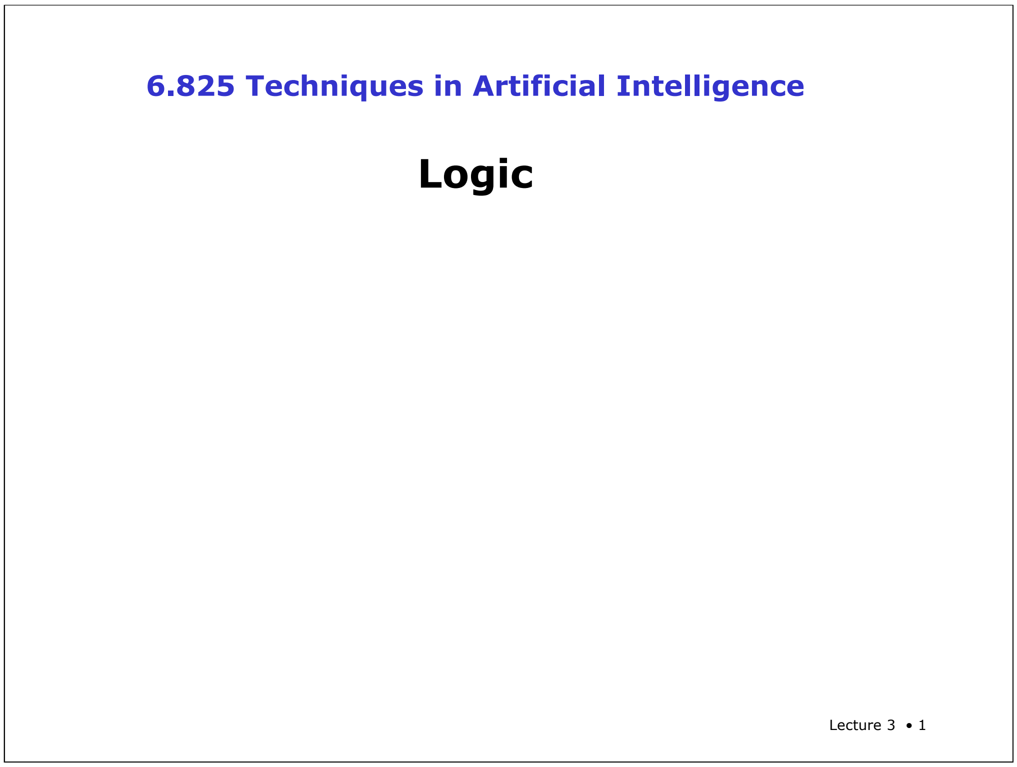
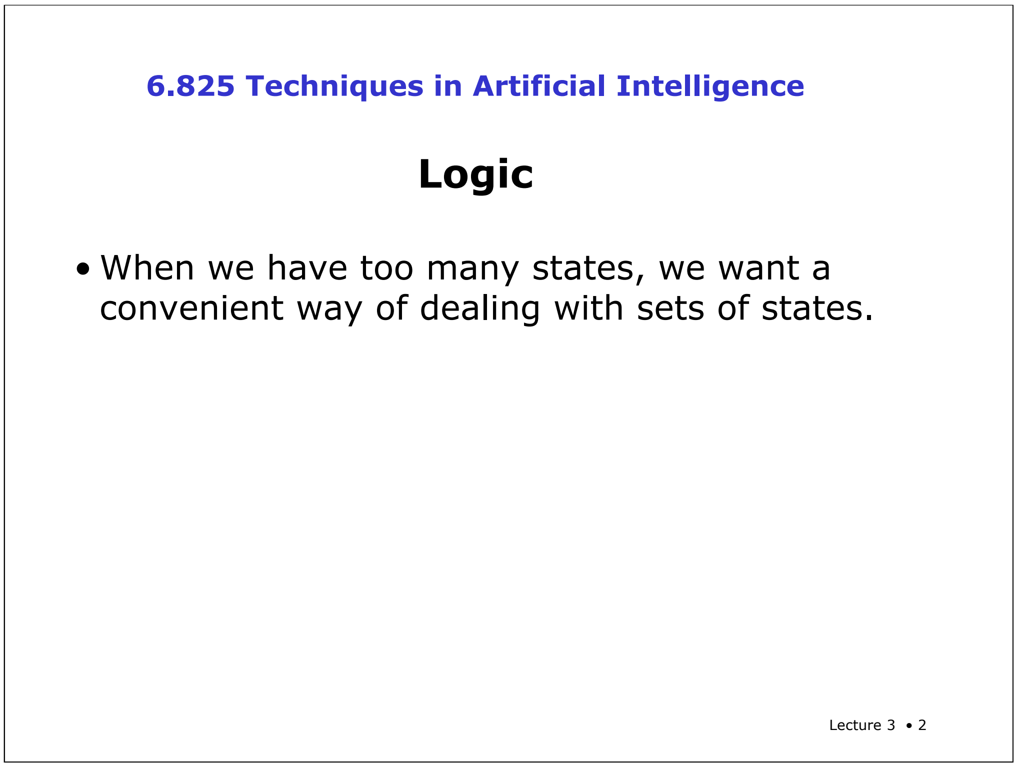


