CMSC 471
Propositional and First-Order Logic

KMA Solaiman
ksolaima@umbc.edu

Some slides courtesy Tim Finin

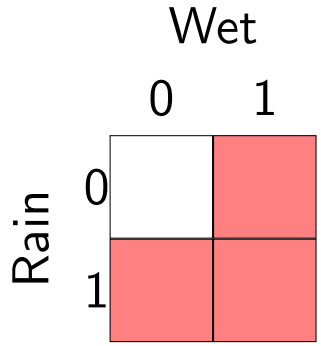
Models: example

Formula:

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

Models:

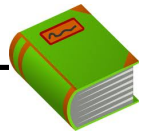
$\mathcal{M}(f) =$



Key idea: compact representation

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

Knowledge base



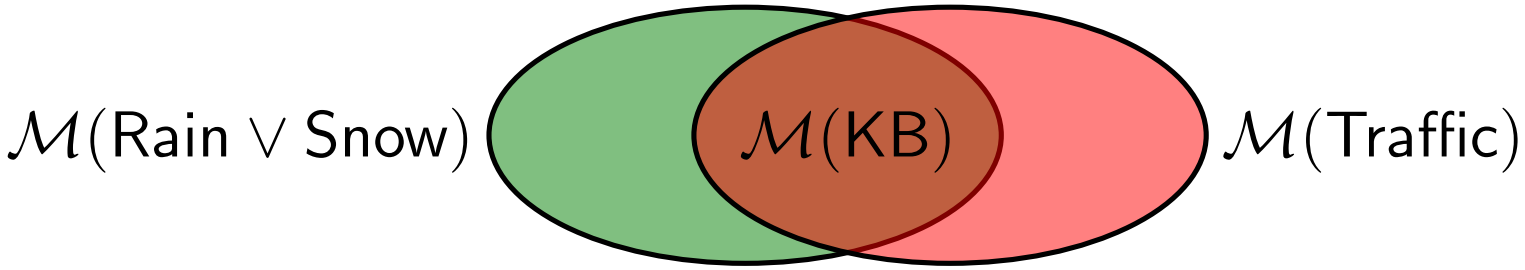
Definition: Knowledge base

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

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

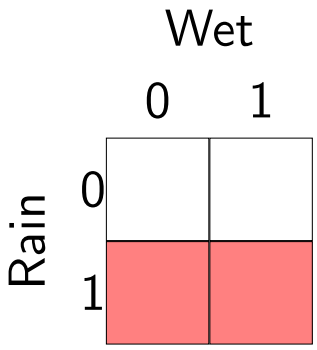
Intuition: KB specifies constraints on the world. $\mathcal{M}(\text{KB})$ is the set of all worlds satisfying those constraints.

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

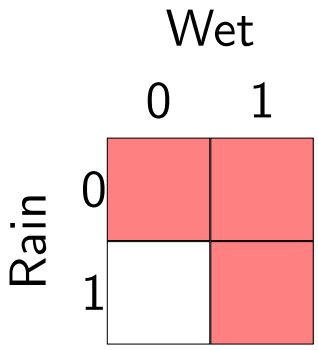


Knowledge base: example

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

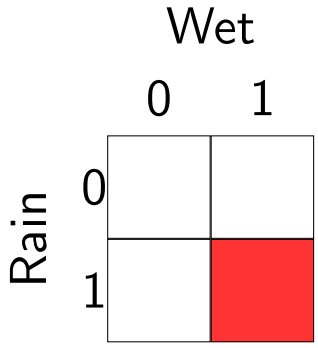


$\mathcal{M}(\text{Rain} \rightarrow \text{Wet})$



Intersection:

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



Adding to the knowledge base

Adding more formulas to the knowledge base:

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

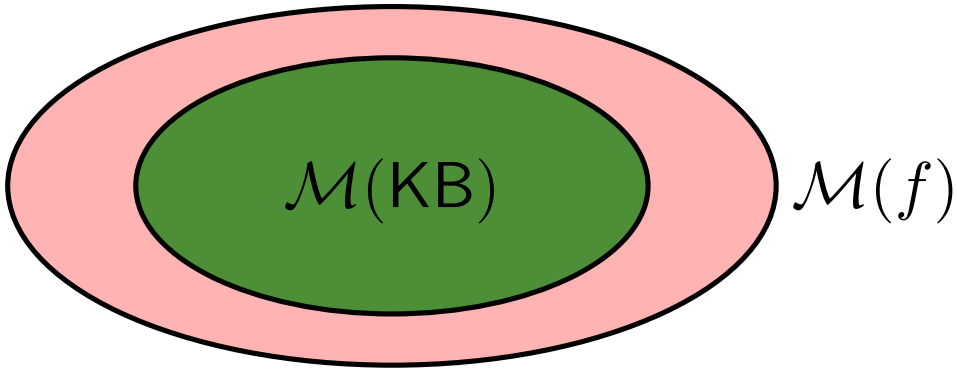
Shrinks the set of models:

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

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

[whiteboard]

Entailment



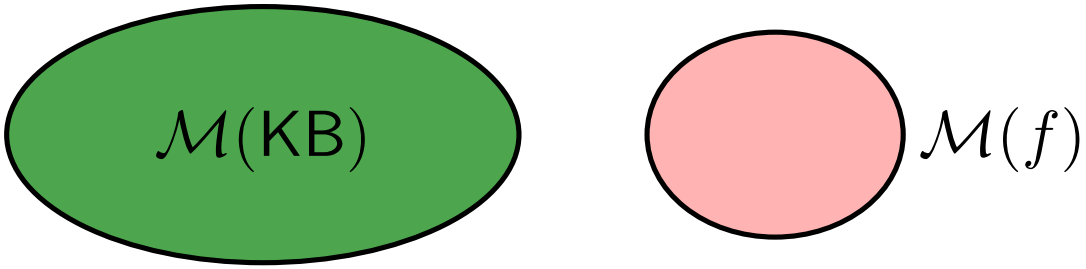
Intuition: f added no information/constraints (it was already known).

Definition: entailment

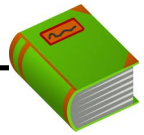
KB entails f (written KB $\models f$) iff $\mathcal{M}(\text{KB}) \subseteq \mathcal{M}(f)$.

