CMSC 471
Propositional and First-Order Logic

KMA Solaiman
ksolaima@umbc.edu

Some slides courtesy Tim Finin

Big Ideas

- **Logic:** great knowledge representation (KR) language for many AI problems
- **Propositional logic:** simple foundation and fine for many AI problems
- **First order logic (FOL):** more expressive as a KR language; needed for many AI problems
- **Variations** on classical FOL are common: horn logic, higher-order logic, modal logic, three-valued logic, probabilistic logic, fuzzy logic, etc.

AI Use Cases for Logic

Logic has many use cases even in a time dominated by deep learning, including these examples:

- Modeling and using knowledge
- Allowing agents to develop complex plans to achieve a goal and create optimal plans
- Defining and using semantic knowledge graphs such as schema.org and Wikidata
- Adding features to neural network systems

Question #2

Try to determine, as quickly as you can, if the argument is logically valid. Does the conclusion follow the premises?

- **(P) All roses are flowers**
- **(P) Some flowers fade quickly**
- **(C) Therefore some roses fade quickly**

Question #2

Try to determine, as quickly as you can, if the argument is logically valid. Does the conclusion follow the premises?

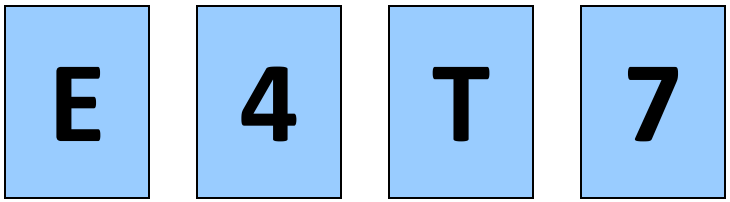
- All roses are flowers
- Some flowers fade quickly
- Therefore some roses fade quickly

It is possible that there are no roses among the flowers that fade quickly

Wason Selection Task

- I have a pack of cards; each has a *letter* written on one side and a *number* on the other
- I claim the following rule is true:

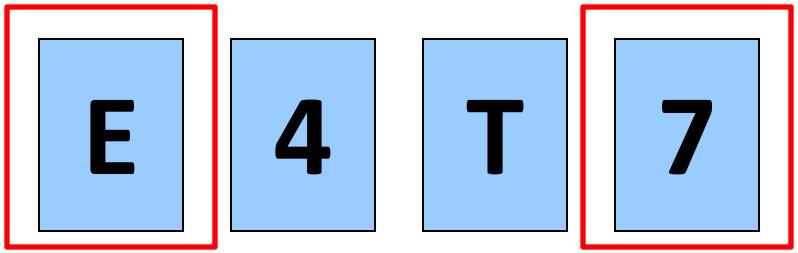
  If a card has a *vowel* on one side, then it has an *even number* on the other
- Which cards should you turn over in order to decide whether the rule is true or false?



Wikipedia

Wason Selection Task

- Wason (1966) showed that people are bad at this task
- To disprove rule P=>Q, find a situation in which P is true but Q is false, i.e., show $P \wedge \sim Q$
- To disprove **vowel => even**, find a card with a vowel and an odd number
- Thus, turn over the cards showing **vowels** and those showing **odd numbers**



Logic as a Methodology

Even if people don't use formal logical reasoning for solving a problem, logic might be a good approach for AI for a number of reasons

- Airplanes don't need to flap their wings
- Logic may be a good implementation strategy
- Solution in a formal system can offer other benefits, e.g., letting us prove properties of the approach

•See neats vs. scruffies

Knowledge-based agents

- Knowledge-based agents have a knowledge base (KB) and an inference system
- **KB:** a set of representations of facts believed true
- Each individual representation is called a **sentence**
- Sentences are expressed in a **knowledge representation language**
- The agent operates as follows:
  1. It **TELL**s the KB what it perceives
  2. It **ASK**s the KB what action it should perform
  3. It performs the chosen action

Motivation: smart personal assistant

Tell information

Ask questions

Use natural language!

Need to:

- Digest **heterogenous** information
- Reason **deeply** with that information

Negation in Natural Language

- We often model the **meaning of natural language sentences** as **a logic statements**
- This maps these into equivalent statements
  - All elephants are gray
  - No elephant are not gray
- Double negation is common in informal language: th*at won't do you no good*
- *But what does this mean: we cannot underestimate the importance of logic*

Language

Language *is a mechanism for expression.*

Natural languages (informal):

English: *Two divides even numbers.*

German: *Zwei dividieren geraden zahlen.*

Programming languages (formal):

Python: `def even(x): return x % 2 == 0`

C++: `bool even(int x) { return x % 2 == 0; }`

Logical languages (formal):

First-order-logic: $\forall x.\text{Even}(x) \rightarrow \text{Divides}(x, 2)$

Architecture of a KB agent

- **Knowledge Level**
  - Most abstract: describe agent by what it knows
  - Ex: Autonomous vehicle knows Golden Gate Bridge connects San Francisco with the Marin County
- **Logical Level**
  - Level where knowledge is encoded into *sentences*
  - Ex: **links**(GoldenGateBridge, SanFran, MarinCounty)
- **Implementation Level**
  - Software representation of sentences, e.g.

```
(links goldengatebridge sanfran marincounty)
```

Does your agent have complete knowledge?

- Closed world assumption (CWA): the lack of knowledge is assumed to mean it's false
- Open world assumption: no such assumption is made

Q: Why would we ever make a closed world assumption?

Logic

- **Logics** are formal languages for representing information so that conclusions can be drawn

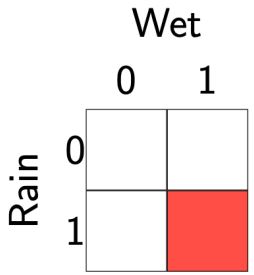
Ingredients of a Logic

- **Syntax** defines a set of valid sentences/formulas ($\mathbf{f}$ **or** $\boldsymbol{\alpha}$)

  Example: Rain $\wedge$ Wet

- **Semantics** define the "meaning" of sentences: $\mathbf{m}/\mathbf{w}$
  - i.e., for each formula, specify a set of **models** (assignments / configurations of the world)

Example:



Inference rules: given f, what new formulas g can be added that are guaranteed to follow ($\frac{f}{g}$)?

Example: from Rain $\wedge$ Wet, derive Rain

Syntax versus semantics

Syntax: what are valid expressions in the language?

Semantics: what do these expressions mean?

Different syntax, same semantics (5):

$$2 + 3 \Leftrightarrow 3 + 2$$

Same syntax, different semantics (1 versus 1.5):

$$3 \ / \ 2 \text{ (Python 2.7)} \not\Leftrightarrow 3 \ / \ 2 \text{ (Python 3)}$$

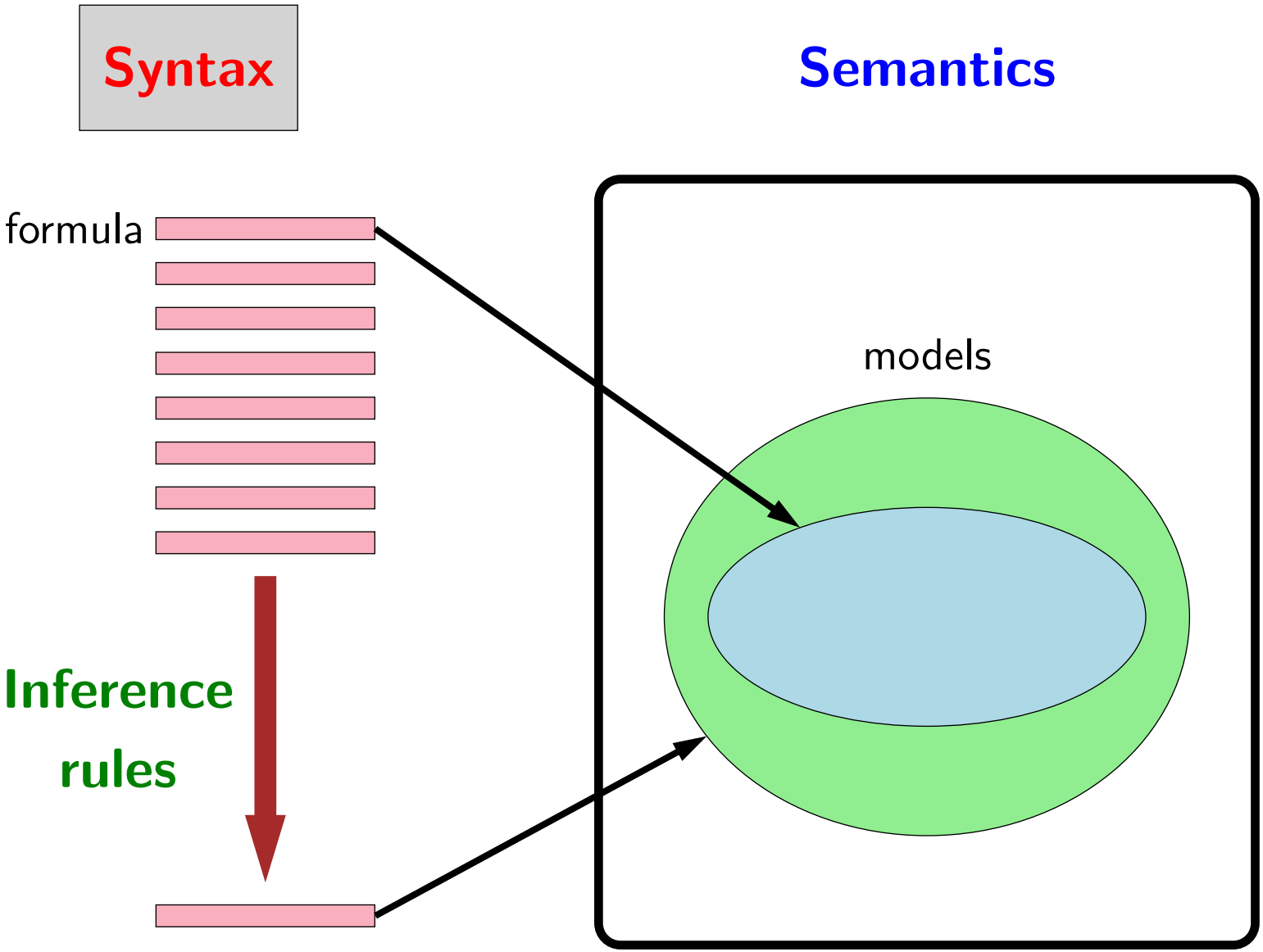
Logics

- **Propositional logic with only Horn clauses**
- **Propositional logic**
- Modal logic
- **First-order logic with only Horn clauses**
- **First-order logic**
- Second-order logic
- ...

Key idea: tradeoff

Balance **expressivity** and **computational efficiency**.

Propositional logic



- We begin with the syntax of propositional logic: what are the allowable formulas?

Syntax of propositional logic

Propositional symbols (atomic formulas): A, B, C

Logical connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$

Build up formulas recursively—if f and g are formulas, so are the following:

- Negation: $\neg f$
- Conjunction: $f \wedge g$
- Disjunction: $f \vee g$
- Implication: $f \rightarrow g$
- Biconditional: $f \leftrightarrow g$

- The building blocks of the syntax are the propositional symbols and connectives. The set of propositional symbols can be anything (e.g., A, Wet, etc.), but the set of connectives is fixed to these five.
- All the propositional symbols are **atomic formulas** (also called atoms). We can **recursively** create larger formulas by combining smaller formulas using connectives.

Syntax of propositional logic

- Formula: A
- Formula: $\neg A$
- Formula: $\neg B \rightarrow C$
- Formula: $\neg A \wedge (\neg B \rightarrow C) \vee (\neg B \vee D)$
- Formula: $\neg\neg A$
- Non-formula: $A\neg B$
- Non-formula: $A + B$

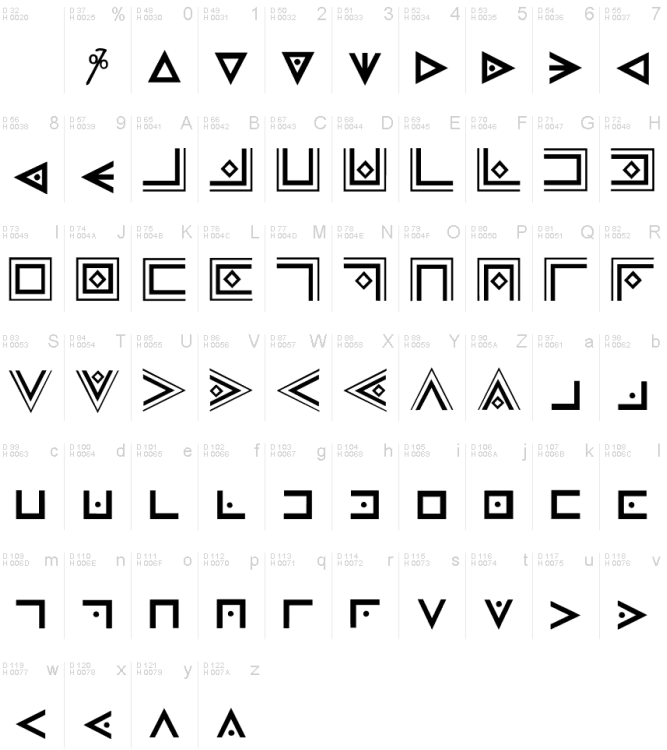
- Here are some examples of valid and invalid propositional formulas.