6.825 Techniques in Artificial Intelligence

Logic

Today we're going to start talking about logic. Now, my guess is that almost everybody's been exposed to basic propositional logic in the context of machine architecture or something like that. But, it turns out that that exposure to logic was just a little tiny piece of an enormous iceberg that we'll uncover a little bit more. We're not going to go crazy with logic in this course, but we're going to do a fair amount of it, partly because it's relevant -- it's certainly of great historical relevance in AI. And partly because it's making a resurgence, in real applications. So, for example, the web consortium now is interested in the idea of annotating web pages with logical descriptions of the content of what's on the web pages. We'll use some of those ideas to motivate our second assignment.

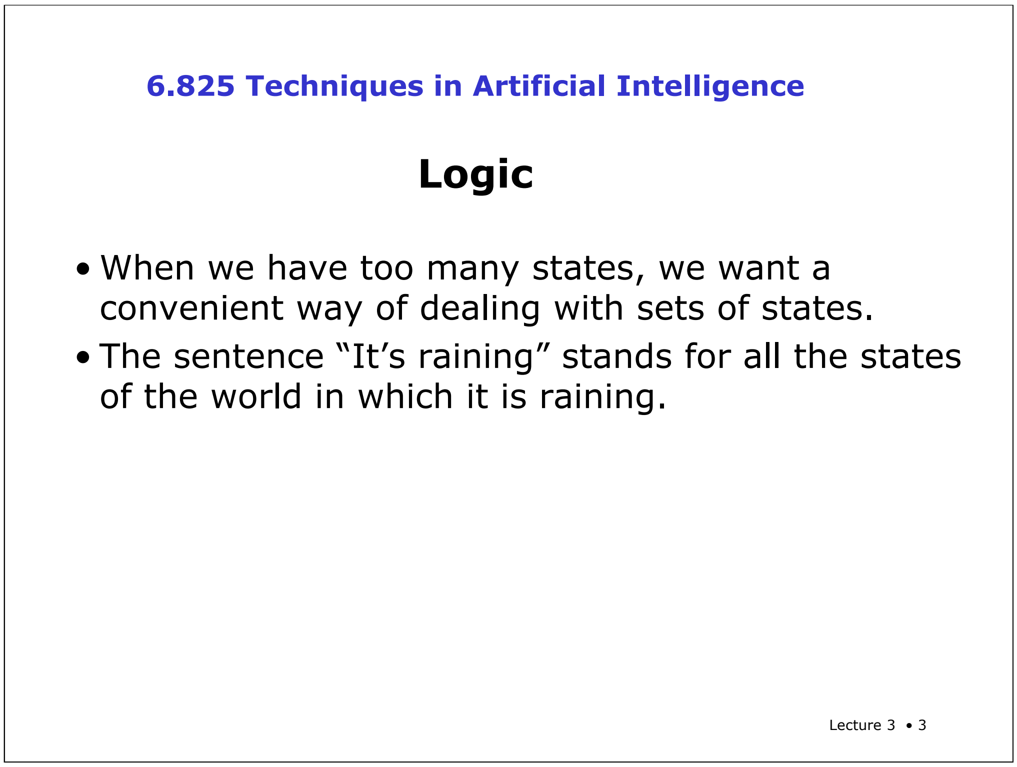
We're going to start right now with a simpler logic than they're using. We're going to try to introduce the basic and important ideas of logic in the context of a simple logic called propositional logic. And then we'll move on to first-order logic, which is a little more complicated.


6.825 Techniques in Artificial Intelligence

Logic

- When we have too many states, we want a convenient way of dealing with sets of states.