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

Contradiction



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

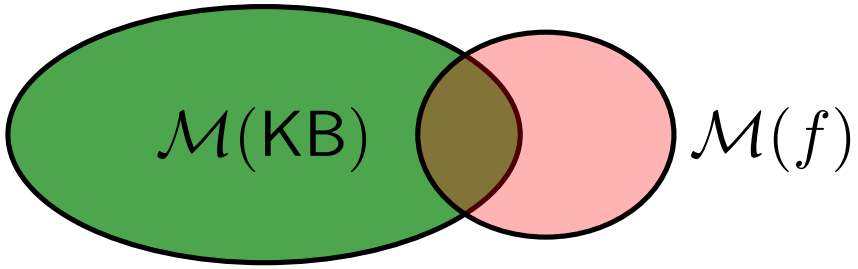


Definition: contradiction

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

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

Contingency

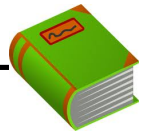


Intuition: f adds non-trivial information to KB

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

Example: Rain and Snow

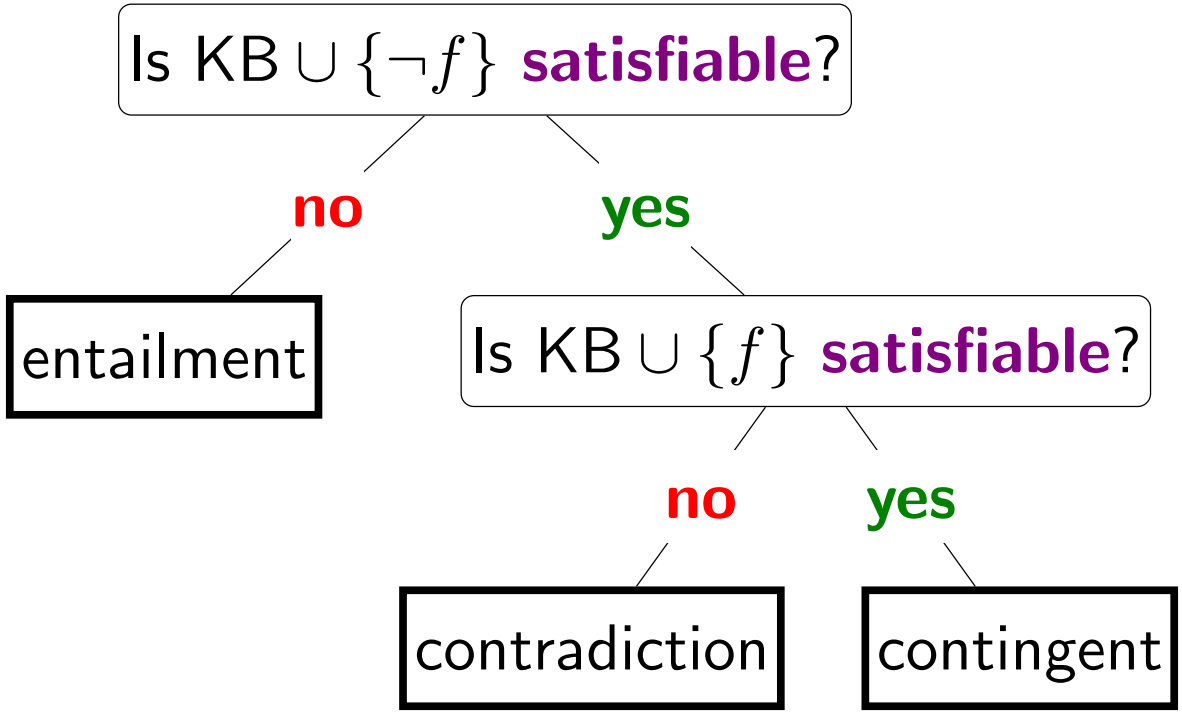
Satisfiability



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

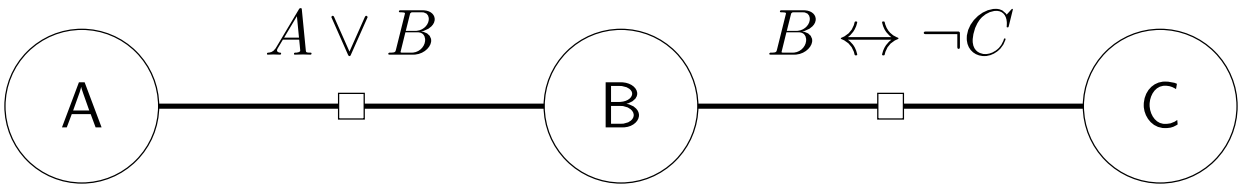
Example: model checking

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

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 ($\mathcal{M}(\text{KB}) \neq \emptyset$)?

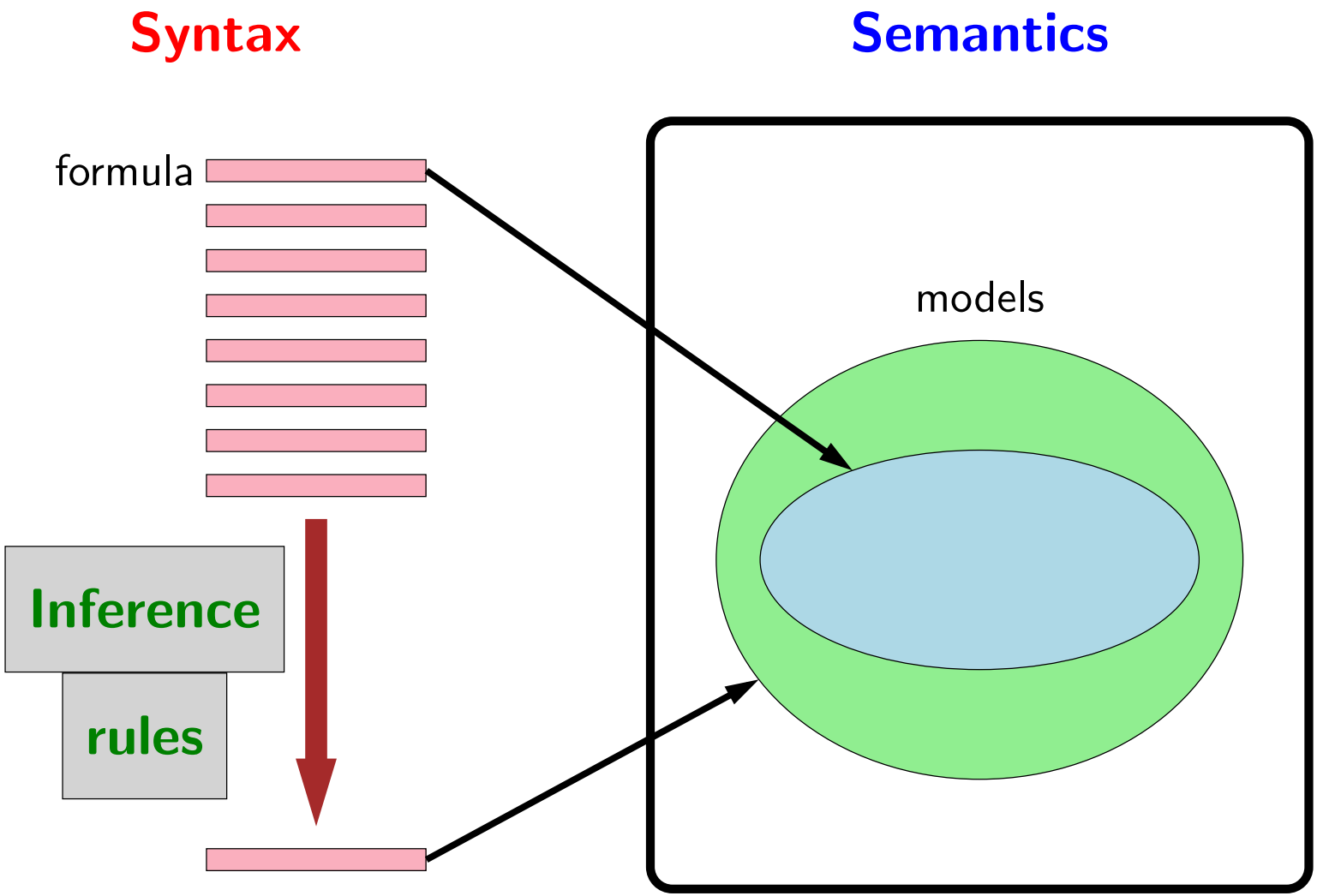
Popular algorithms:

- 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

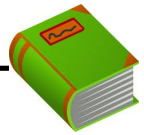
Example of making an inference:

It is raining. (Rain)

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

Therefore, it is wet. (Wet)

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



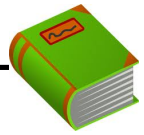
Definition: Modus ponens inference rule

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).

Inference algorithm

Algorithm: forward inference

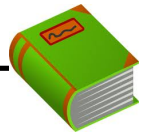
Input: set of inference rules Rules.

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:

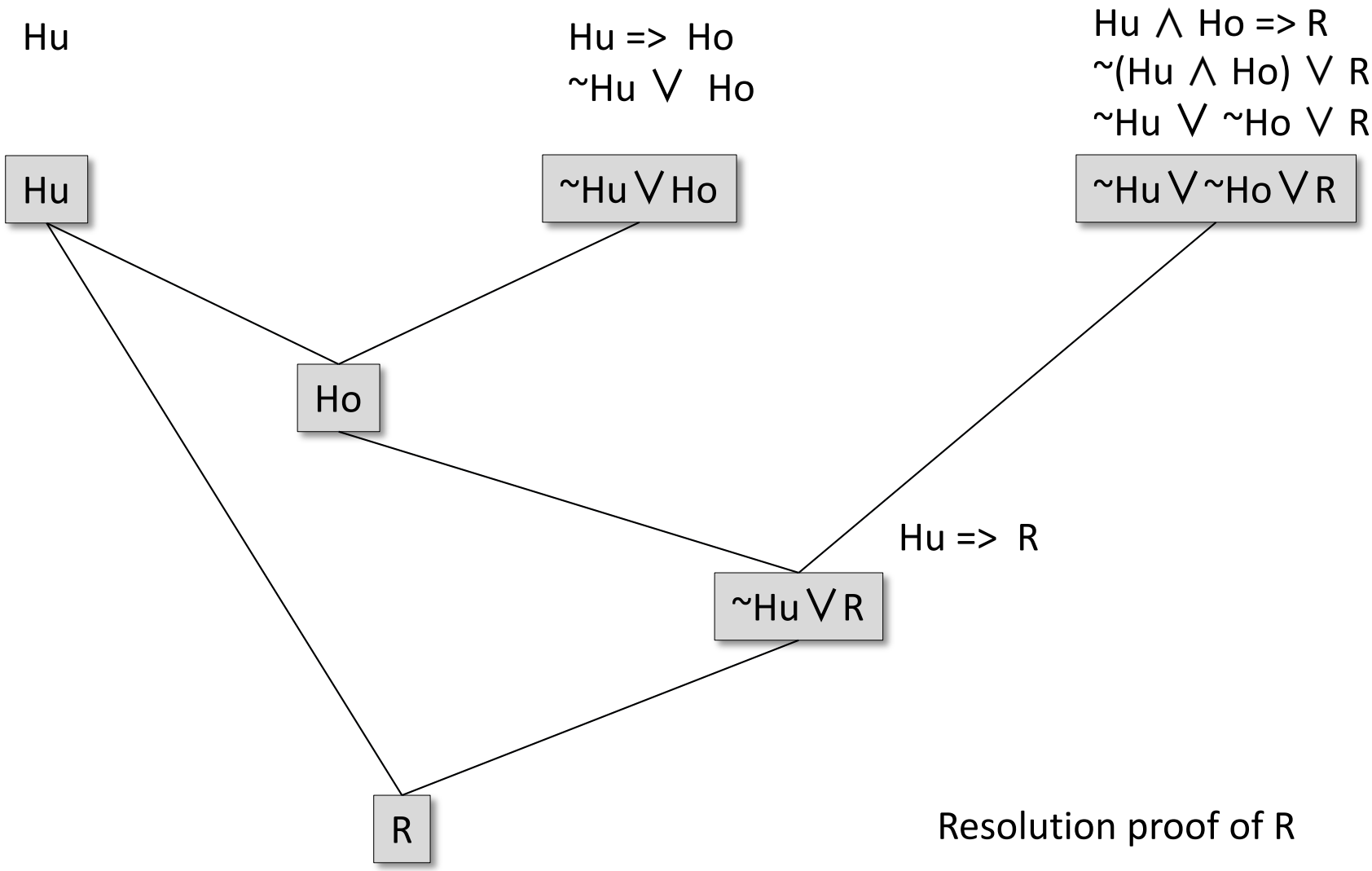
  $\sim P \vee S$ *(i.e., $P \rightarrow S$)*
- New KB: $[\sim P \vee Q , \sim Q \vee R, \sim Q \vee S, \sim P \vee R, \sim P \vee S]$