Syntax of propositional logic

Key idea: syntax provides symbols

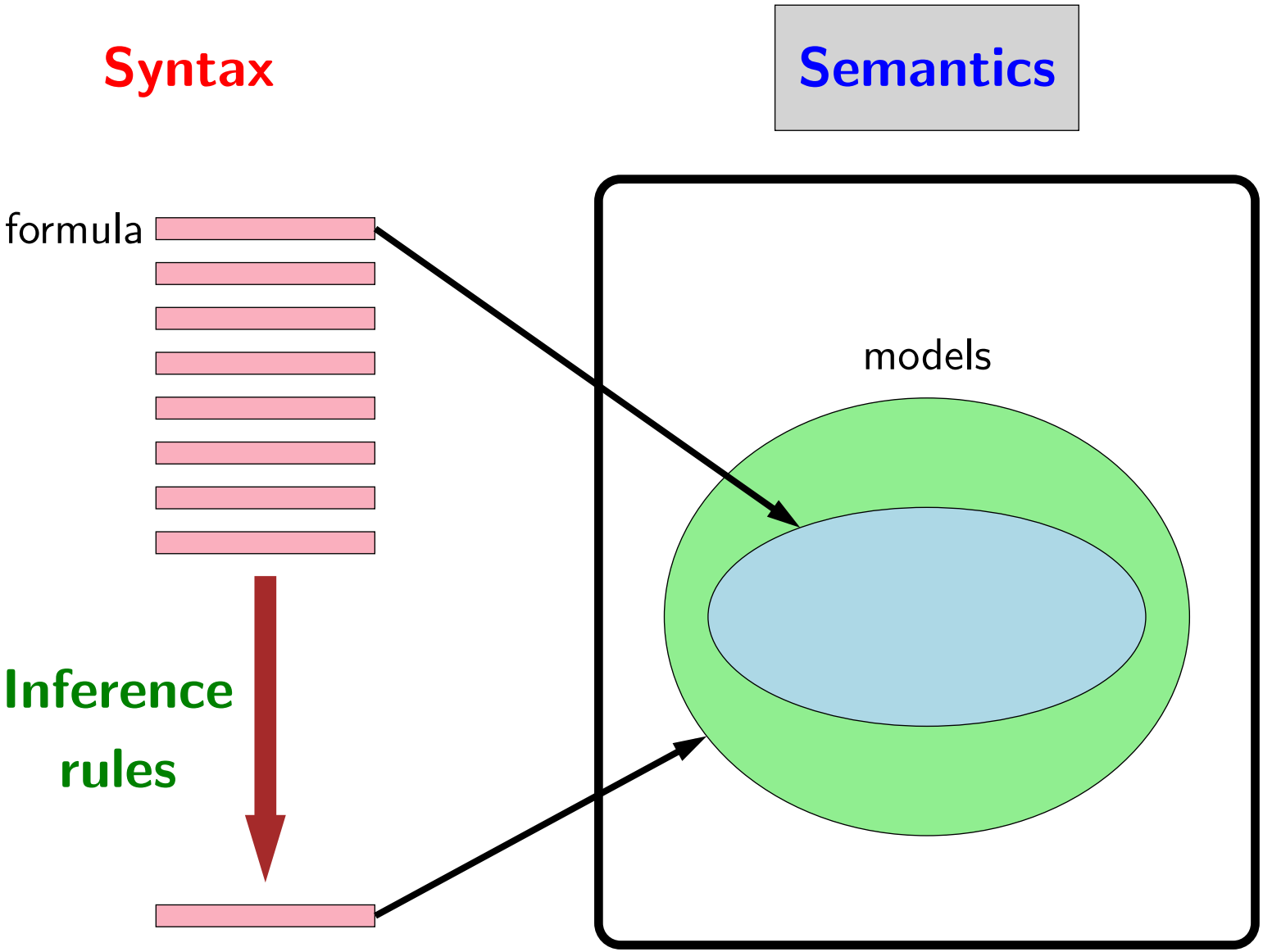
Formulas by themselves are just symbols (syntax).

No meaning yet (semantics)!



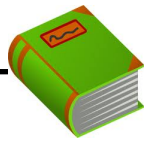
- It's important to remember that whenever we talk about syntax, we're just talking about symbols; we're not actually talking about what they mean — that's the role of semantics. Of course it will be difficult to ignore the semantics for propositional logic completely because you already have a working knowledge of what the symbols mean.

Propositional logic



- Having defined the syntax of propositional logic, let's talk about their semantics or meaning.

Model



Definition: model

A **model** w in propositional logic is an **assignment** of truth values to propositional symbols.

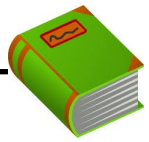
Example:

- 3 propositional symbols: A, B, C
- $2^3 = 8$ possible models w:

$$\{A:0, B:0, C:0\}$$
$$\{A:0, B:0, C:1\}$$
$$\{A:0, B:1, C:0\}$$
$$\{A:0, B:1, C:1\}$$
$$\{A:1, B:0, C:0\}$$
$$\{A:1, B:0, C:1\}$$
$$\{A:1, B:1, C:0\}$$
$$\{A:1, B:1, C:1\}$$

- In logic, the word **model** has a special meaning, quite distinct from the way we've been using it in the class (quite an unfortunate collision). A model (in the logical sense) represents a possible state of affairs in the world. In propositional logic, this is an assignment that specifies a truth value (true or false) for each propositional symbol.

Interpretation function



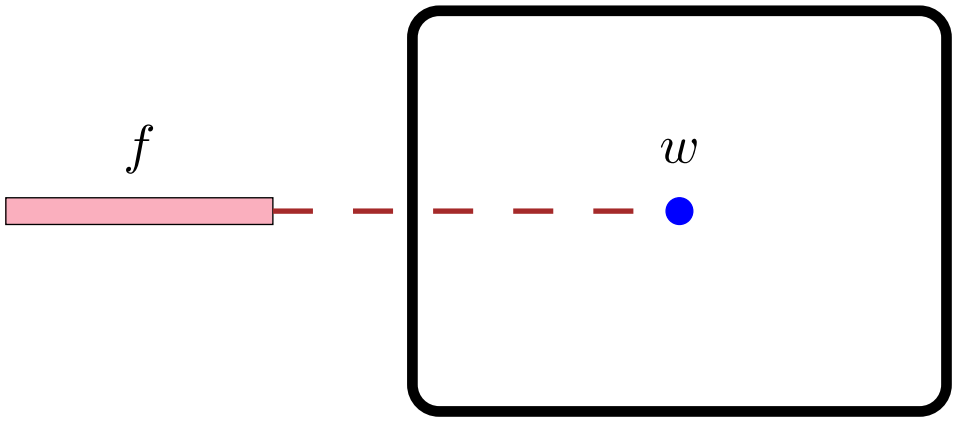
Definition: interpretation function

Let f be a formula.

Let w be a model.

An **interpretation function** $\mathcal{I}(f, w)$ returns:

- true (1) (say that w satisfies f)
- false (0) (say that w does not satisfy f)



- The semantics is given by an **interpretation function**, which takes a formula f and a model w, and returns whether w satisfies f. In other words, is f true in w?
- For example, if f represents "it is Wednesday" and w corresponds to right now, then $\mathcal{I}(f, w) = 1$. If w corresponded to yesterday, then $\mathcal{I}(f, w) = 0$.

Interpretation function: definition

Base case:

- For a propositional symbol p (e.g., A, B, C): $\mathcal{I}(p, w) = w(p)$

Recursive case:

- For any two formulas f and g, define:

$\mathcal{I}(f,w)$	$\mathcal{I}(g,w)$	$\mathcal{I}(\neg f,w)$	$\mathcal{I}(f \wedge g,w)$	$\mathcal{I}(f \vee g,w)$	$\mathcal{I}(f \rightarrow g,w)$	$\mathcal{I}(f \leftrightarrow g,w)$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

- The interpretation function is defined recursively, where the cases neatly parallel the definition of the syntax.
- Formally, for propositional logic, the interpretation function is fully defined as follows. In the base case, the interpretation of a propositional symbol p is just gotten by looking p up in the model w. For every possible value of $(\mathcal{I}(f, w), \mathcal{I}(g, w))$, we specify the interpretation of the combination of f and g.

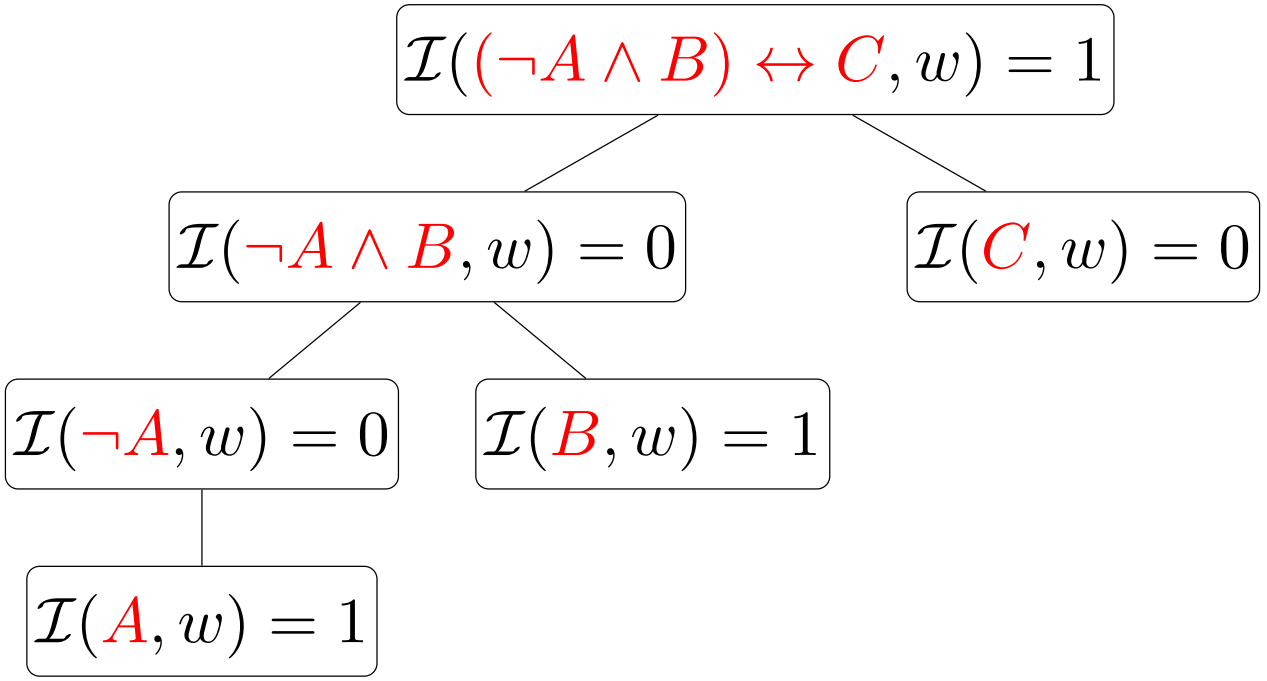
Interpretation function: example

Example: interpretation function

Formula: $f = (\neg A \wedge B) \leftrightarrow C$

Model: $w = \{A : 1, B : 1, C : 0\}$

Interpretation:



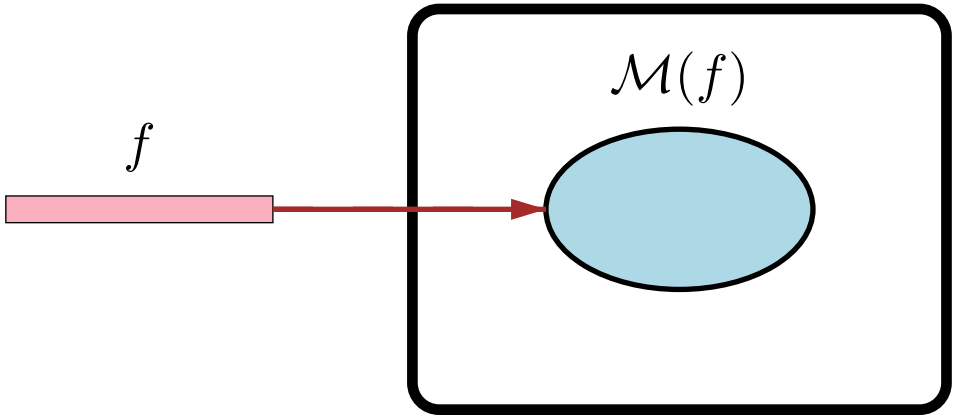
- For example, given the formula, we break down the formula into parts, recursively compute the truth value of the parts, and then finally combines these truth values based on the connective.