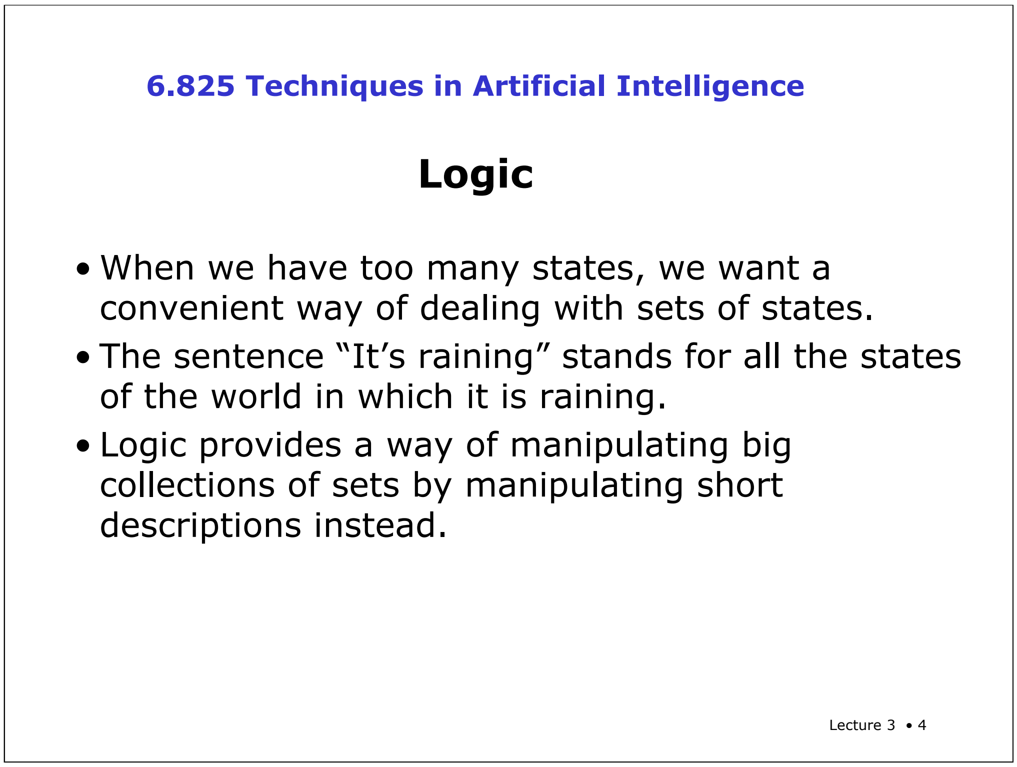
OK, so why do we need logic? Well, last time we talked about problem solving. We assumed that there were few enough states in the world that we could easily enumerate them, or make an array of them in our computer. But most domains we care about have way too many states to do that. In really big domains, you don't want to talk about individual states in particular, you'd like to talk about them in terms of sets. You'd like to have a convenient way to name sets of states, and you'd like not even to have to think about the individual states



Logic

- When we have too many states, we want a convenient way of dealing with sets of states.
- The sentence “It’s raining” stands for all the states of the world in which it is raining.

What if I say "It's raining."? One way to think about what it means -- what that assertion means, that it's raining -- is to say that it stands for all those states of the world in which it's really raining. So there's a very short name for an enormous set of states.