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



Resolution proof of R

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

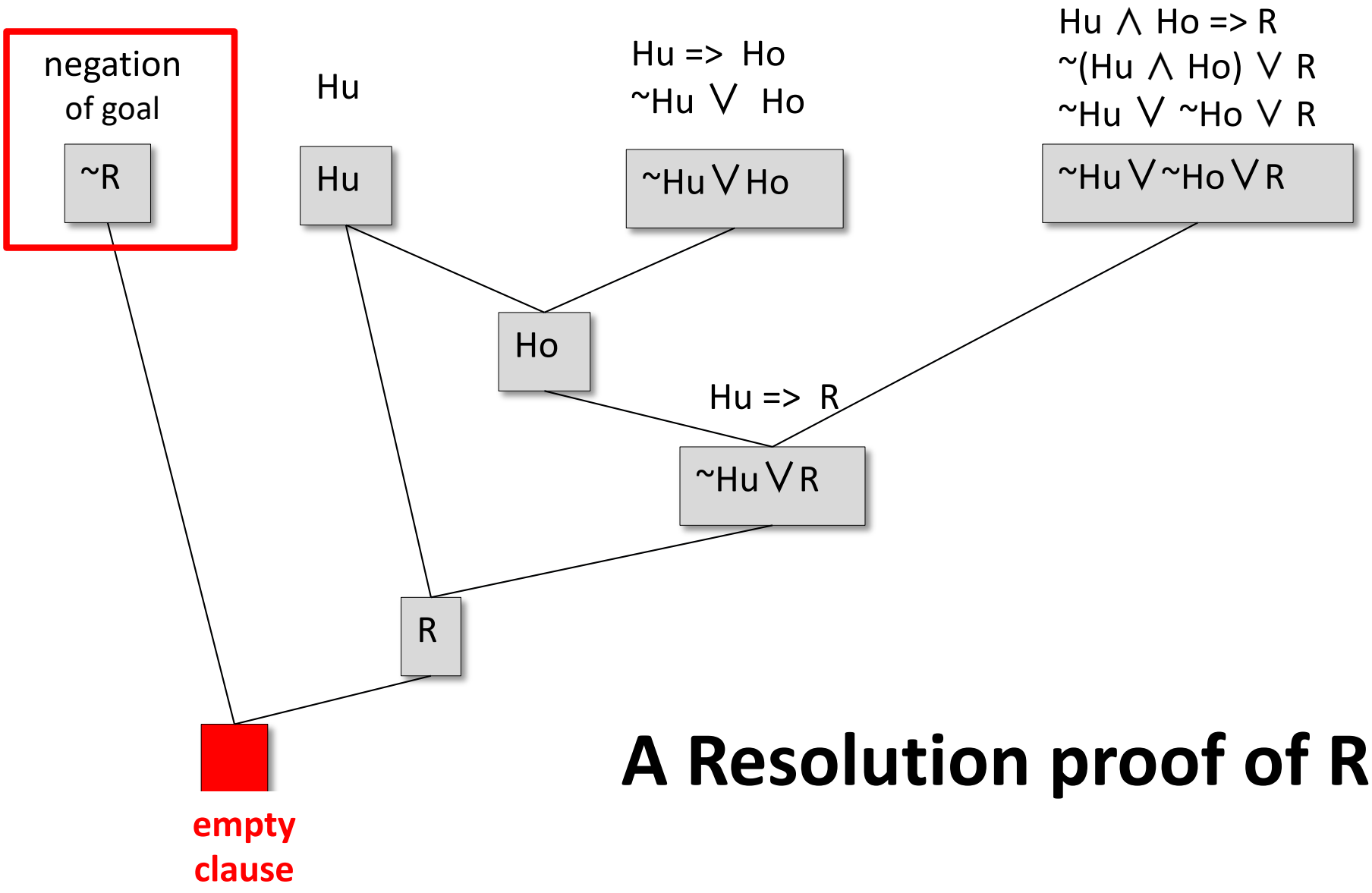
- It is sound!
- It's not *generatively complete* in that it can't derive all clauses that follow from the KB
  - The issues are not serious limitations, though
  - Example: if the KB includes P and includes Q we won't derive P ^ Q
- It will always terminate
- But generating all clauses that follow can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

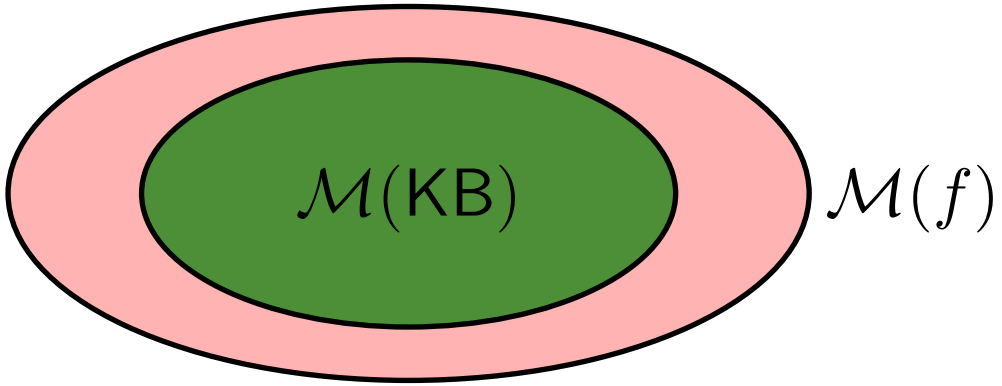


A Resolution proof of R

Desiderata for inference rules

Semantics

Interpretation defines **entailed/true** formulas: $\text{KB} \models f$:



Syntax:

Inference rules **derive** formulas: $\text{KB} \vdash f$

How does $\{f : \text{KB} \models f\}$ relate to $\{f : \text{KB} \vdash f\}$?

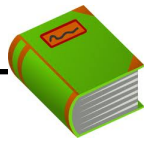
- We can apply inference rules all day long, but now we desperately need some guidance on whether a set of inference rules is doing anything remotely sensible.
- For this, we turn to semantics, which gives an objective notion of truth. Recall that the semantics provides us with $\mathcal{M}$, the set of satisfiable models for each formula f or knowledge base. This defines a set of formulas $\{f : \mathsf{KB} \models f\}$ which are defined to be true.
- On the other hand, inference rules also gives us a mechanism for generating a set of formulas, just by repeated application. This defines another set of formulas $\{f : \mathsf{KB} \vdash f\}$.

Truth

$$\{f : \text{KB} \models f\}$$

- Imagine a glass that represents the set of possible formulas entailed by the KB (these are necessarily true).
- By applying inference rules, we are filling up the glass with water.

Soundness



Definition: soundness

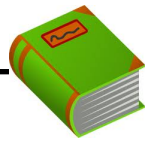
A set of inference rules Rules is sound if:

$$\{f : \text{KB} \vdash f\} \subseteq \{f : \text{KB} \models f\}$$

An inference rule is sound if every formula f it produces from a KB logically follows from the KB i.e., inference rule creates no contradictions

- We say that a set of inference rules is **sound** if using those inference rules, we never overflow the glass: the set of derived formulas is a subset of the set of true/entailed formulas.