Formula represents a set of models

So far: each formula f and model w has an interpretation $\mathcal{I}(f, w) \in \{0, 1\}$

Definition: models

Let $\mathcal{M}(f)$ be the set of **models** w for which $\mathcal{I}(f, w) = 1$.



- So far, we've focused on relating a single model. A more useful but equivalent way to think about semantics is to think about the formula $\mathcal{M}(f)$ as **a set of models** — those for which $\mathcal{I}(f, w) = 1$.

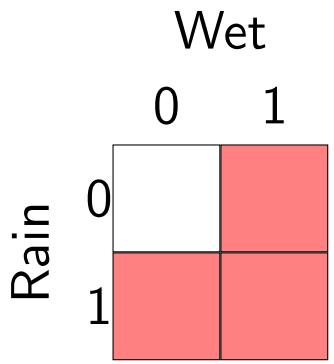
Models: example

Formula:

$$f = \text{Rain} \vee \text{Wet}$$

Models:

$\mathcal{M}(f) =$

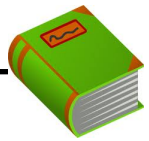


Key idea: compact representation

A **formula** *compactly* represents a set of **models**.

- In this example, there are four models for which the formula holds, as one can easily verify. From the point of view of $\mathcal{M}$, a formula's main job is to define a set of models.
- Recall that a model is a possible configuration of the world. So a formula like "it is raining" will pick out all the hypothetical configurations of the world where it's raining; in some of these configurations, it will be Wednesday; in others, it won't.

Knowledge base



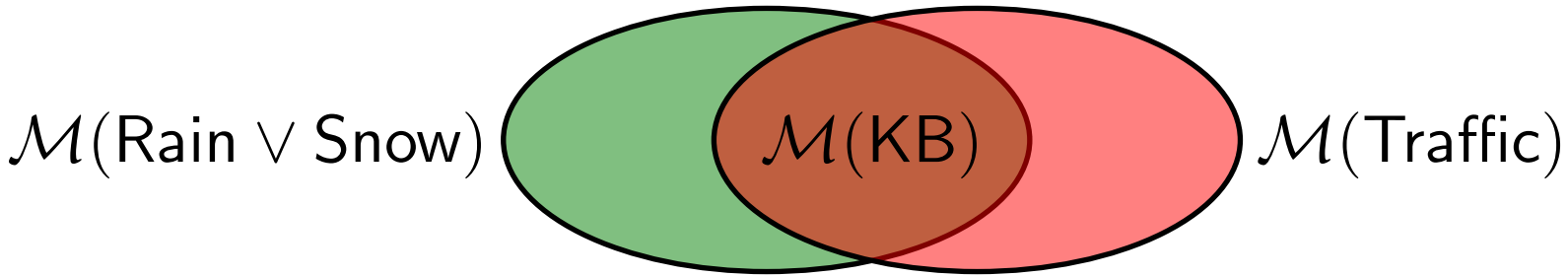
Definition: Knowledge base

A **knowledge base** KB is a set of formulas representing their conjunction / intersection:

$$\mathcal{M}(\text{KB}) = \bigcap_{f \in \text{KB}} \mathcal{M}(f).$$

Intuition: KB specifies constraints on the world. $\mathcal{M}(\text{KB})$ is the set of all worlds satisfying those constraints.

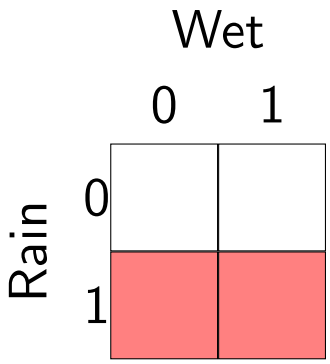
Let $\text{KB} = \{\text{Rain} \vee \text{Snow}, \text{Traffic}\}$.



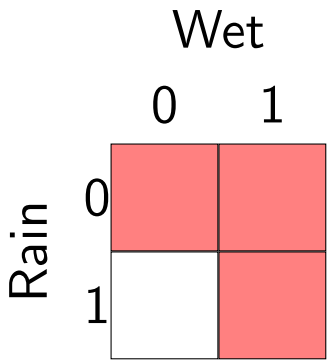
- If you take a set of formulas, you get a **knowledge base**. Each knowledge base defines a set of models — exactly those which are satisfiable by all the formulas in the knowledge base.
- Think of each formula as a fact that you know, and the **knowledge** is just the collection of those facts. Each fact narrows down the space of possible models, so the more facts you have, the fewer models you have.

Knowledge base: example

$\mathcal{M}(\text{Rain})$

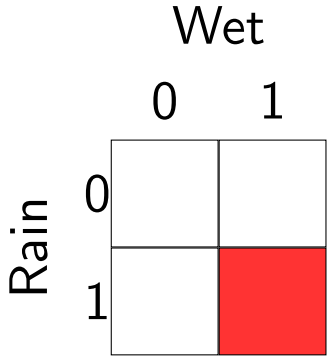


$\mathcal{M}(\text{Rain} \rightarrow \text{Wet})$



Intersection:

$\mathcal{M}(\{\text{Rain}, \text{Rain} \rightarrow \text{Wet}\})$



- As a concrete example, consider the two formulas Rain and Rain $\rightarrow$ Wet. If you know both of these facts, then the set of models is constrained to those where it is raining and wet.

Models for a KB

- KB: [PVQ, P→R, Q→R]
- What are the formulas?

  f1: PVQ
  f2: P→R
  f3: Q→R
- What are the propositional variables?

  P, Q, R
- What are the candidate models?

1) Consider all **eight** possible assignments of T|F to P, Q, R
2) Check if each sentence is consistent with the model

P	Q	R	s1	s2	s3
F	F	F	x	✓	✓
F	F	T	x	✓	✓
F	T	F	✓	✓	x
F	T	T	✓	✓	✓
T	F	F	✓	x	✓
T	F	T	✓	✓	✓
T	T	F	✓	x	x
T	T	T	✓	✓	✓

Here **x** means the model makes the sentence False and ✓means it doesn't make it False

Models for a KB

- KB: [PVQ, P→R, Q→R]
- What are the formulas?
  f1: PVQ
  f2: P→R
  f3: Q→R
- What are the propositional variables?
  P, Q, R
- What are the candidate models?

1) Consider all **eight** possible assignments of T|F to P, Q, R
2) Check truth tables for consistency, eliminating any row that does not make every KB sentence true

P	Q	R	s1	s2	s3
F	F	F	✗	✓	✓
F	F	T	✗	✓	✓
F	T	F	✓	✓	✗
F	**T**	**T**	✓	✓	✓
T	F	F	✓	✗	✓
T	**F**	**T**	✓	✓	✓
T	T	F	✓	✗	✗
T	**T**	**T**	✓	✓	✓

- Only 3 models are consistent with KB
- R true in all of them
- Therefore, R is true and can be added to the KB

A simple example

The KB

$\mathbf{P}$

$\mathbf{Q} \vee \neg \mathbf{R}$

The KB has 2 formulas.

The KB has 3 variables.

The KB has 3 models for which $I(f, w) = 1$.

Another way to look at this is:
$\mathcal{M}(\boldsymbol{P})$ is true in first 3
$\mathcal{M}(\mathbf{Q} \vee \neg \mathbf{R})$is true in first 3
So $\mathcal{M}(\boldsymbol{KB})$ is first 3

Models for the KB, $\mathcal{M}(\boldsymbol{KB})$

P	Q	R	KB
T	T	F	T
T	T	T	T
T	F	F	T
T	F	T	F
F	T	F	F
F	T	T	F
F	F	T	F
F	F	F	F

Another simple example

The KB

$P \wedge Q$

$R \wedge \neg P$

The KB has 2 formulas.

The KB has 3 variables.

Models for the KB

P	Q	R

The KB has no models. There is no assignment of True or False to every variable that makes every sentence in the KB true

Adding to the knowledge base

Adding more formulas to the knowledge base:

$$\text{KB} \quad \longrightarrow \quad \text{KB} \cup \{f\}$$

Shrinks the set of models:

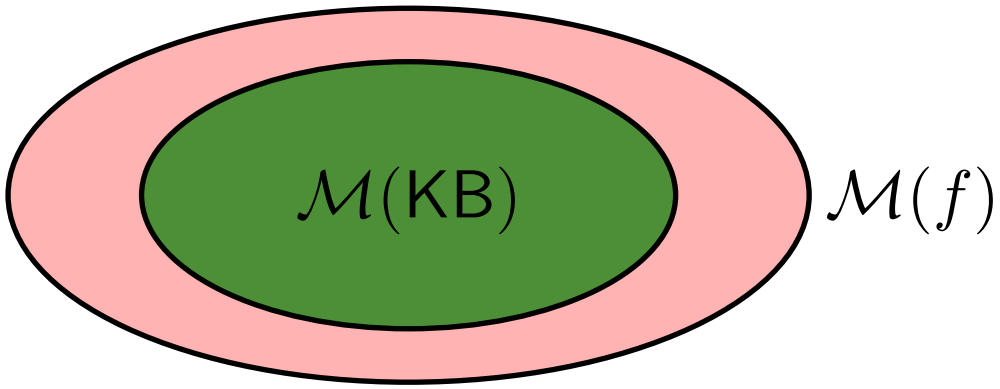
$$\mathcal{M}(\text{KB}) \quad \longrightarrow \quad \mathcal{M}(\text{KB}) \cap \mathcal{M}(f)$$

How much does $\mathcal{M}(\text{KB})$ shrink?

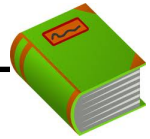
[whiteboard]

- We should think about a knowledge base as carving out a set of models. Over time, we will add additional formulas to the knowledge base, thereby winnowing down the set of models.
- Intuitively, adding a formula to the knowledge base imposes yet another constraint on our world, which naturally decreases the set of possible worlds.
- Thus, as the number of formulas in the knowledge base gets larger, the set of models gets smaller.
- A central question is how much f shrinks the set of models. There are three cases of importance.

Entailment



Intuition: f added no information/constraints (it was already known).



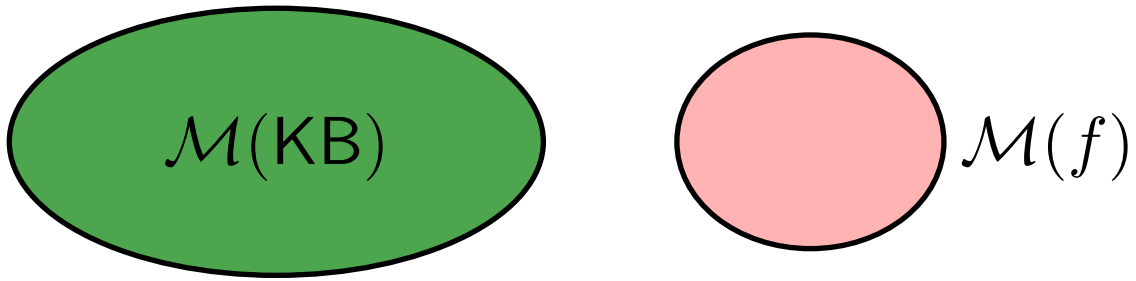
Definition: entailment

KB entails f (written KB $\models f$) iff $\mathcal{M}(\text{KB}) \subseteq \mathcal{M}(f)$.

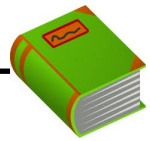
Example: Rain $\wedge$ Snow $\models$ Snow

- The first case is if the set of models of f is a superset of the models of KB, then f adds no information. We say that KB **entails** f.

Contradiction



Intuition: f contradicts what we know (captured in KB).



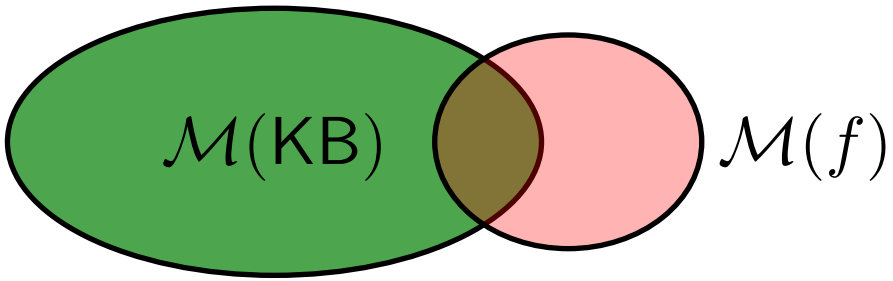
Definition: contradiction

KB contradicts f iff $\mathcal{M}(\text{KB}) \cap \mathcal{M}(f) = \emptyset$.