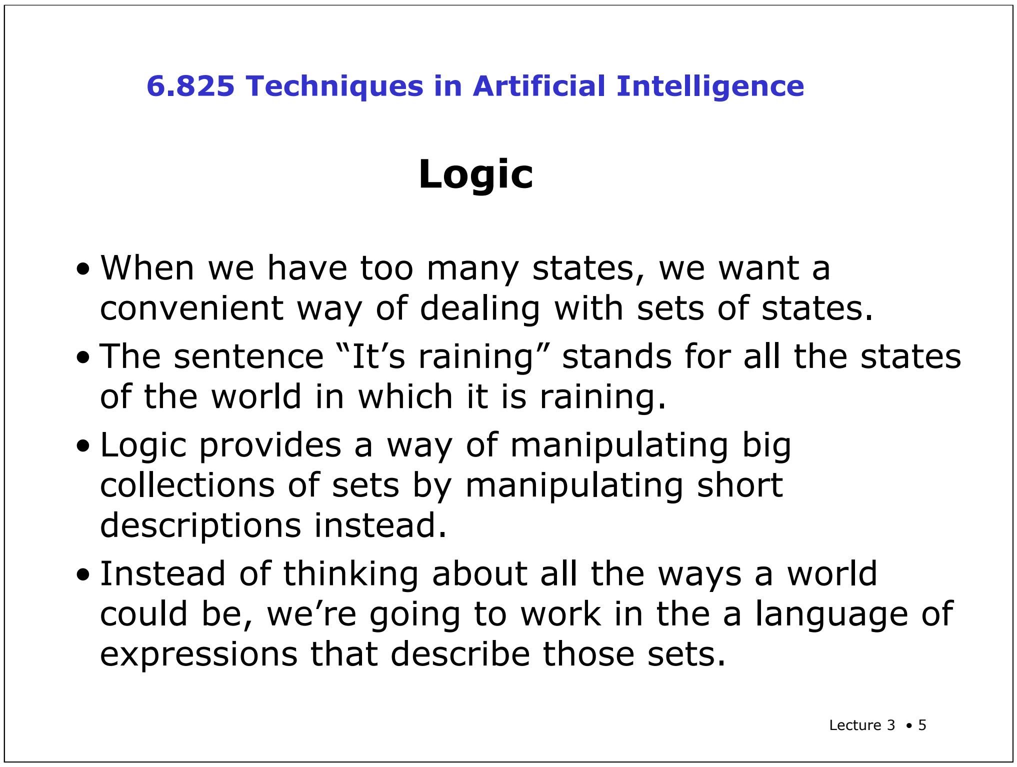


6.825 Techniques in Artificial Intelligence

Logic

- When we have too many states, we want a convenient way of dealing with sets of states.
- The sentence "It's raining" stands for all the states of the world in which it is raining.
- Logic provides a way of manipulating big collections of sets by manipulating short descriptions instead.

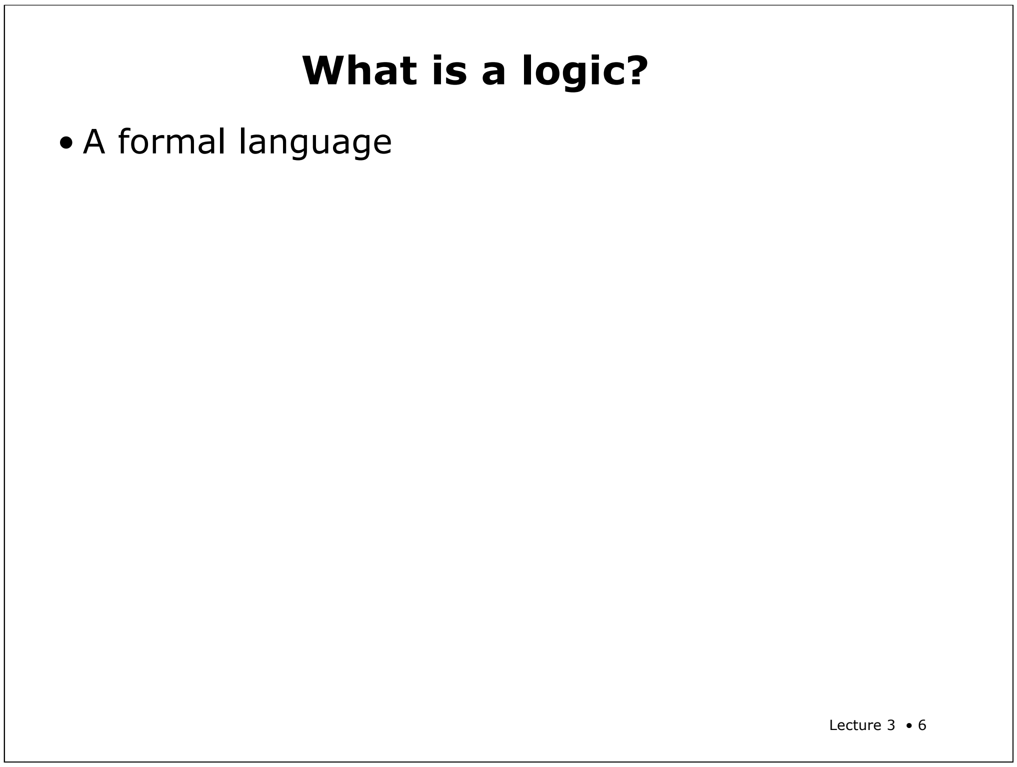
And logic is going to be a way to do that. To deal with little names for big collections of things, and to manipulate the big collections absolutely implicitly by manipulating the descriptions instead. So that's the enterprise -- short names for big things.



Logic

- When we have too many states, we want a convenient way of dealing with sets of states.
- The sentence “It’s raining” stands for all the states of the world in which it is raining.
- Logic provides a way of manipulating big collections of sets by manipulating short descriptions instead.
- Instead of thinking about all the ways a world could be, we’re going to work in the a language of expressions that describe those sets.

Instead of thinking about all the ways that the world could be, we're going to instead work in this language of names. We’ll use a language of expressions that describe those sets of states, and not worry about the sets.



What is a logic?

