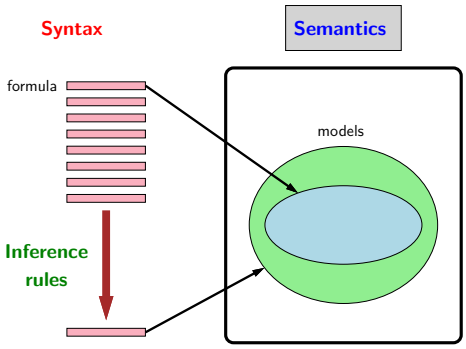
Logic: propositional logic semantics

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

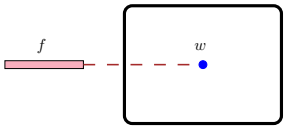
Definition: interpretation function

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.

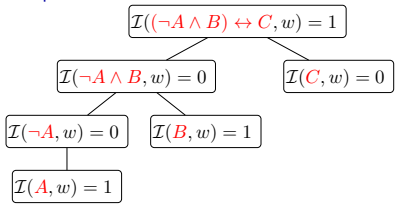
Interpretation function: example

Example: interpretation function

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

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

Interpretation:



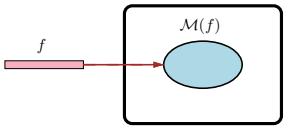
- For example, given the formula, we break down the formula into parts, recursively compute the truth value of the parts, and then finally combines these truth values based on the connective.

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 $\mathcal{I}(f, w) = 1$.

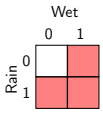
Models: example

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

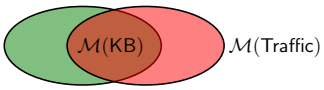
Definition: Knowledge base

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.

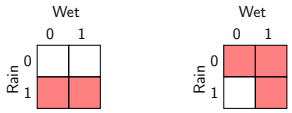
Let KB $= \{\text{Rain} \vee \text{Snow}, \text{Traffic}\}$.



- 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.

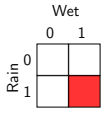
Knowledge base: example

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



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.

Adding to the knowledge base

Adding more formulas to the knowledge base:

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

Shrinks the set of models:

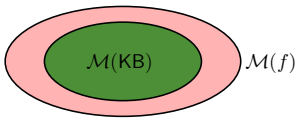
$\mathcal{M}(\text{KB})$ $\longrightarrow$ $\mathcal{M}(\text{KB}) \cap \mathcal{M}(f)$

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).

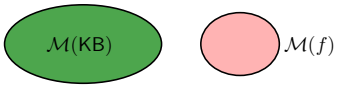
Definition: entailment

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

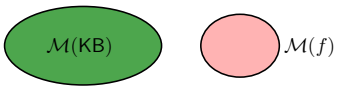
KB contradicts f iff $\mathcal{M}(\text{KB}) \cap \mathcal{M}(f) = \emptyset$.

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

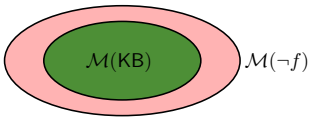
- The second case is if the set of models defined by f is completely disjoint from those of KB. Then we say that the KB and f **contradict** each other. If we believe KB, then we cannot possibly believe f.

Contradiction and entailment

Contradiction:



Entailment:

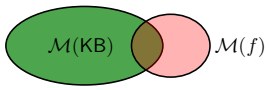


Proposition: contradiction and entailment

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.

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.

Tell operation

Tell[f] → KB → ?

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 = 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[f] → KB → ?

Ask: *Is it raining?*

Ask[Rain]

Possible responses:

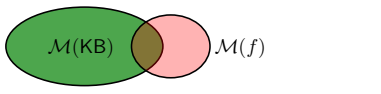
- Yes: entailment (KB $\models f$)
- No: contradiction (KB $\models \neg f$)
- I don't know: contingent

- 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.

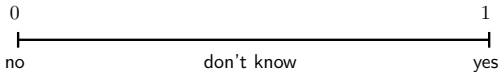
Digression: probabilistic generalization

Bayesian network: distribution over assignments (models)

w	$\mathbb{P}(W=w)$
{ A: 0, B: 0, C: 0 }	0.3
{ A: 0, B: 0, C: 1 }	0.1
...	...



$$\mathbb{P}(f \mid \text{KB}) = \frac{\sum_{w \in \mathcal{M}(\text{KB} \cup \{f\})} \mathbb{P}(W=w)}{\sum_{w \in \mathcal{M}(\text{KB})} \mathbb{P}(W=w)}$$



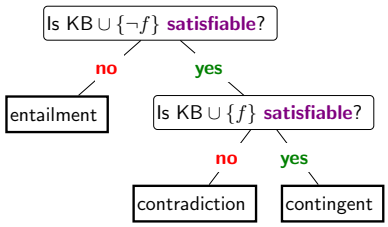
- Note that logic captures uncertainty in a very crude way. We can't say that we're almost sure or not very sure or not sure at all.
- Probability can help here. Remember that a Bayesian network (or more generally a factor graph) defines a distribution over assignments to the variables in the Bayesian network. Then we could ask questions such as: conditioned on having a cough but not itchy eyes, what's the probability of having a cold?
- Recall that in propositional logic, models are just assignments to propositional symbols. So we can think of KB as the evidence that we're conditioning on, and f as the query.

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 $f \in$ KB.
- With this definition in hand, we can implement Ask$[f]$ and Tell$[f]$ as follows:
- First, we check if 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 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

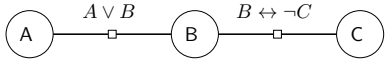
Example: model checking

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.