Completeness



Definition: completeness

A set of inference rules Rules is complete if:

$$\{f : \text{KB} \vdash f\} \supseteq \{f : \text{KB} \models f\}$$

it can produce every formula that logically follows from (is entailed by) the KB
- Similar to complete search algorithms

- We say that a set of inference rules is **complete** if using those inference rules, we fill up the glass to the brim (and possibly go over): the set of derived formulas is a superset of the set of true/entailed formulas.

Sound rules of inference

Examples of sound rules of inference

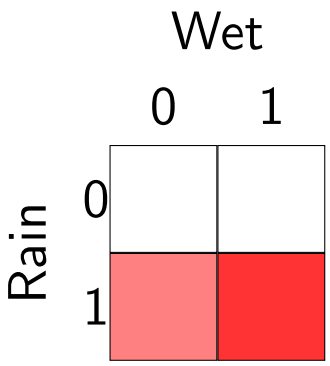
Each can be shown to be sound using a truth table

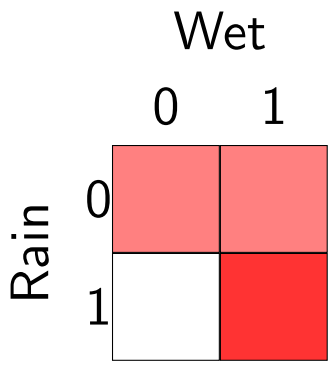
RULE	PREMISE	CONCLUSION
Modus Ponens	$A, A \rightarrow B$	B
And Introduction	A, B	$A \wedge B$
And Elimination	$A \wedge B$	A
Double Negation	$\neg\neg A$	A
Unit Resolution	$A \vee B, \neg B$	A
Resolution	$\mathbf{A \vee B, \neg B \vee C}$	$\mathbf{A \vee C}$

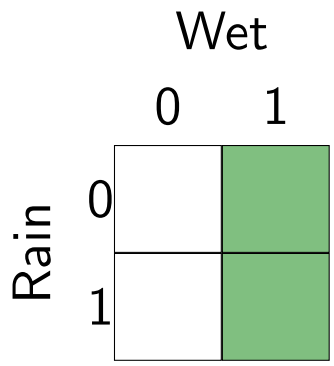
Soundness: example

Is $\dfrac{\text{Rain}, \quad \text{Rain} \to \text{Wet}}{\text{Wet}}$ (Modus ponens) sound?

$\mathcal{M}(\text{Rain}) \quad \cap \quad \mathcal{M}(\text{Rain} \to \text{Wet}) \quad \subseteq ? \quad \mathcal{M}(\text{Wet})$







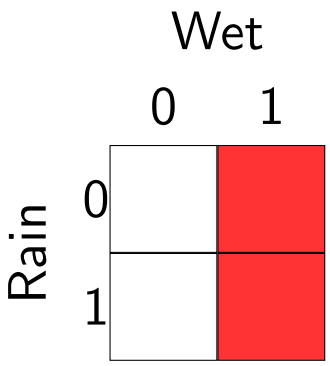
Sound!

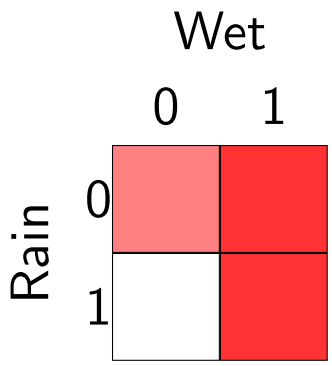
- To check the soundness of a set of rules, it suffices to focus on one rule at a time.
- Take the modus ponens rule, for instance. We can derive Wet using modus ponens. To check entailment, we map all the formulas into semantics-land (the set of satisfiable models). Because the models of Wet is a superset of the intersection of models of Rain and Rain $\rightarrow$ Wet (remember that the models in the KB are an intersection of the models of each formula), we can conclude that Wet is also entailed. If we had other formulas in the KB, that would reduce both sides of $\subseteq$ by the same amount and won't affect the fact that the relation holds. Therefore, this rule is sound.
- Note, we use Wet and Rain to make the example more colorful, but this argument works for arbitrary propositional symbols.

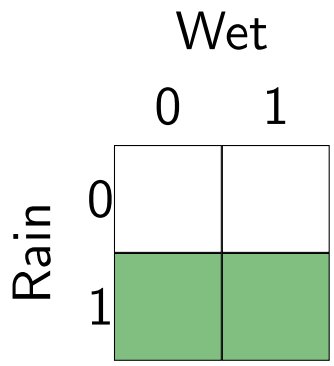
Soundness: example

Is $\dfrac{\text{Wet}, \quad \text{Rain} \to \text{Wet}}{\text{Rain}}$ sound?

$\mathcal{M}(\text{Wet}) \quad \cap \quad \mathcal{M}(\text{Rain} \to \text{Wet}) \quad \subseteq ? \quad \mathcal{M}(\text{Rain})$







Unsound!

- Here is another example: given Wet and Rain $\rightarrow$ Wet, can we infer Rain? To check it, we mechanically construct the models for the premises and conclusion. Here, the intersection of the models in the premise are not a subset, then the rule is unsound.
- Indeed, backward reasoning is faulty. Note that we can actually do a bit of backward reasoning using Bayesian networks, since we don't have to commit to 0 or 1 for the truth value.

Completeness: example

Recall completeness: inference rules derive all entailed formulas (f such that KB $\models f$)

Example: Modus ponens is incomplete

Setup:

$$\text{KB} = \{\text{Rain}, \text{Rain} \vee \text{Snow} \rightarrow \text{Wet}\}$$

$$f = \text{Wet}$$

$$\text{Rules} = \left\{\frac{f, \quad f \rightarrow g}{g}\right\} \text{ (Modus ponens)}$$

Semantically: KB $\models f$ (f is entailed).

Syntactically: KB $\nvdash f$ (can't derive f).

Incomplete!

- Completeness is trickier, and here is a simple example that shows that modus ponens alone is not complete, since it can't derive Wet, when semantically, Wet is true!

Fixing completeness

Option 1: Restrict the allowed set of formulas

propositional logic

propositional logic with only Horn clauses

Option 2: Use more powerful inference rules

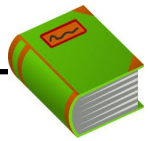
Modus ponens

resolution

- At this point, there are two ways to fix completeness. First, we can restrict the set of allowed formulas, making the water glass smaller in hopes that modus ponens will be able to fill that smaller glass.
- Second, we can use more powerful inference rules, pouring more vigorously into the same glass in hopes that this will be able to fill the glass; we'll look at one such rule, resolution, in the next lecture.

Definite clauses

aka Strict Horn Clauses



Definition: Definite clause

A **definite clause** has the following form:

$$(p_1 \wedge \cdots \wedge p_k) \rightarrow q$$

where $p_1, \ldots, p_k, q$ are propositional symbols.

Intuition: if $p_1, \ldots, p_k$ hold, then q holds.

Example: $(\text{Rain} \wedge \text{Snow}) \rightarrow \text{Traffic}$

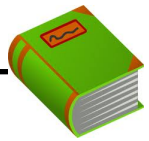
Example: Traffic

Non-example: $\neg$Traffic

Non-example: $(\text{Rain} \wedge \text{Snow}) \rightarrow (\text{Traffic} \vee \text{Peaceful})$

- First we will choose to restrict the allowed set of formulas. Towards that end, let's define a **definite clause** as a formula that says, if a conjunction of propositional symbols holds, then some other propositional symbol q holds. Note that this is a formula, not to be confused with an inference rule.

Horn clauses



Definition: Horn clause

A **Horn clause** is either:

- a definite clause $(p_1 \wedge \cdots \wedge p_k \rightarrow q)$
- a goal clause $(p_1 \wedge \cdots \wedge p_k \rightarrow \text{false})$

Example (definite): $(\text{Rain} \wedge \text{Snow}) \rightarrow \text{Traffic}$

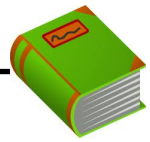
Example (goal): $\text{Traffic} \wedge \text{Accident} \rightarrow \text{false}$

equivalent: $\neg(\text{Traffic} \wedge \text{Accident})$

- A **Horn clause** is basically a definite clause, but includes another type of clause called a **goal clause**, which is the conjunction of a bunch of propositional symbols implying false. The form of the goal clause might seem a bit strange, but the way to interpret it is simply that it's the negation of the conjunction.

Modus ponens

Inference rule:



Definition: Modus ponens

$$\frac{p_1, \quad \cdots \quad , p_k, \quad (p_1 \wedge \cdots \wedge p_k) \rightarrow q}{q}$$

Example:

Example: Modus ponens

$$\frac{\text{Wet}, \quad \text{Weekday}, \quad \text{Wet} \wedge \text{Weekday} \rightarrow \text{Traffic}}{\text{Traffic}}$$

- Recall the Modus ponens rule from before. We simply have generalized it to arbitrary number of premises.