- A formal language

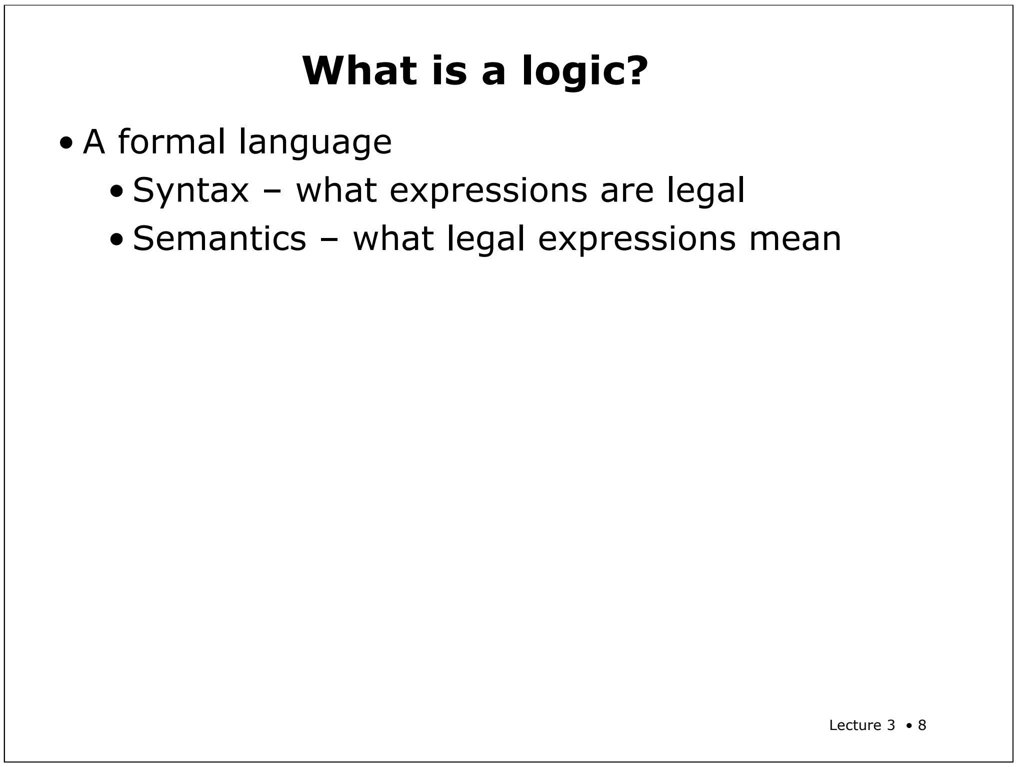
Lecture 3 • 6

So, what is a logic? Well, a logic is a formal language. And what does that mean? It has a syntax and a semantics, and a way of manipulating expressions in the language. We'll talk about each of these.

What is a logic?

- A formal language
    - Syntax – what expressions are legal

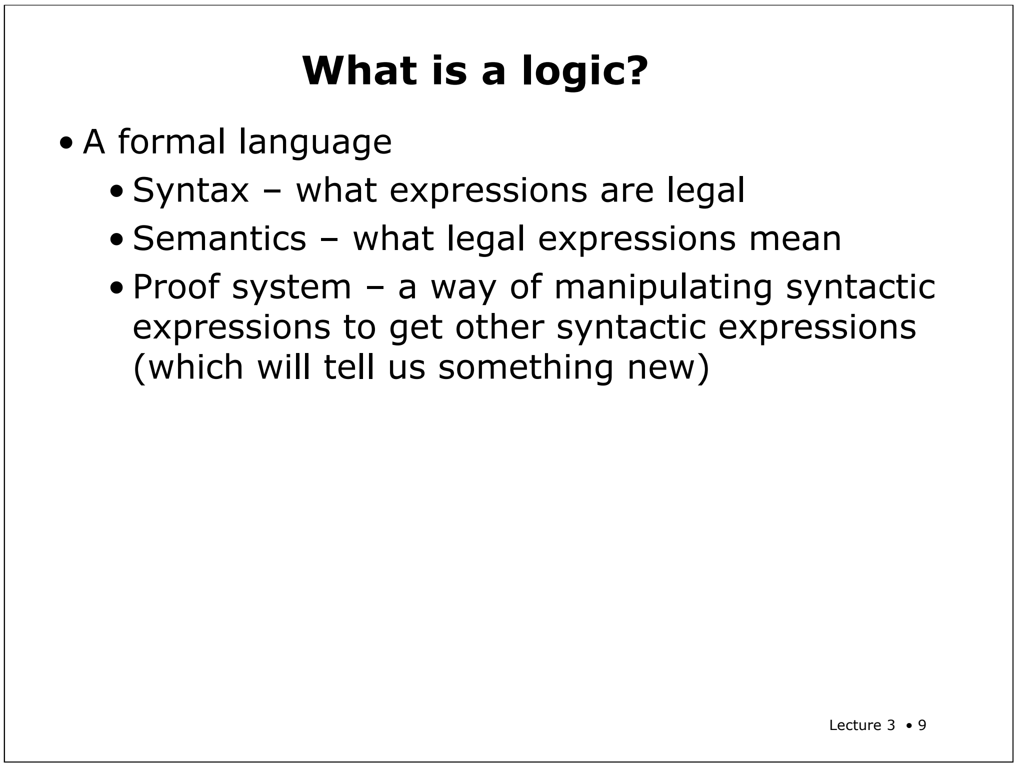
The syntax is a description of what you're allowed to write down, what the expressions are that are legal in a language. We'll define the syntax of a propositional logic in complete detail later in this lecture.