Example: Rain $\wedge$ Snow contradicts $\neg$Snow

- The second case is if the set of models defined by f is completely disjoint from those of KB. Then we say that the KB and f **contradict** each other. If we believe KB, then we cannot possibly believe f.

Contingency



Intuition: f adds non-trivial information to KB

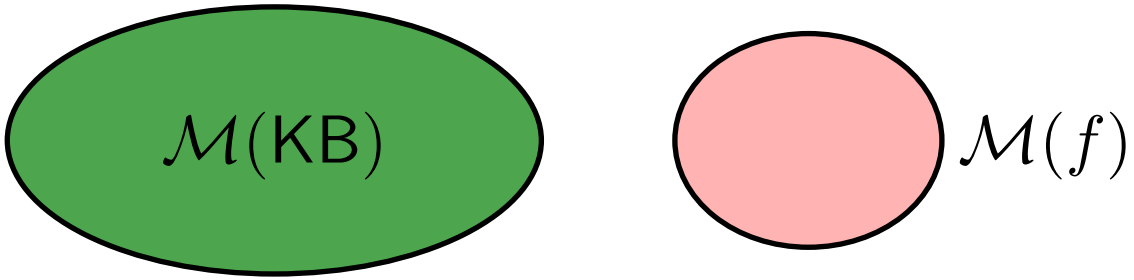
$$\emptyset \subsetneq \mathcal{M}(\text{KB}) \cap \mathcal{M}(f) \subsetneq \mathcal{M}(\text{KB})$$

Example: Rain and Snow

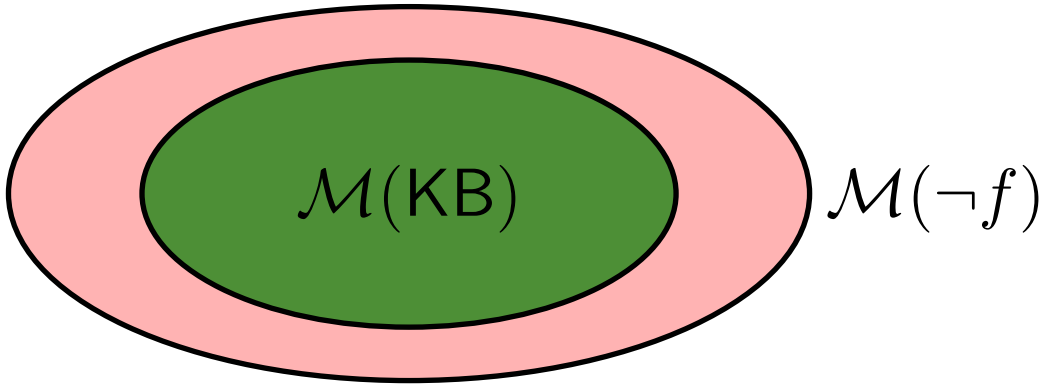
- In the third case, we have a non-trivial overlap between the models of KB and f. We say in this case that f is **contingent**; f could be satisfied or not satisfied depending on the model.

Contradiction and entailment

Contradiction:



Entailment:

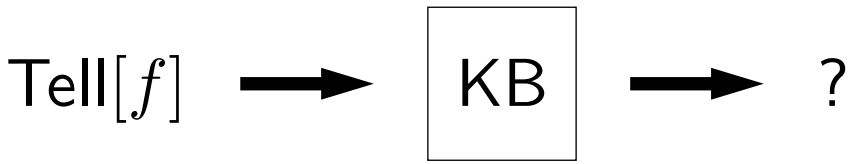


Proposition: contradiction and entailment

KB **contradicts** f iff KB **entails** $\neg f$.

- There is a useful connection between entailment and contradiction. If f is contradictory, then its negation $(\neg f)$ is entailed, and vice-versa.
- You can see this because the models $\mathcal{M}(f)$ and $\mathcal{M}(\neg f)$ partition the space of models.

Tell operation



Tell: *It is raining.*

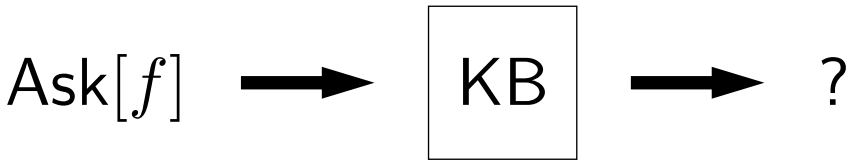
Tell[Rain]

Possible responses:

- Already knew that: entailment (KB $\models f$)
- Don't believe that: contradiction (KB $\models \neg f$)
- Learned something new (update KB): contingent

- Having defined the three possible relationships that a new formula f can have with respect to a knowledge base KB, let's try to determine the appropriate response that a system should have.
- Suppose we tell the system that it is raining ($f = \text{Rain}$). If f is entailed, then we should reply that we already knew that. If f contradicts the knowledge base, then we should reply that we don't believe that. If f is contingent, then this is the interesting case, where we have non-trivially restricted the set of models, so we reply that we've learned something new.

Ask operation



Ask: *Is it raining?*

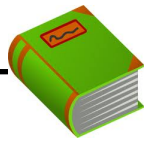
Ask[Rain]

Possible responses:

- Yes: entailment (KB $\models f$)
- No: contradiction (KB $\models \neg f$)
- I don't know: contingent

- Suppose now that we ask the system a question: is it raining? If f is entailed, then we should reply with a definitive yes. If f contradicts the knowledge base, then we should reply with a definitive no. If f is contingent, then we should just confess that we don't know.

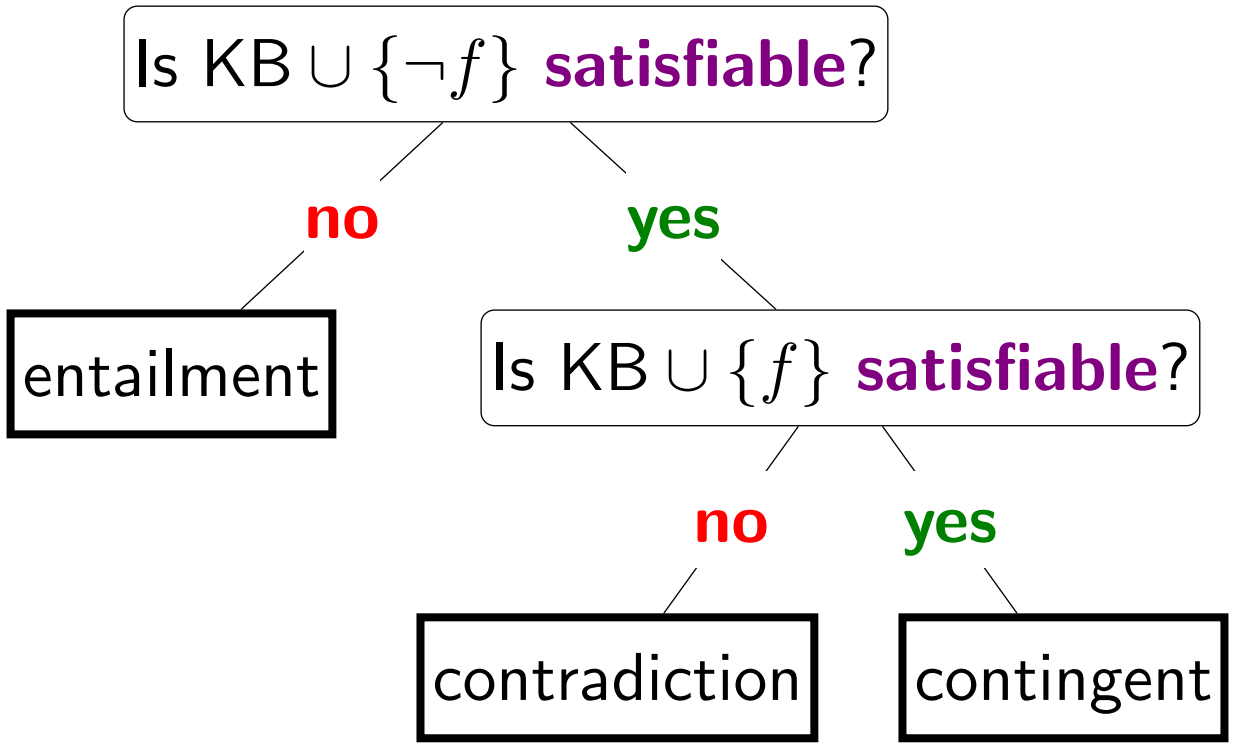
Satisfiability



Definition: satisfiability

A knowledge base KB is **satisfiable** if $\mathcal{M}(\text{KB}) \neq \emptyset$.

Reduce Ask$[f]$ and Tell$[f]$ to satisfiability:



- Now let's return to pure logic land again. How can we go about actually checking entailment, contradiction, and contingency? One useful concept to rule them all is **satisfiability**.
- Recall that we said a particular model w satisfies f if the interpretation function returns true $\mathcal{I}(f, w) = 1$. We can say that a formula f by itself is satisfiable if there is some model that satisfies f. Finally, a knowledge base (which is no more than just the conjunction of its formulas) is satisfiable if there is some model that satisfies all the formulas $f \in \text{KB}$.
- With this definition in hand, we can implement Ask$[f]$ and Tell$[f]$ as follows:
- First, we check if $\text{KB} \cup \{\neg f\}$ is satisfiable. If the answer is no, that means the models of $\neg f$ and KB don't intersect (in other words, the two contradict each other). Recall that this is equivalent to saying that KB entails f.
- Otherwise, we need to do another test: check whether $\text{KB} \cup \{f\}$ is satisfiable. If the answer is no here, then KB and f are contradictory. Otherwise, we have that both f and $\neg f$ are compatible with KB, so the result is contingent.

Model checking

Checking satisfiability (SAT) in propositional logic is special case of solving CSPs!

Mapping:

propositional symbol	$\Rightarrow$	variable
formula	$\Rightarrow$	constraint
model	$\Leftarrow$	assignment

- Now we have reduced the problem of working with knowledge bases to checking satisfiability. The bad news is that this is an (actually, the canonical) NP-complete problem, so there are no efficient algorithms in general.
- The good news is that people try to solve the problem anyway, and we actually have pretty good SAT solvers these days. In terms of this class, this problem is just a CSP, if we convert the terminology: Each propositional symbol becomes a variable and each formula is a constraint. We can then solve the CSP, which produces an assignment, or in logic-speak, a model.

Model checking

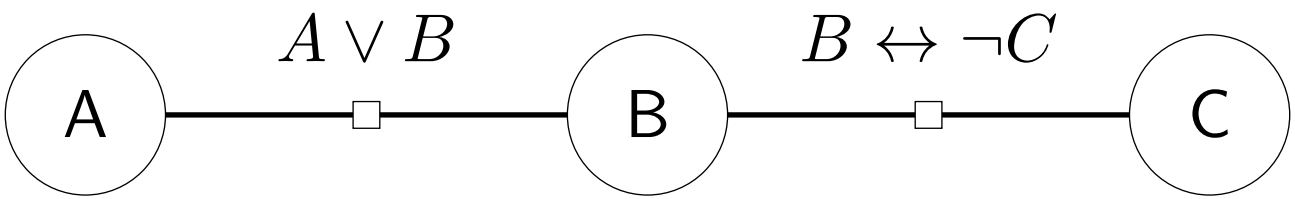
Example: model checking

$\text{KB} = \{A \vee B, B \leftrightarrow \neg C\}$

Propositional symbols (CSP variables):

$$\{A, B, C\}$$

CSP:

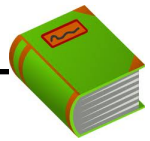


Consistent assignment (satisfying model):

$$\{A : 1, B : 0, C : 1\}$$

- As an example, consider a knowledge base that has two formulas and three variables. Then the CSP is shown. Solving the CSP produces a consistent assignment (if one exists), which is a model that satisfies KB.
- Note that in the knowledge base tell/ask application, we don't technically need the satisfying assignment. An assignment would only offer a counterexample certifying that the answer **isn't** entailment or contradiction. This is an important point: entailment and contradiction is a claim about all models, not about the existence of a model.

Model checking



Definition: model checking

Input: knowledge base KB

Output: exists satisfying model ($\mathcal{M}(\text{KB}) \neq \emptyset$)?