Completeness of modus ponens

Theorem: Modus ponens on Horn clauses

Modus ponens is **complete** with respect to Horn clauses:

- Suppose KB contains only Horn clauses and p is an entailed propositional symbol.
- Then applying modus ponens will derive p.

Upshot:

$$\text{KB} \models p \text{ (entailment) is the same as KB} \vdash p \text{ (derivation)!}$$

- There's a theorem that says that modus ponens is complete on Horn clauses. This means that any propositional symbol that is entailed can be derived by modus ponens too, provided that all the formulas in the KB are Horn clauses.
- We already proved that modus ponens is sound, and now we have that it is complete (for Horn clauses). The upshot of this is that entailment (a semantic notion, what we care about) and being able to derive a formula (a syntactic notion, what we do with inference) are equivalent!

Example: Modus ponens

KB

Rain

Weekday

Rain $\rightarrow$ Wet

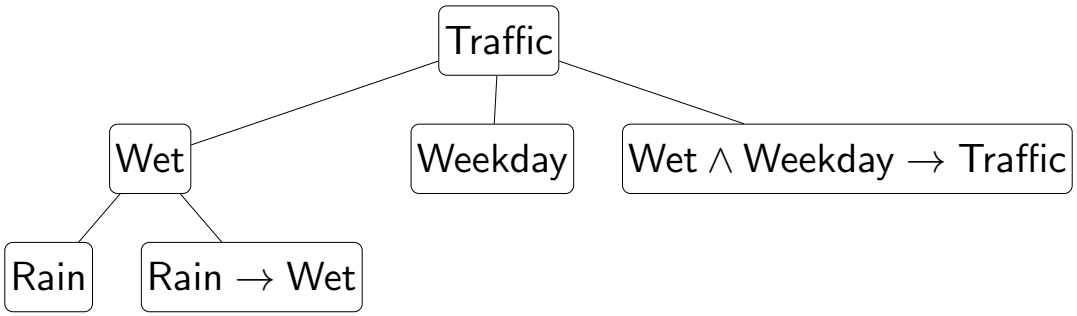
Wet $\wedge$ Weekday $\rightarrow$ Traffic

Traffic $\wedge$ Careless $\rightarrow$ Accident

Definition: Modus ponens

$$\frac{p_1, \quad \cdots \quad , p_k, \quad (p_1 \wedge \cdots \wedge p_k) \rightarrow q}{q}$$

Question: $\text{KB} \models \text{Traffic} \quad \Leftrightarrow \quad \text{KB} \vdash \text{Traffic}$



- Let's see modus ponens on Horn clauses in action. Suppose we have the given KB consisting of only Horn clauses (in fact, these are all definite clauses), and we wish to ask whether the KB entails Traffic.
- We can construct a **derivation**, a tree where the root formula (e.g., Traffic) was derived using inference rules.
- The leaves are the original formulas in the KB, and each internal node corresponds to a formula which is produced by applying an inference rule (e.g., modus ponens) with the children as premises.
- If a symbol is used as the premise in two different rules, then it would have two parents, resulting in a DAG.

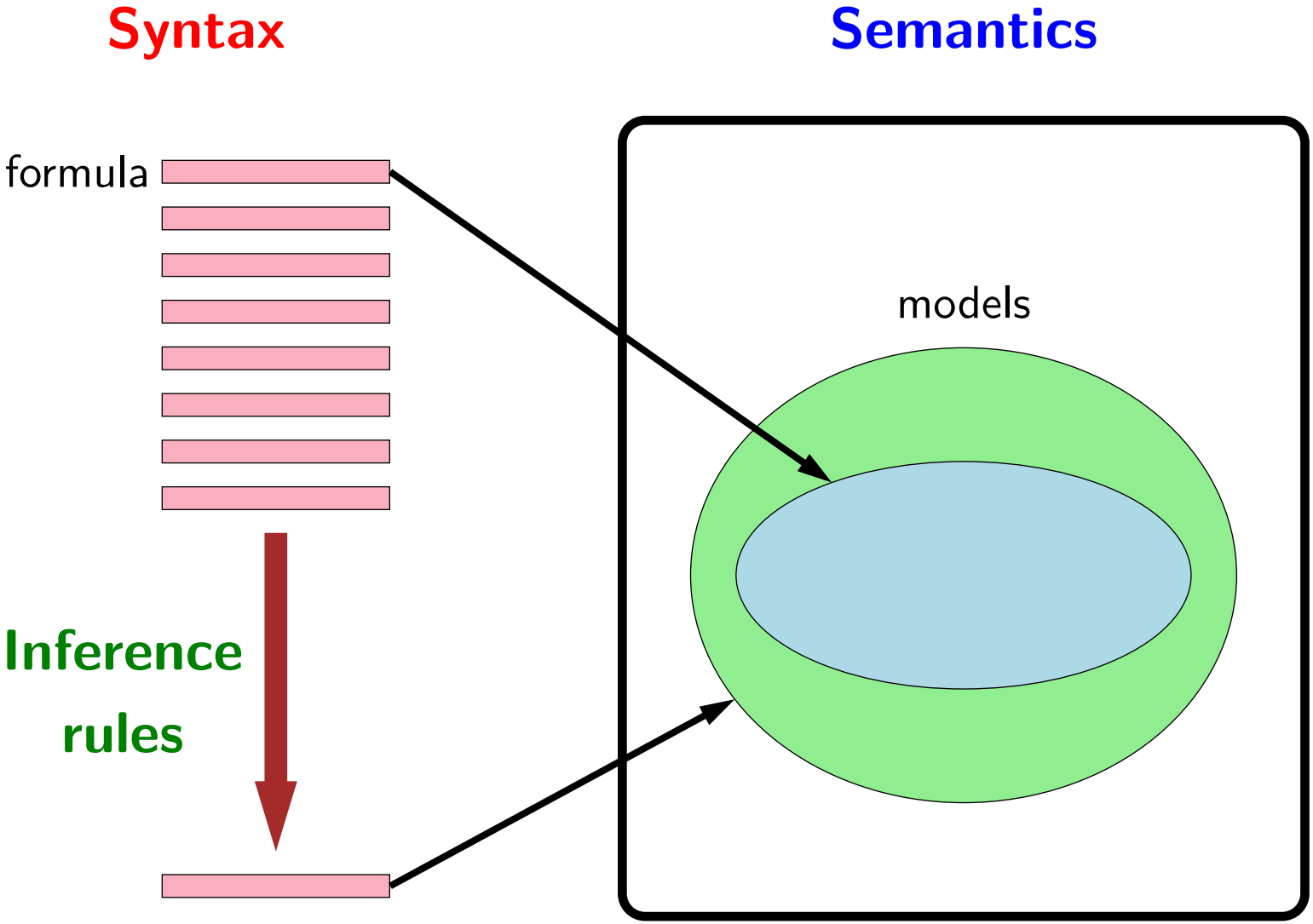
Propositional logic: pro and con

- **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))