What is a logic?

- A formal language
  - Syntax – what expressions are legal
  - Semantics – what legal expressions mean

There's the semantics -- which is some story about what those expressions mean. Syntax is form and semantics is content.

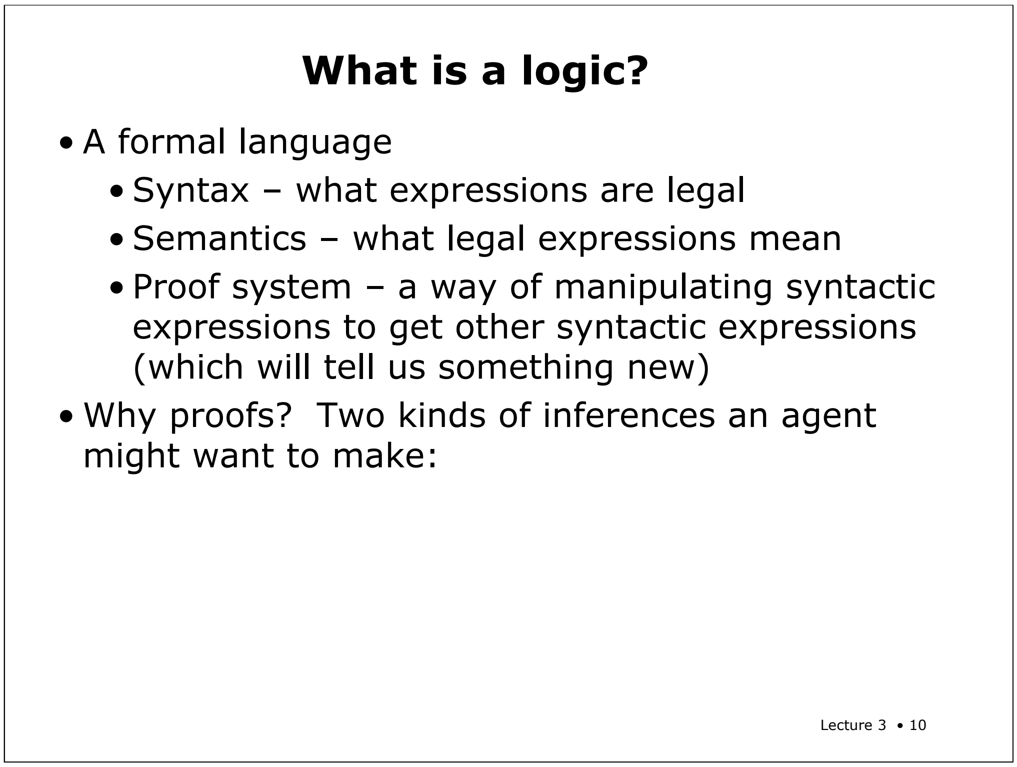


What is a logic?

- A formal language
  - Syntax – what expressions are legal
  - Semantics – what legal expressions mean
  - Proof system – a way of manipulating syntactic expressions to get other syntactic expressions (which will tell us something new)

A logic usually comes with a proof system, which is a way of manipulating syntactic expressions to get other syntactic expressions.

