear in size of KB.
- This is because BC is **goal-driven**, focused on answering specific questions.
- FC, on the other hand, is **data-driven**. It is automatic, unconscious processing, and may do lots of work irrelevant to the goal.

RESOLUTION

- **Resolution** is an inference rule and algorithm.
- Intakes sentences in **Conjunctive Normal Form (CNF)**.
  - A conjunction of (disjunctions of literals)
  - Ex. $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$
- The resolution inference rule for CNF is as follows:
  - $\dfrac{\alpha_1 \lor \cdots \lor \alpha_k, \quad \beta_1 \lor \cdots \lor \beta_n}{\alpha_1 \lor \cdots \lor \alpha_{i-1} \lor \alpha_{i+1} \lor \cdots \lor \alpha_k \lor \beta_1 \lor \cdots \lor \beta_{j-1} \lor \beta_{j+1} \lor \cdots \lor \beta_n}$
    where α_i and β_j are complementary literals.
- Example:
  - $\dfrac{(A \lor B \lor C), \quad (D \lor \neg B, F)}{A \lor C \lor D \lor F}$

RESOLUTION ALGORITHM

Proof by contradiction, i.e., show $KB \land \neg\alpha$ 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

  $clauses \leftarrow$ the set of clauses in the CNF representation of $KB \land \neg\alpha$
  $new \leftarrow \{\}$
  loop do
    for each C_i, C_j **in** $clauses$ **do**
      $resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
      if $resolvents$ contains the empty clause **then return** *true*
      $new \leftarrow new \cup resolvents$
    if $new \subseteq clauses$ **then return** *false*
    $clauses \leftarrow clauses \cup new$

DPLL

```
Algorithm DPLL
    Input: A set of clauses Φ.
    Output: A truth value indicating whether Φ is satisfiable.
```

```
function DPLL(Φ)
    // unit propagation:
    while there is a unit clause {l} in Φ do
        Φ ← unit-propagate(l, Φ);
    // pure literal elimination:
    while there is a literal l that occurs pure in Φ do
        Φ ← pure-literal-assign(l, Φ);
    // stopping conditions:
    if Φ is empty then
        return true;
    if Φ contains an empty clause then
        return false;
    // DPLL procedure:
    l ← choose-literal(Φ);
    return DPLL(Φ ∧ {l}) or DPLL(Φ ∧ {¬l});
```

- "←" denotes assignment. For instance, "*largest* ← *item*" means that the value of *largest* changes to the value of *item*.
- "**return**" terminates the algorithm and outputs the following value.

THANK YOU

Joshua Sheldon

jsheldon2022@my.fit.edu