Summary



Hunt the Wumpus

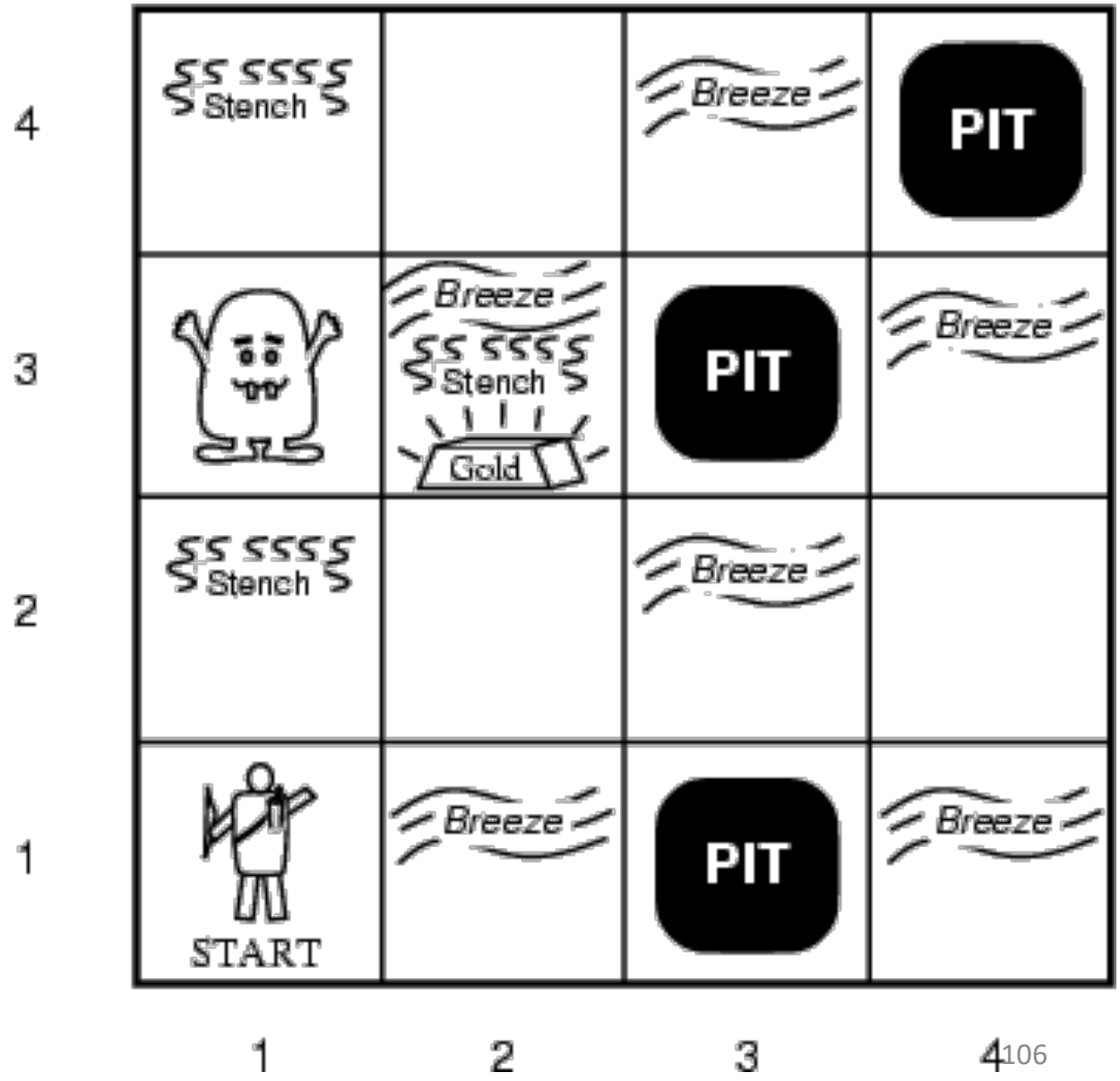
Wumpus World environment

- 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

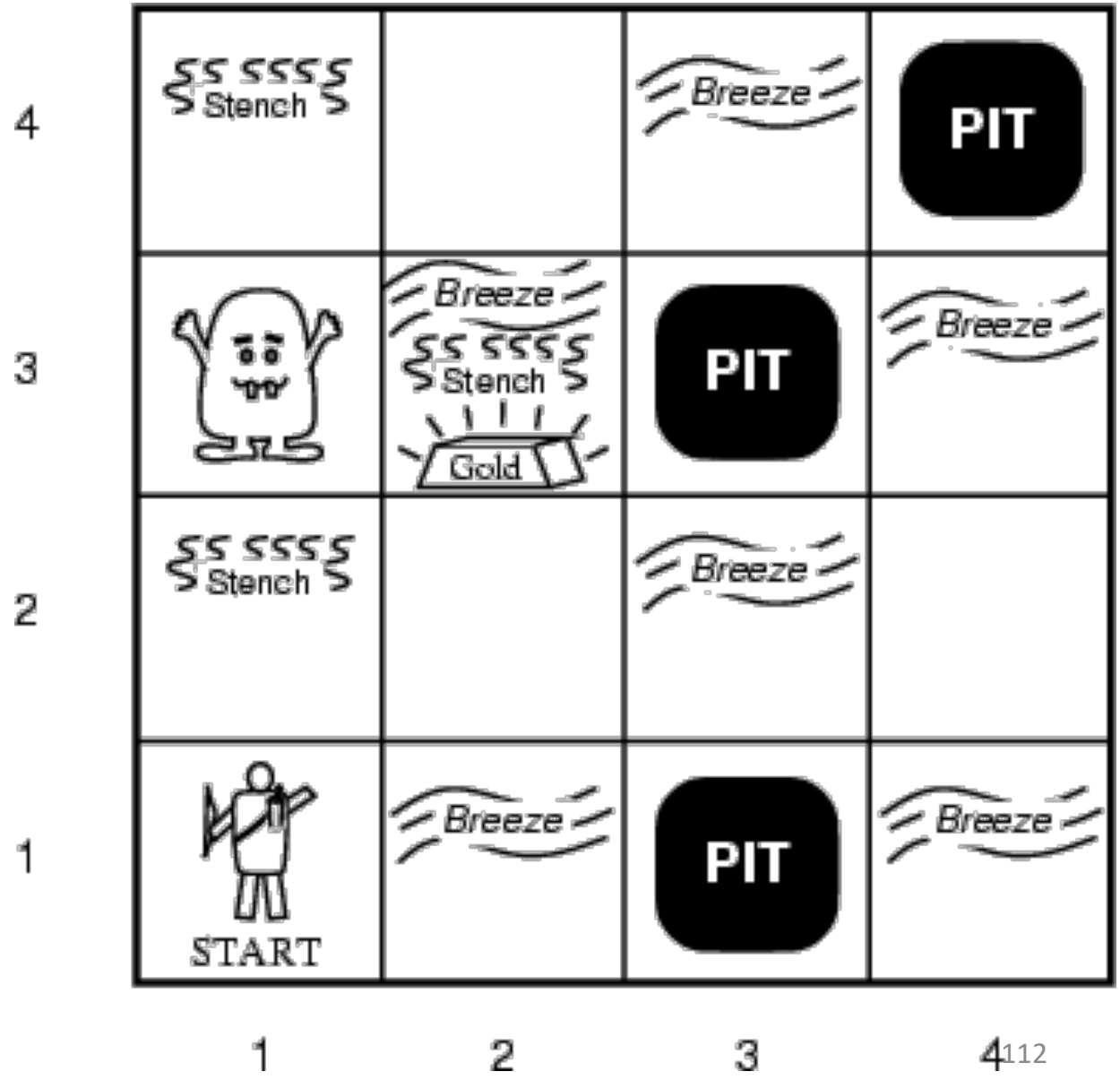
Agent's goal is to find the gold and bring it back to the start square as quickly as possible, without getting killed