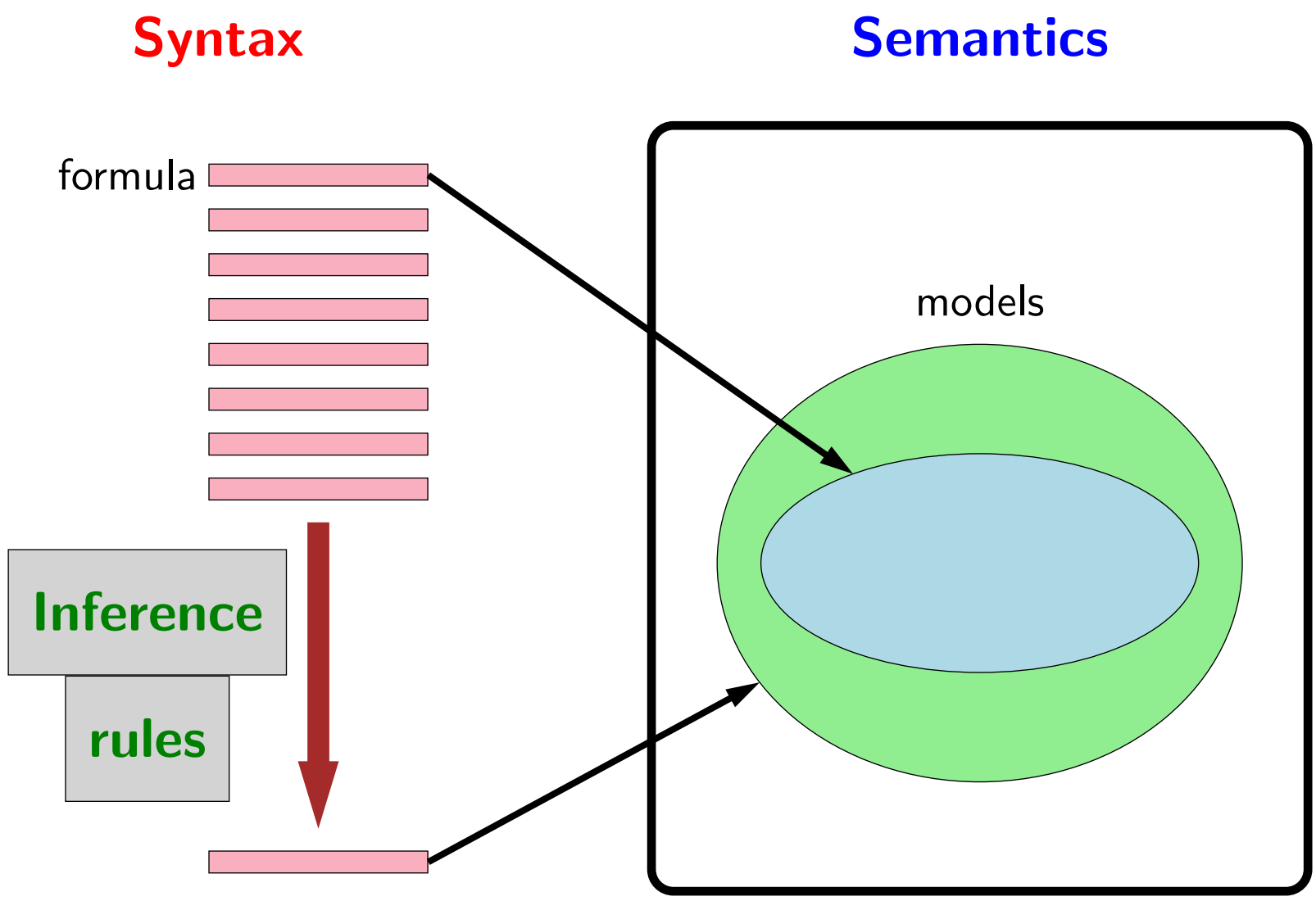
Popular algorithms:

- DPLL (backtracking search + pruning)
- WalkSat (randomized local search)

Next: Can we exploit the fact that factors are formulas?

- Checking satisfiability of a knowledge base is called **model checking**. For propositional logic, there are several algorithms that work quite well which are based on the algorithms we saw for solving CSPs (backtracking search and local search).
- However, can we do a bit better? Our CSP factors are not arbitrary — they are logic formulas, and recall that formulas are defined recursively and have some compositional structure. Let's see how to exploit this.

Propositional logic



- So far, we have used formulas, via semantics, to define sets of models. And all our reasoning on formulas has been through these models (e.g., reduction to satisfiability). Inference rules allow us to do reasoning on the formulas themselves without ever instantiating the models.
- This can be quite powerful. If you have a huge KB with lots of formulas and propositional symbols, sometimes you can draw a conclusion without instantiating the full model checking problem. This will be very important when we move to first-order logic, where the models can be infinite, and so model checking would be infeasible.

Inference rules

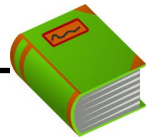
Example of making an inference:

It is raining. (Rain)

If it is raining, then it is wet. (Rain $\rightarrow$ Wet)

Therefore, it is wet. (Wet)

$$\frac{\text{Rain}, \quad \text{Rain} \rightarrow \text{Wet}}{\text{Wet}} \qquad \frac{\text{(premises)}}{\text{(conclusion)}}$$



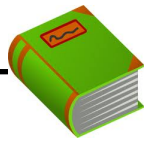
Definition: Modus ponens inference rule

For any propositional symbols p and q:

$$\frac{p, \quad p \rightarrow q}{q}$$

- The idea of making an inference should be quite intuitive to you. The classic example is **modus ponens**, which captures the if-then reasoning pattern.

Inference framework



Definition: inference rule

If $f_1, \ldots, f_k, g$ are formulas, then the following is an **inference rule**:

$$\frac{f_1, \quad \ldots \quad , f_k}{g}$$

Key idea: inference rules

Rules operate directly on **syntax**, not on **semantics**.

- In general, an inference rule has a set of premises and a conclusion. The rule says that if the premises are in the KB, then you can add the conclusion to the KB.
- We haven't yet specified whether this is a valid thing to do, but it is a thing to do. Remember, syntax is just about symbol pushing; it is only by linking to models that we have notions of truth and meaning (semantics).

Sound rules of inference

Examples of sound rules of inference

Each can be shown to be sound using a truth table

RULE	PREMISE	CONCLUSION
Modus Ponens	$A, A \rightarrow B$	B
And Introduction	A, B	$A \wedge B$
And Elimination	$A \wedge B$	A
Double Negation	$\neg\neg A$	A
Unit Resolution	$A \vee B, \neg B$	A
Resolution	$\mathbf{A \vee B, \neg B \vee C}$	$\mathbf{A \vee C}$

Inference algorithm

Algorithm: forward inference

Input: set of inference rules Rules.

Repeat until no changes to KB:

  Choose set of formulas $f_1, \ldots, f_k \in$ KB.

  If matching rule $\frac{f_1, \quad \ldots \quad , f_k}{g}$ exists:

    Add g to KB.

Definition: derivation

KB **derives/proves** f (KB $\vdash f$) iff f eventually gets added to KB.

- Given a set of inference rules (e.g., modus ponens), we can just keep on trying to apply rules. Those rules generate new formulas which get added to the knowledge base, and those formulas might then be premises of other rules, which in turn generate more formulas, etc.
- We say that the KB derives or proves a formula f if by blindly applying rules, we can eventually add f to the KB.

Inference example

Example: Modus ponens inference

Starting point:

$$\text{KB} = \{\text{Rain}, \text{Rain} \rightarrow \text{Wet}, \text{Wet} \rightarrow \text{Slippery}\}$$

Apply modus ponens to Rain and Rain $\rightarrow$ Wet:

$$\text{KB} = \{\text{Rain}, \text{Rain} \rightarrow \text{Wet}, \text{Wet} \rightarrow \text{Slippery}, \text{Wet}\}$$

Apply modus ponens to Wet and Wet $\rightarrow$ Slippery:

$$\text{KB} = \{\text{Rain}, \text{Rain} \rightarrow \text{Wet}, \text{Wet} \rightarrow \text{Slippery}, \text{Wet}, \text{Slippery}\}$$

Converged.

Can't derive some formulas: $\neg\text{Wet}$, $\text{Rain} \rightarrow \text{Slippery}$

- Here is an example where we've applied modus ponens twice. Note that Wet and Slippery are derived by the KB.
- But there are some formulas which cannot be derived. Some of these underivable formulas will look bad anyway ($\neg$Wet), but others will seem reasonable (Rain $\rightarrow$ Slippery).

Resolution

- **Resolution** is a valid inference rule producing a new clause implied by two clauses containing *complementary literals*

  Literal: atomic symbol or its negation, i.e., P, ~P

- Amazingly, this is the **only** interference rule needed to build a sound & complete theorem prover
  - Based on proof by contradiction, usually called resolution refutation
- The resolution rule was discovered by Alan Robinson (CS, U. of Syracuse) in the mid 1960s

Resolution

- A KB is a set of sentences all of which are true, i.e., a conjunction of sentences
- To use resolution, put KB into *conjunctive normal form* (CNF)
  - Each sentence is a disjunction of one or more literals (positive or negative atoms)
- Every KB can be put into CNF, it's just a matter of rewriting its sentences using standard tautologies, e.g.: $P \rightarrow Q \equiv \sim P \vee Q$

CNF (Conjunctive Normal Form)

All of the following formulas in the variables A, B, C, D, E, and F are in conjunctive normal form:

- $(A \vee \neg B \vee \neg C) \wedge (\neg D \vee E \vee F)$
- $(A \vee B) \wedge (C)$
- $(A \vee B)$
- (A)

Each sentence is a disjunction of one or more literals (positive or negative atoms)

The following formulas are **not** in conjunctive normal form:

- $\neg(B \vee C)$, since an OR is nested within a NOT
- $(A \wedge B) \vee C$
- $A \wedge (B \vee (D \wedge E))$, since an AND is nested within an OR

Every formula can be equivalently written as a formula in conjunctive normal form. The three non-examples in CNF are:

- $(\neg B) \wedge (\neg C)$
- $(A \vee C) \wedge (B \vee C)$
- $(A) \wedge (B \vee D) \wedge (B \vee E)$.

Resolution Example

Tautologies

$(A \rightarrow B) \leftrightarrow (\sim A \vee B)$

$(A \vee (B \wedge C)) \leftrightarrow (A \vee B) \wedge (A \vee C)$

- KB: $[P \rightarrow Q , Q \rightarrow R \wedge S]$
- KB: $[P \rightarrow Q , Q \rightarrow R, Q \rightarrow S]$
- KB in CNF: $[\sim P \vee Q , \sim Q \vee R , \sim Q \vee S]$
- Resolve KB[0] and KB[1] producing:

  $\sim P \vee R$ *(i.e., $P \rightarrow R$)*
- Resolve KB[0] and KB[2] producing:

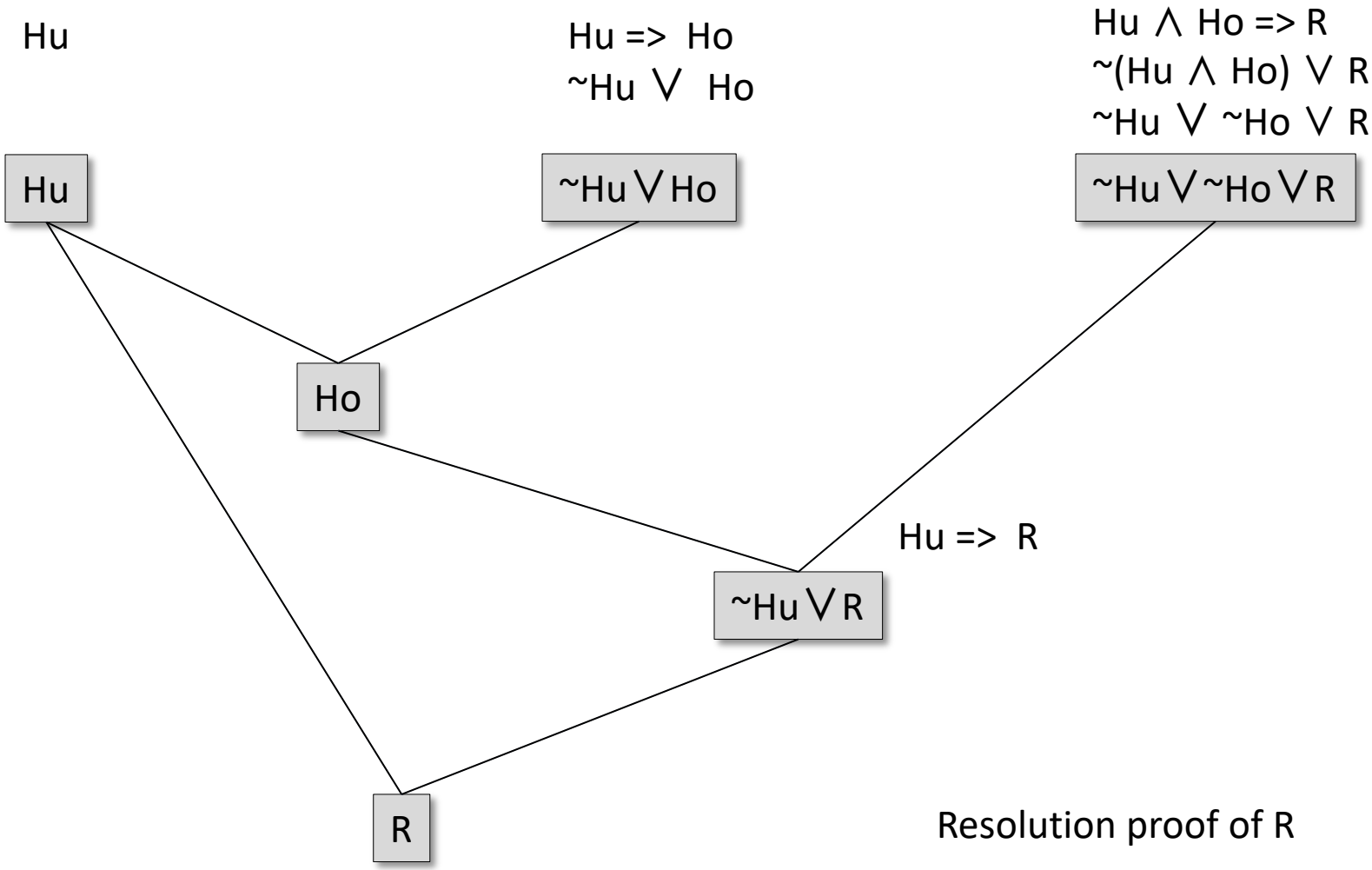
  $\sim P \vee S$ *(i.e., $P \rightarrow S$)*
- New KB: $[\sim P \vee Q , \sim Q \vee R, \sim Q \vee S, \sim P \vee R, \sim P \vee S]$