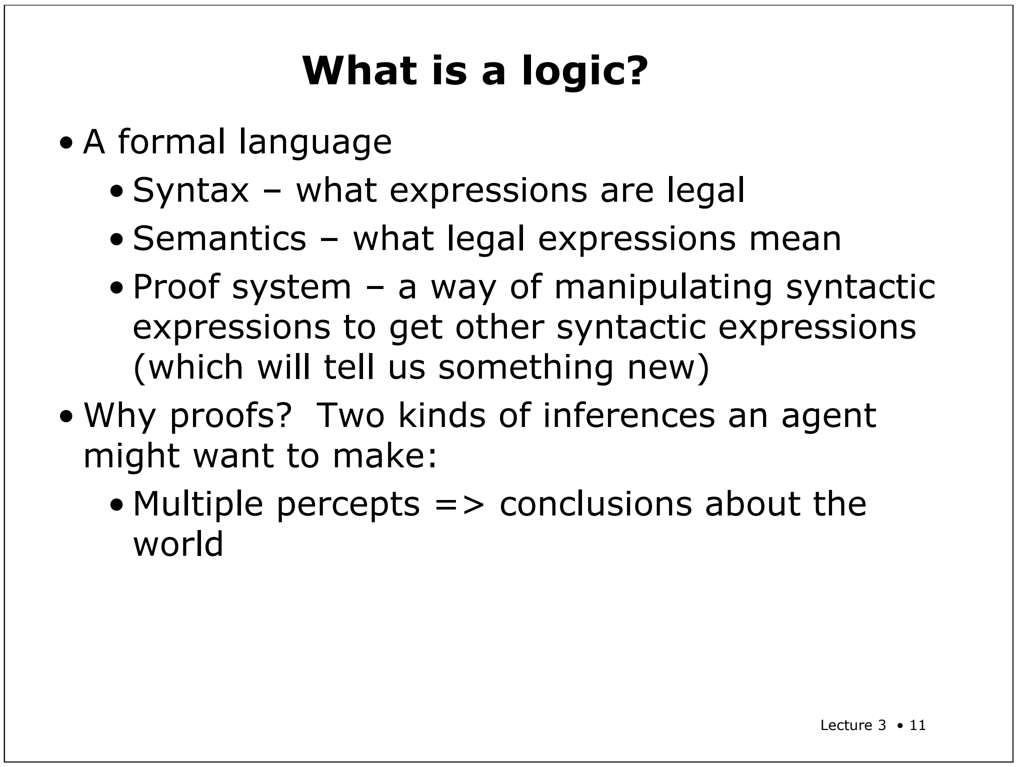
And, why are we interested in manipulating syntactic expressions? The idea is that if we use a proof system with the right kinds of properties, then the new syntactic expressions we create will have semantics or meanings that tell us something "new" about the world.



What is a logic?

- A formal language
  - Syntax – what expressions are legal
  - Semantics – what legal expressions mean
  - Proof system – a way of manipulating syntactic expressions to get other syntactic expressions (which will tell us something new)
- Why proofs? Two kinds of inferences an agent might want to make:

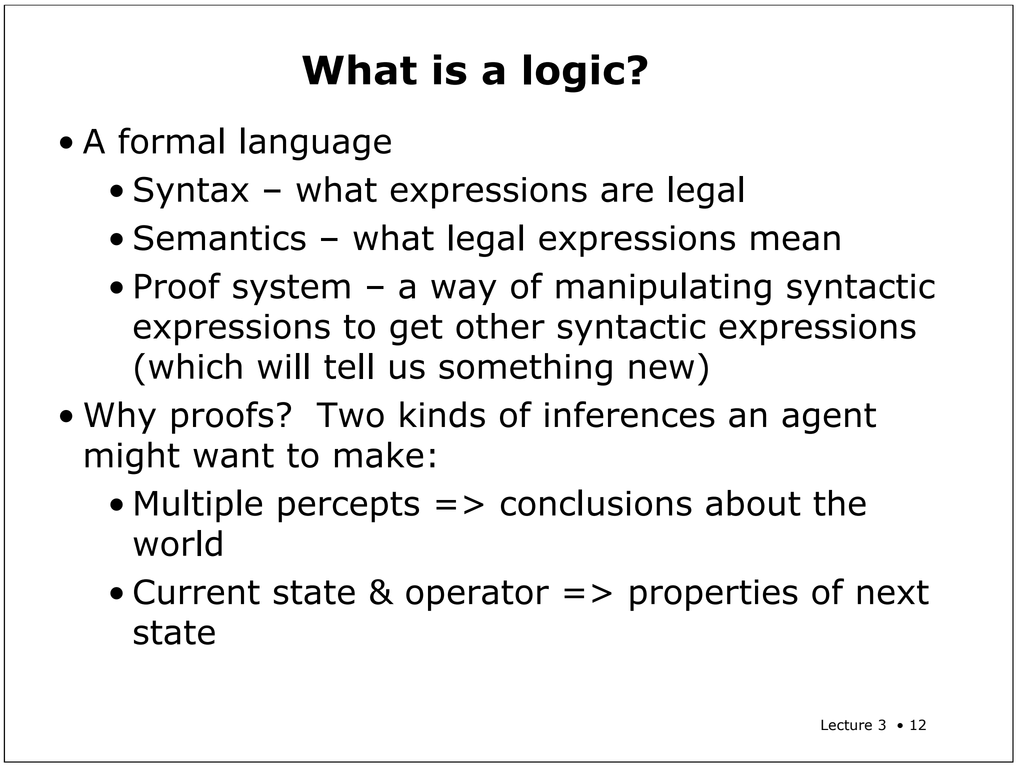
So, why do we want to do proofs? There are lots of situations.



What is a logic?