- 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

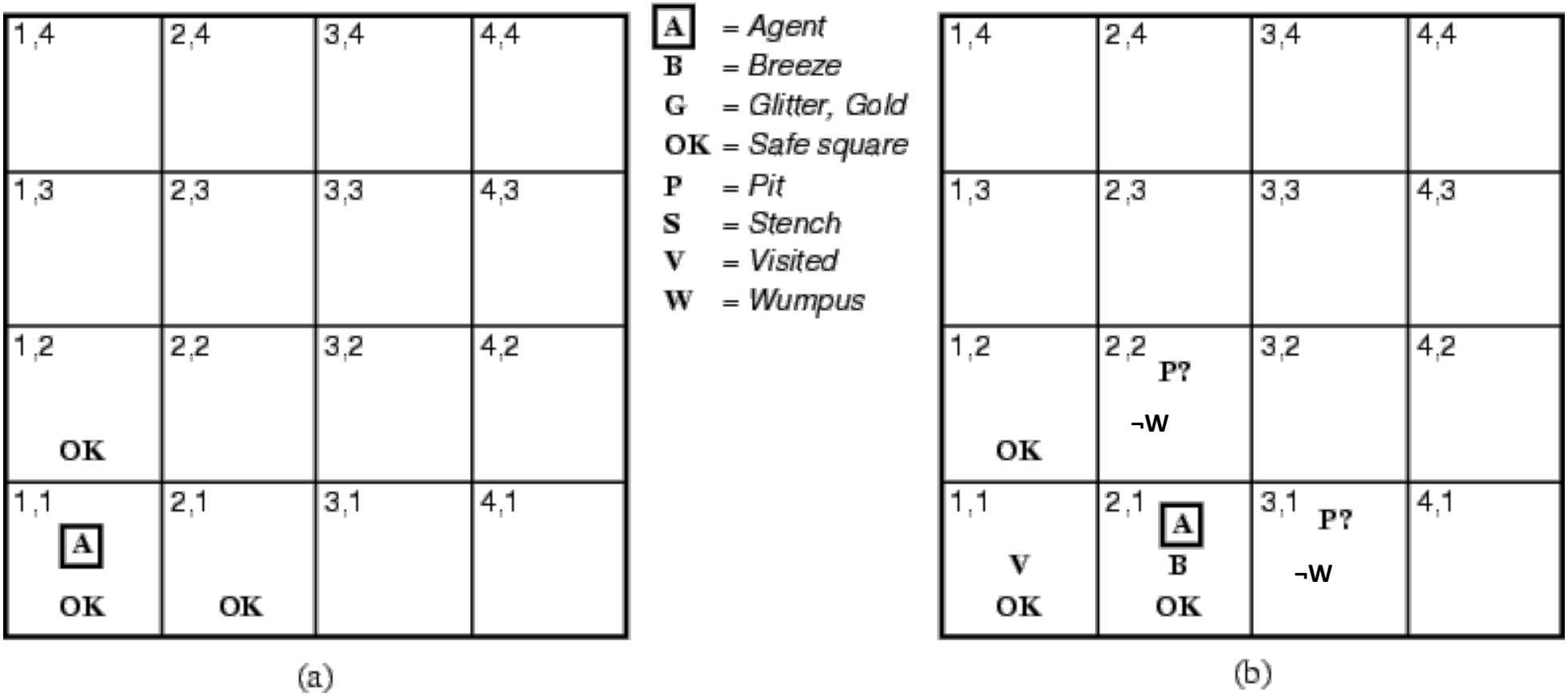
AIMA's Wumpus World

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

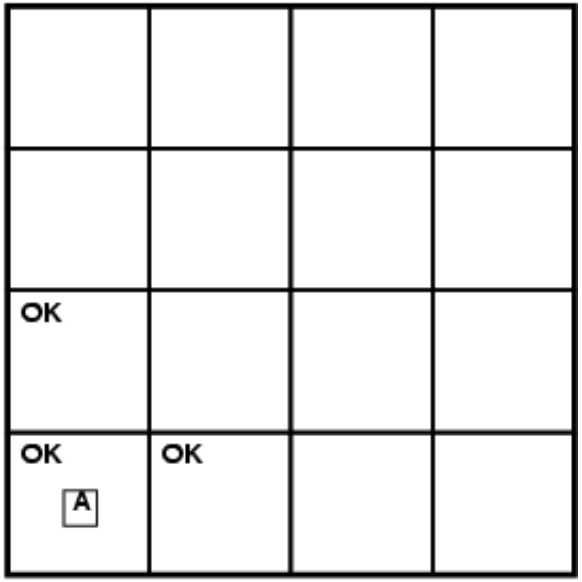


(a) (b)

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

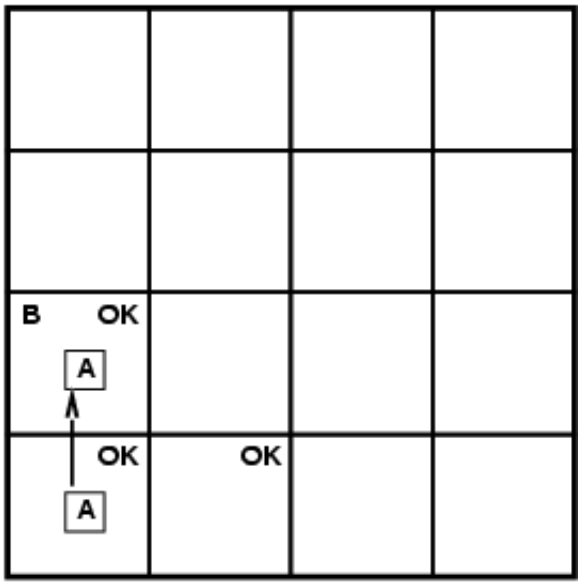
Moving to [2,1] is a **safe move** that reveals a breeze but no stench, **implying** that Wumpus isn't adjacent but one or more pits are

Exploring a wumpus world



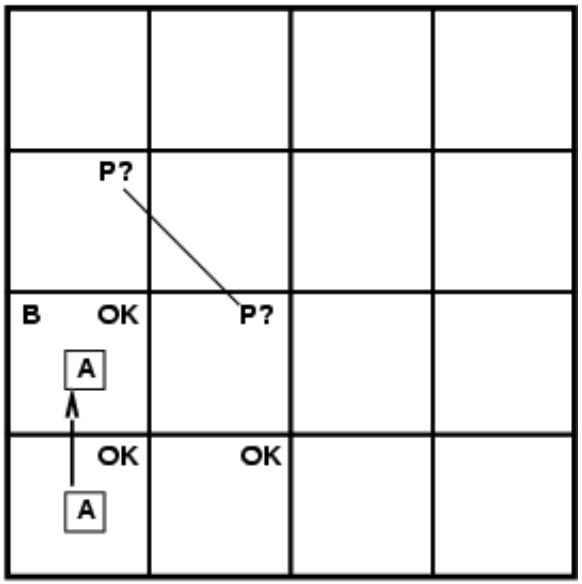
A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

Exploring a wumpus world



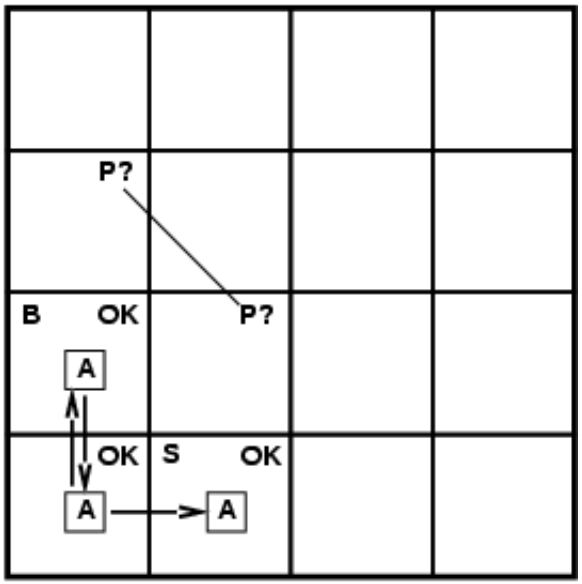
A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

Exploring a wumpus world



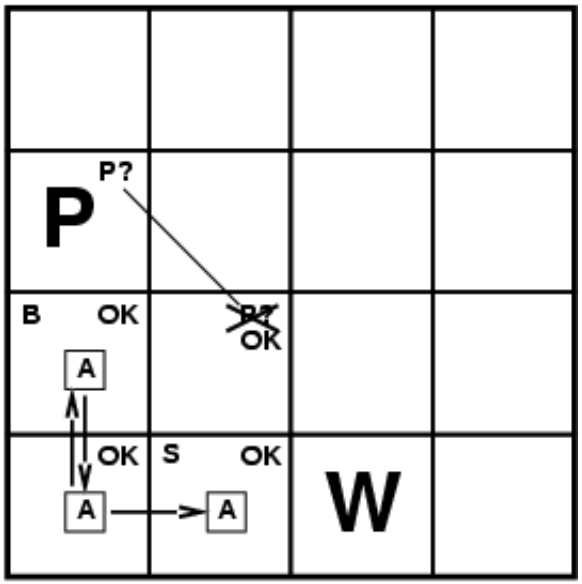
A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

Exploring a wumpus world



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

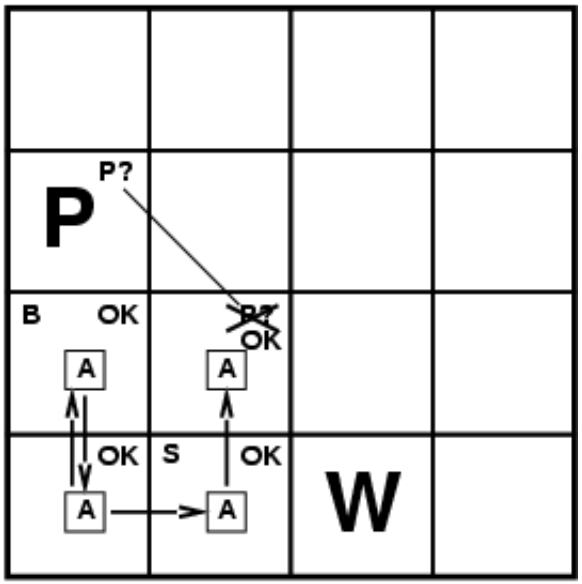
Exploring a wumpus world



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

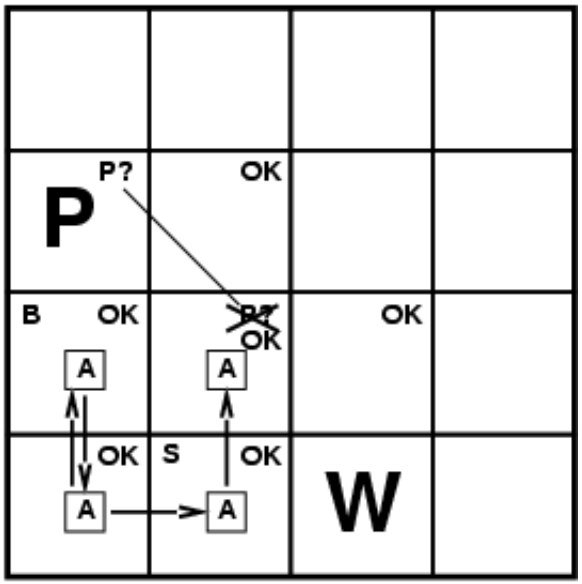
No stench in (1,2) => Wumpus not in (2,2)
No breeze in (2,1) => no pit in (2,2) => pit in (1,3)