Proving it's raining with rules

- A **proof** is a sequence of sentences, where each is a premise (i.e., a given) or is derived from earlier sentences in the proof by an inference rule
- Last sentence is the **theorem** (also called goal or query) that we want to prove
- The *weather problem* using traditional reasoning

1 Hu	premise	"It's humid"
2 Hu$\rightarrow$Ho	premise	"If it's humid, it's hot"
3 Ho	modus ponens(1,2)	"It's hot"
4 (Ho$\wedge$Hu)$\rightarrow$R	premise	"If it's hot & humid, it's raining"
5 Ho$\wedge$Hu	and introduction(1,3)	"It's hot and humid"
6 R	modus ponens(4,5)	"It's raining"

Proving it's raining with resolution



A simple proof procedure

This procedure generates new sentences in a KB

1. Convert all sentences in the KB to CNF[1]
2. Find all pairs of sentences in KB with complementary literals[2] that have not yet been resolved
3. If there are no pairs stop else resolve each pair, adding the result to the KB and go to 2

- Is it sound?
- Is it complete?
- Will it always terminate?

1: a KB in conjunctive normal form is a set of disjunctive sentences

2: a literal is a variable or its negation

Propositional Resolution

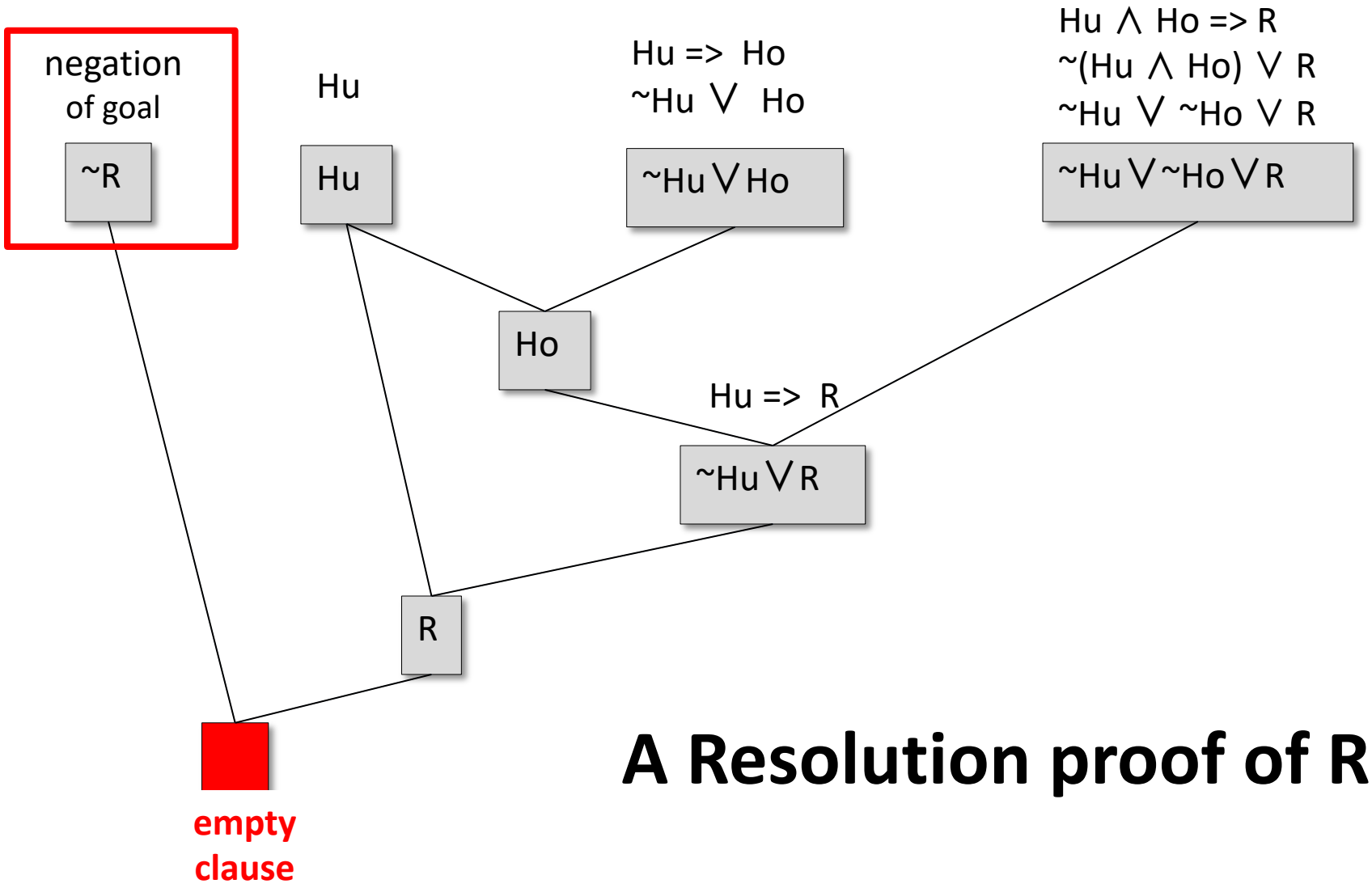
- It is sound!
- It's not *generatively complete* in that it can't derive all clauses that follow from the KB
  - The issues are not serious limitations, though
  - Example: if the KB includes P and includes Q we won't derive P ^ Q
- It will always terminate
- But generating all clauses that follow can take a long time and many may be useless

Resolution refutation

1. Add negation of goal to the KB
2. Convert all sentences in KB to CNF
3. Find all pairs of sentences in KB with complementary literals that have not yet been resolved
4. If there are no pairs stop else resolve each pair, adding the result to the KB and go to 2

- If we derived an empty clause (i.e., a contradiction) then the conclusion follows from the KB
- If we did not, the conclusion cannot be proved from the KB

Proving it's raining with refutation resolution



A Resolution proof of R

Propositional logic: pro and con

- **Advantages**
  - Simple KR language good for many problems
  - Lays foundation for higher logics (e.g., FOL)
  - Reasoning is decidable, though NP complete; efficient techniques exist for many problems
- **Disadvantages**
  - Not expressive enough for most problems
  - Even when it is, it can very "un-concise"

PL is a weak KR language

- Hard to identify *individuals* (e.g., Mary, 3)
- Can't directly represent properties of individuals or relations between them (e.g., "Bill age 24")
- Generalizations, patterns, regularities hard to represent (e.g., "all triangles have 3 sides")
- First-Order Logic (FOL) represents this information via **relations**, **variables** & **quantifier**s, e.g.,
  - *John loves Mary: loves(John, Mary)*
  - *Every elephant is gray:* $\forall$ x (elephant(x) $\rightarrow$ gray(x))
  - *There is a black swan:* $\exists$ x (swan(X) ^ black(X))

Hunt the Wumpus

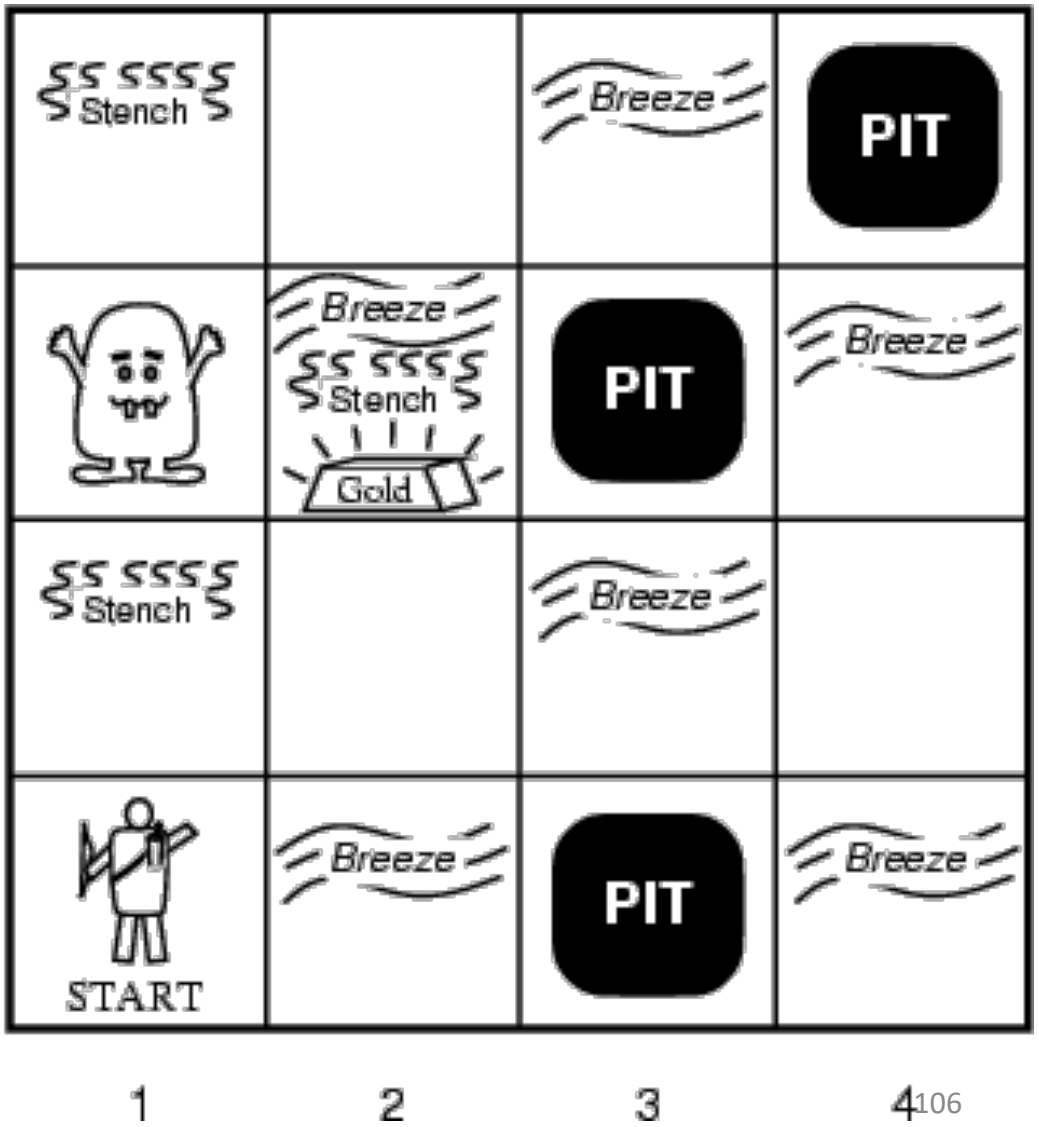
Wumpus World environment

- Based on Hunt the Wumpus computer game
- Agent explores cave of rooms connected by passageways
- Lurking in a room is the *Wumpus*, a beast that eats any agent that enters its room
- Some rooms have *bottomless pits* that trap any agent that wanders into the room
- Somewhere is a heap of gold in a room
- Goal: collect gold & exit w/o being eaten

AIMA's Wumpus World

The agent always starts in the field [1,1]

Agent's task is to find the gold, return to the field [1,1] and climb out of the cave



Agent in a Wumpus world: Percepts

- The agent perceives
  - **stench** in square containing Wumpus and in adjacent squares (not diagonally)
  - **breeze** in squares adjacent to a pit
  - **glitter** in the square where the gold is
  - **bump**, if it walks into a wall
  - Woeful **scream** everywhere in cave, if Wumpus killed
- Percepts given as five-tuple, e.g., if stench and breeze, but no glitter, bump or scream:

  [Stench, Breeze, None, None, None]
- Agent cannot perceive its location, e.g., (2,2)

Wumpus World Actions

- **go forward**
- **turn right** 90 degrees
- **turn left** 90 degrees
- **grab**: Pick up object in same square as agent
- **shoot**: Fire arrow in direction agent faces. It continues until it hits & kills Wumpus or hits outer wall. Agent has one arrow, so only first shoot action has effect
- **Climb:** leave cave, only effective in start square
- **die**: automatically and irretrievably happens if agent enters square with pit or living Wumpus

Wumpus World Goal

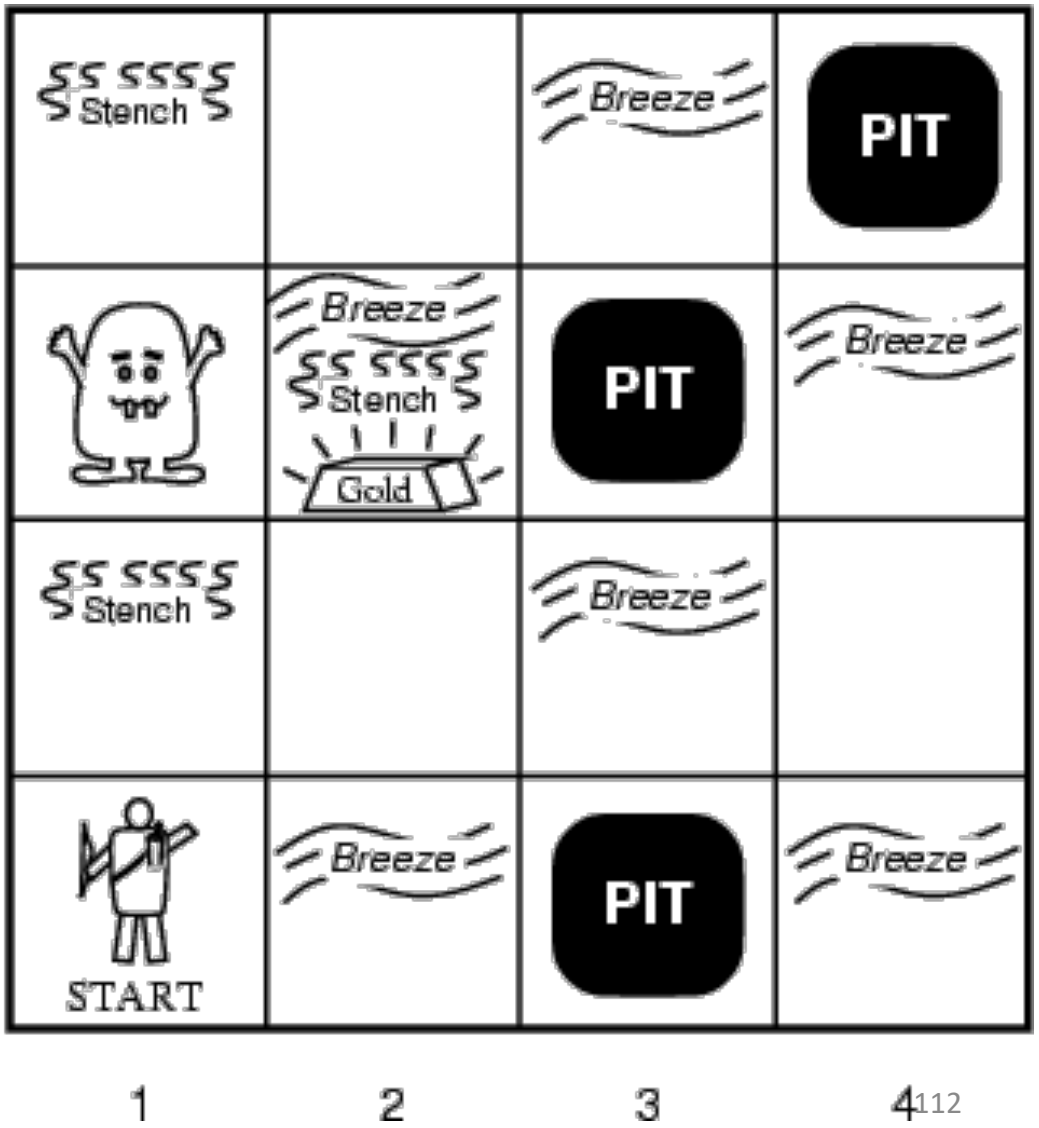
Agent's goal is to find the gold and bring it back to the start square as quickly as possible, without getting killed

- 1,000 point reward for climbing out of cave with gold
- 1 point deducted for every action taken
- 10,000 point penalty for getting killed

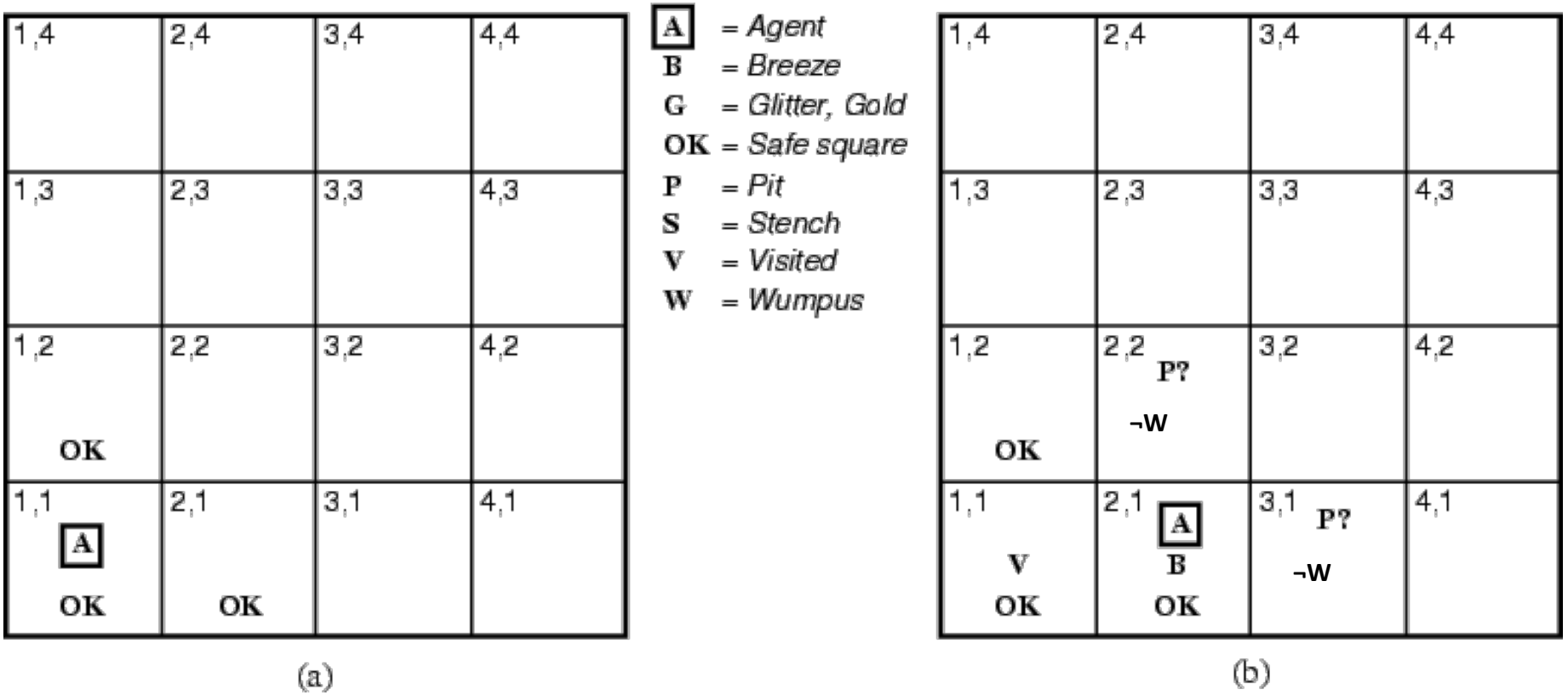
AIMA's Wumpus World

The agent always starts in the field [1,1]

Agent's task is to find the gold, return to the field [1,1] and climb out of the cave



The Hunter's first step

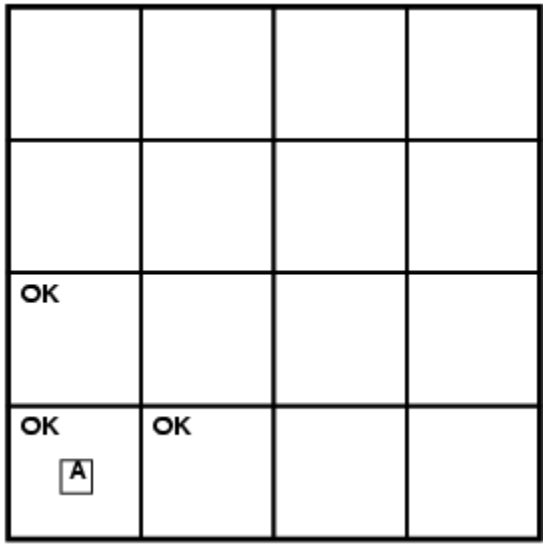


(a) (b)

Since agent is alive and perceives neither breeze nor stench at [1,1], it **knows** [1,1] and its neighbors are OK

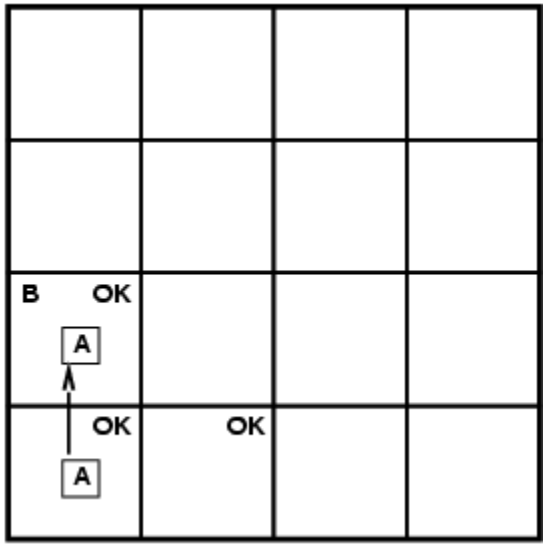
Moving to [2,1] is a **safe move** that reveals a breeze but no stench, **implying** that Wumpus isn't adjacent but one or more pits are

Exploring a wumpus world



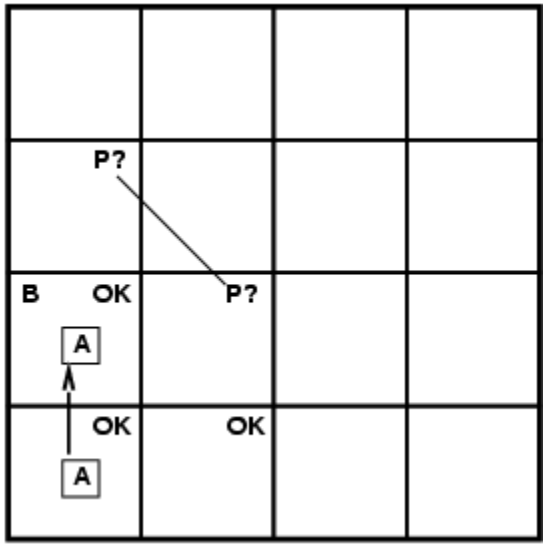
A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

Exploring a wumpus world



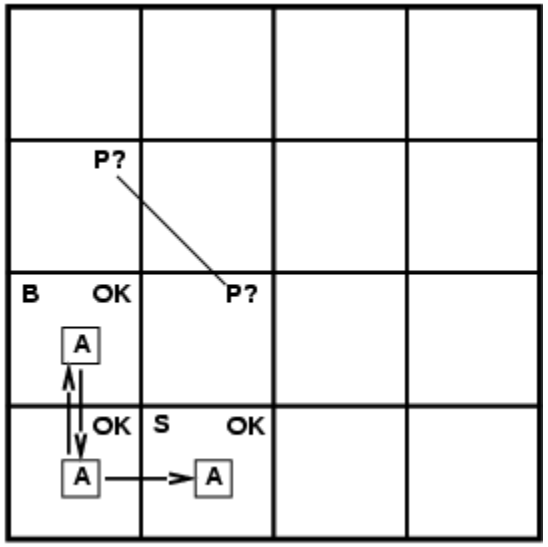
A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