Exploring a wumpus world



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

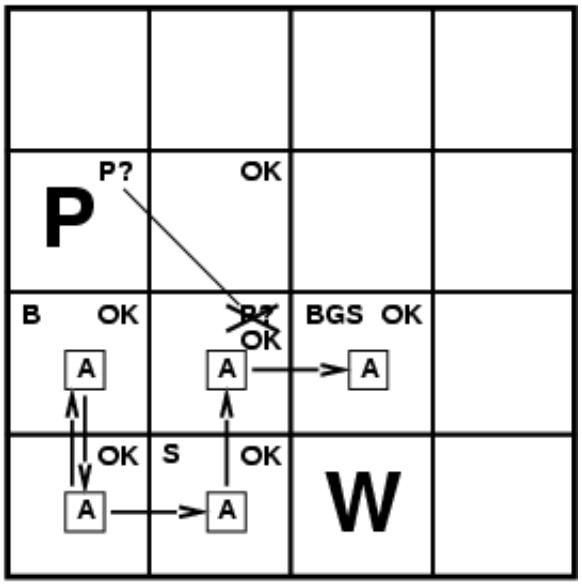
Exploring a wumpus world



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

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

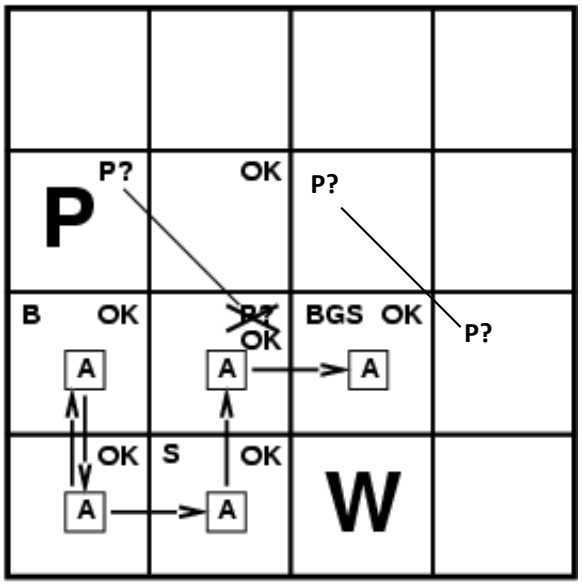
Exploring a wumpus world



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



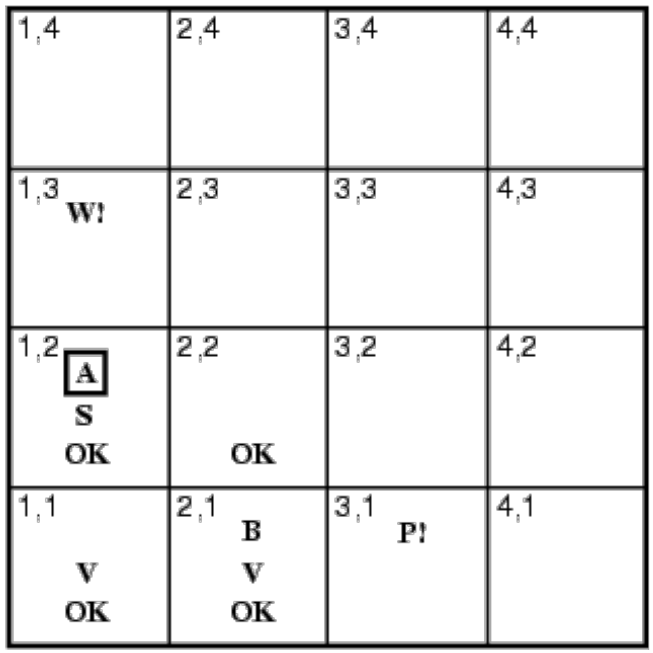
A	agent
B	breeze
G	glitter
OK	safe cell
P	pit
S	stench
W	wumpus

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:
  - $\neg S22 \rightarrow \neg W12 \wedge \neg W23 \wedge \neg W32 \wedge \neg W21$
  - $S22 \rightarrow W12 \vee W23 \vee W32 \vee W21$
  - $B22 \rightarrow P12 \vee P23 \vee P32 \vee P21$
  - $W22 \rightarrow S12 \wedge S23 \wedge S32 \wedge W21$
  - $W22 \rightarrow \neg W11 \wedge \neg W21 \wedge \ldots \neg W44$
  - $A22 \rightarrow V22$
  - $A22 \rightarrow \neg W11 \wedge \neg W21 \wedge \ldots \neg W44$
  - $V22 \rightarrow OK22$

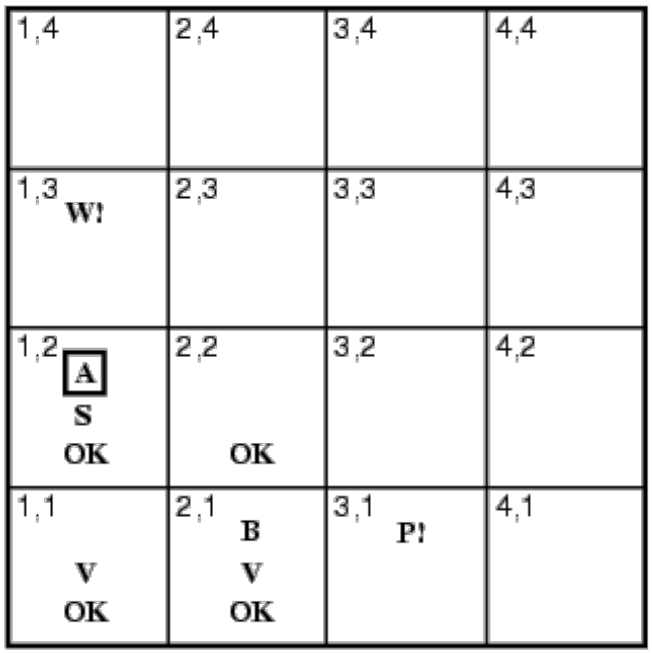


If there's no stench in cell 2,2 then the Wumpus isn't in cell 21, 23 32 or 21

Hunt the Wumpus domain

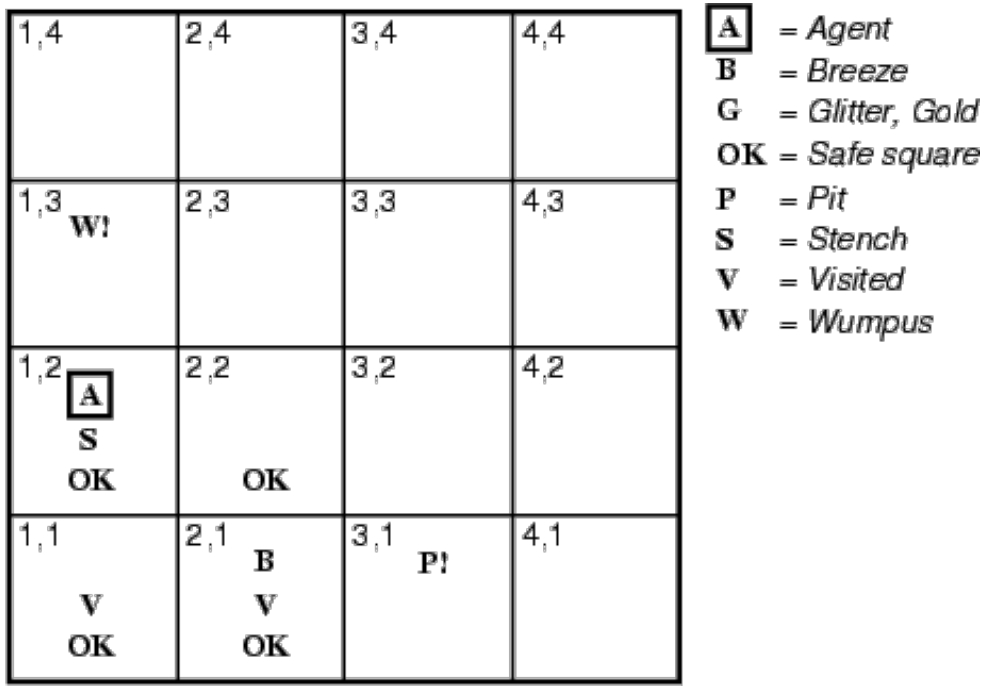
- Eight symbols for each cell, i.e.: A11, B11, G11, OK11, P11, S11, V11, W11
- Lack of variables requires giving similar rules for each cell!
- Ten rules (I think) for each

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



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

After third move



- 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