Exploring a wumpus world



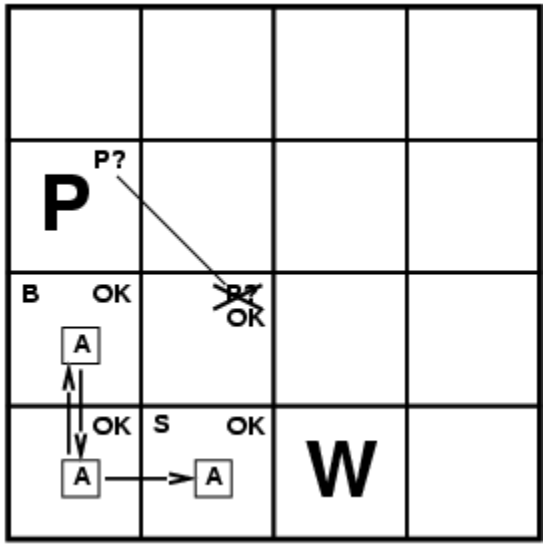
A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

Exploring a wumpus world



A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

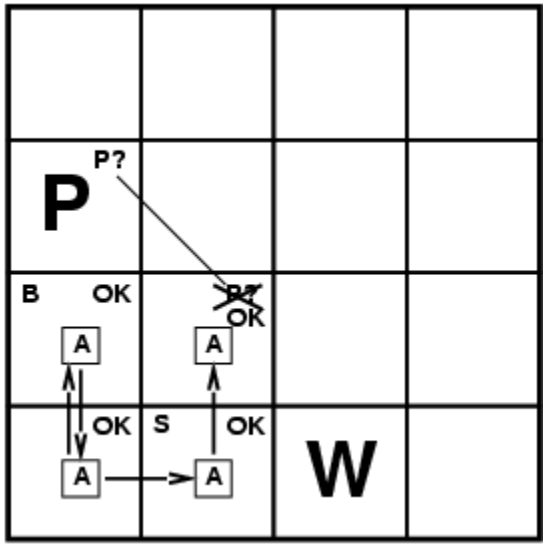
Exploring a wumpus world



A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

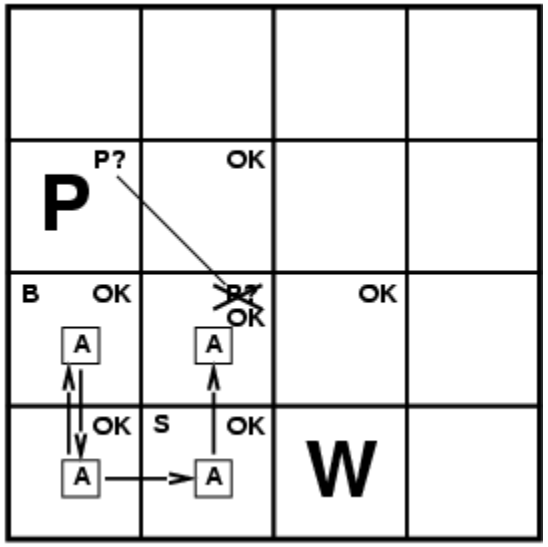
No stench in (1,2) => Wumpus not in (2,2)
No breeze in (2,1) => no pit in (2,2) => pit in (1,3)

Exploring a wumpus world



A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

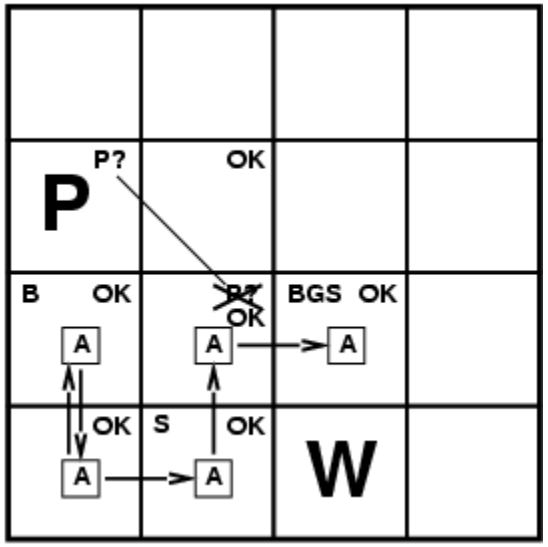
Exploring a wumpus world



A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

Going to (2,2) is the only “safe” move

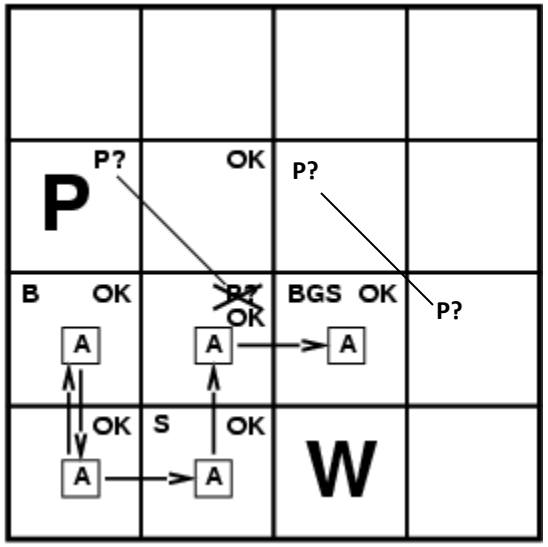
Exploring a wumpus world



A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

Going to (2,3) is a “safe” move

Exploring a wumpus world



A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

Found gold! Now find way back to (1,1)

Reasoning in Hunt the Wumpus

Hunt the Wumpus domain

- Some atomic propositions:
  - A12 = agent is in call (1,2)
  - S12 = There's a stench in cell (1,2)
  - B34 = There's a breeze in cell (3,4)
  - W22 = Wumpus is in cell (2,2)
  - V11 = We've visited cell (1,1)
  - OK11 = cell (1,1) is safe
  - …
- Some rules:
  - $\neg S22 \rightarrow \neg W12 \wedge \neg W23 \wedge \neg W32 \wedge \neg W21$
  - $S22 \rightarrow W12 \vee W23 \vee W32 \vee W21$
  - $B22 \rightarrow P12 \vee P23 \vee P32 \vee P21$
  - $W22 \rightarrow S12 \wedge S23 \wedge S32 \wedge W21$
  - $W22 \rightarrow \neg W11 \wedge \neg W21 \wedge \ldots \neg W44$
  - $A22 \rightarrow V22$
  - $A22 \rightarrow \neg W11 \wedge \neg W21 \wedge \ldots \neg W44$
  - $V22 \rightarrow OK22$

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

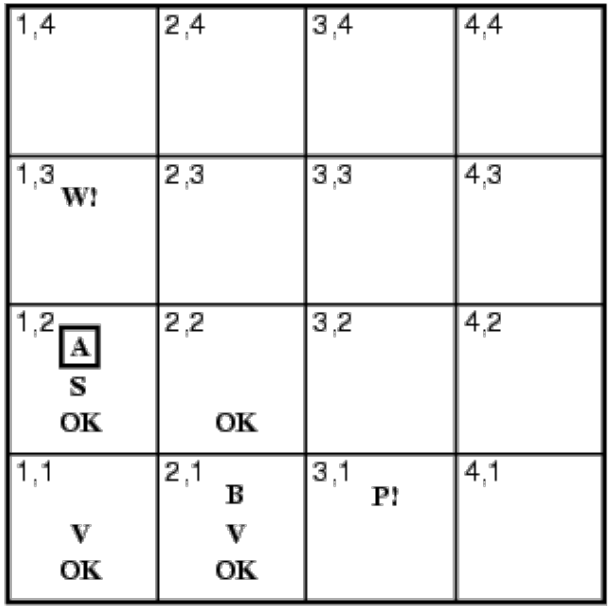
A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

If there's no stench in cell 2,2 then the Wumpus isn't in cell 21, 23 32 or 21

Hunt the Wumpus domain

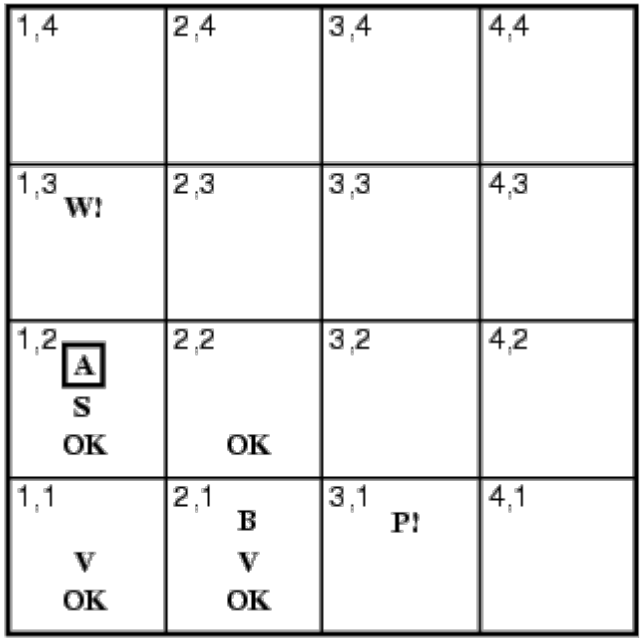
- Eight symbols for each cell, i.e.: A11, B11, G11, OK11, P11, S11, V11, W11
- Lack of variables requires giving similar rules for each cell!
- Ten rules (I think) for each

$A11 \rightarrow \ldots$
$V11 \rightarrow \ldots$
$P11 \rightarrow \ldots$
$\neg P11 \rightarrow \ldots$
$W11 \rightarrow \ldots$
$\neg W11 \rightarrow \ldots$
$S11 \rightarrow \ldots$
$\neg S11 \rightarrow \ldots$
$B11 \rightarrow \ldots$
$\neg B11 \rightarrow \ldots$



- 8 symbols for 16 cells => 128 symbols
- 2^{128} possible models ☹
- Must do better than brute force

After third move



A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

- We can prove that the Wumpus is in (1,3) using these four rules
- See RN section 7.5

(R1) $\neg S11 \rightarrow \neg W11 \wedge \neg W12 \wedge \neg W21$

(R2) $\neg S21 \rightarrow \neg W11 \wedge \neg W21 \wedge \neg W22 \wedge \neg W31$

(R3) $\neg S12 \rightarrow \neg W11 \wedge \neg W12 \wedge \neg W22 \wedge \neg W13$

(R4) $S12 \rightarrow W13 \vee W12 \vee W22 \vee W11$

Proving W13: Wumpus is in cell 1,3

Apply **MP** with $\neg$S11 and R1:

$\neg$ W11 $\wedge$ $\neg$ W12 $\wedge$ $\neg$ W21

Apply **AE**, yielding three sentences:

$\neg$ W11, $\neg$ W12, $\neg$ W21

Apply **MP** to ~S21 and R2, then apply **AE**:

$\neg$ W22, $\neg$ W21, $\neg$ W31

Apply **MP** to S12 and R4 to obtain:

W13 $\vee$ W12 $\vee$ W22 $\vee$ W11

Apply **UR** on (W13 $\vee$ W12 $\vee$ W22 $\vee$ W11) and $\neg$W11:

W13 $\vee$ W12 $\vee$ W22

Apply **UR** with (W13 $\vee$ W12 $\vee$ W22) and $\neg$W22:

W13 $\vee$ W12

Apply **UR** with (W13 $\vee$ W12) and $\neg$W12:

W13

QED

(R1) $\neg$S11 $\rightarrow$ $\neg$W11 $\wedge$ $\neg$ W12 $\wedge$ $\neg$ W21

(R2) $\neg$ S21 $\rightarrow$ $\neg$W11 $\wedge$ $\neg$ W21 $\wedge$ $\neg$ W22 $\wedge$ $\neg$ W31

(R3) $\neg$ S12 $\rightarrow$ $\neg$W11 $\wedge$ $\neg$ W12 $\wedge$ $\neg$ W22 $\wedge$ $\neg$ W13

(R4) S12 $\rightarrow$ W13 $\vee$ W12 $\vee$ W22 $\vee$ W11

Rule Abbreviation

MP: modes ponens

AE: and elimination

R: unit resolution

Propositional Wumpus problems

- Lack of variables prevents general rules, e.g.:
  - $\forall$ x, y V(x,y) $\rightarrow$ OK(x,y)
  - $\forall$ x, y S(x,y) $\rightarrow$ W(x-1,y) $\vee$ W(x+1,y) …
- Change of KB over time difficult to represent
  - In classical logic; a fact is true or false for all time
  - A standard technique is to index dynamic facts with the time when they're true
    - A(1, 1, 0) *# agent was in cell 1,1 at time 0*
    - A(2, 1, 1) *# agent was in cell 2,1 at time 1*
  - Thus we have a separate KB for every time point