- A formal language
  - Syntax – what expressions are legal
  - Semantics – what legal expressions mean
  - Proof system – a way of manipulating syntactic expressions to get other syntactic expressions (which will tell us something new)
- Why proofs? Two kinds of inferences an agent might want to make:
  - Multiple percepts => conclusions about the world

In the context of an agent trying to reason about its world, think about a situation where we have a bunch of percepts. You know, we found somebody come in with a dripping umbrella, we saw muddy tracks in the hallway, we see that there's not much light coming in the windows, we hear pitter-pitter-patter. We have all these percepts, and we'd like to draw some conclusion from them, meaning that we'd like to figure out something about what's going on in the world. We'd like to take all these percepts together and draw some conclusion about the world. And we might use logic to do it.



What is a logic?

- A formal language
  - Syntax – what expressions are legal
  - Semantics – what legal expressions mean
  - Proof system – a way of manipulating syntactic expressions to get other syntactic expressions (which will tell us something new)
- Why proofs? Two kinds of inferences an agent might want to make:
  - Multiple percepts => conclusions about the world
  - Current state & operator => properties of next state

Another use of logic would be that you know something about the current state of the world and you know something about the operator that you're considering doing. You wonder what will happen if you take that action. You have a formal description of what that action does in the world. You might want to take those things together and infer something about the next state of the world. So these are two kinds of inferences that an agent might want to do. We could come up with a lot of other ones, but those are two good examples to keep in mind.

Propositional Logic Syntax

In the book they start by talking about logic in the abstract. But, the abstraction of logic is impossibly abstract. But anyway, they talk about logic in the abstract and then they talk about propositional logic. So, we're just going to dive right into propositional logic, learn something about how that works, and then try to generalize later on.

We'll start by talking about the syntax of propositional logic. Syntax is what you're allowed to write on your paper.

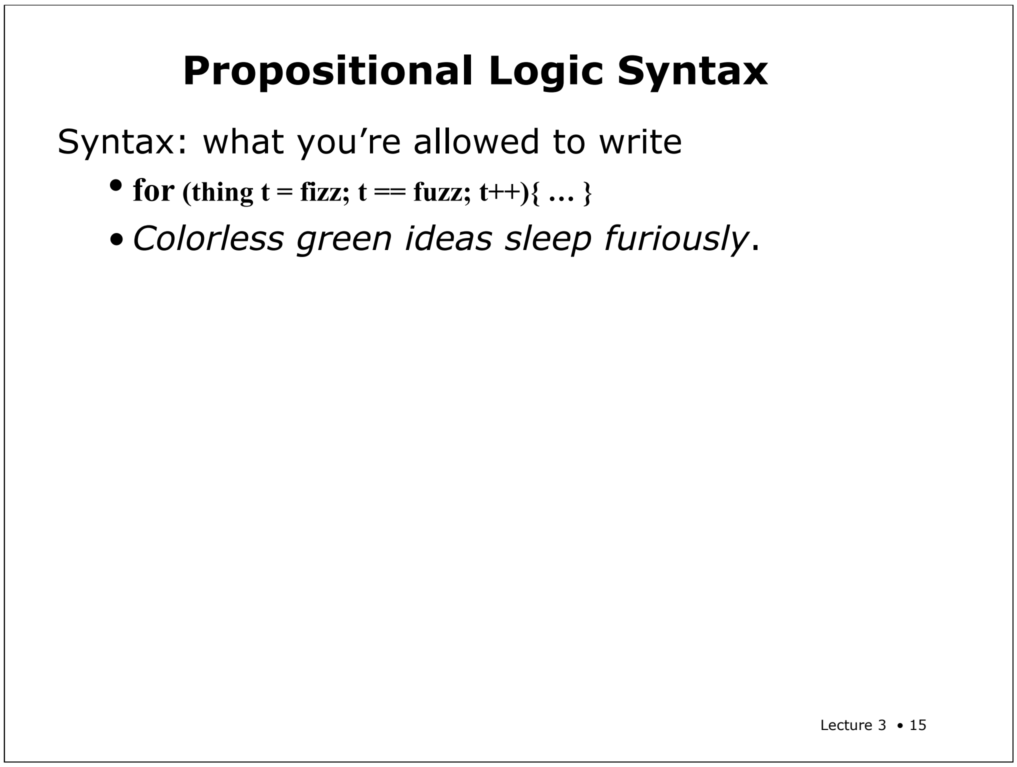
Propositional Logic Syntax

Syntax: what you're allowed to write

- **for (thing t = fizz; t == fuzz; t++){ … }**

You're all used to rules of syntax from programming languages, right? So, in Java you can write a for loop.

There are rules of syntax given by a formal grammar. They tell you there has to be a semicolon after fizz; that the parentheses have to match, and so on. You can't make random changes to the characters in your program and expect the compiler to be able to interpret it. So, the syntax is what symbols you're allowed to write down in what order. Not what they mean, not what computation they symbolize, but just what symbols you can write down.

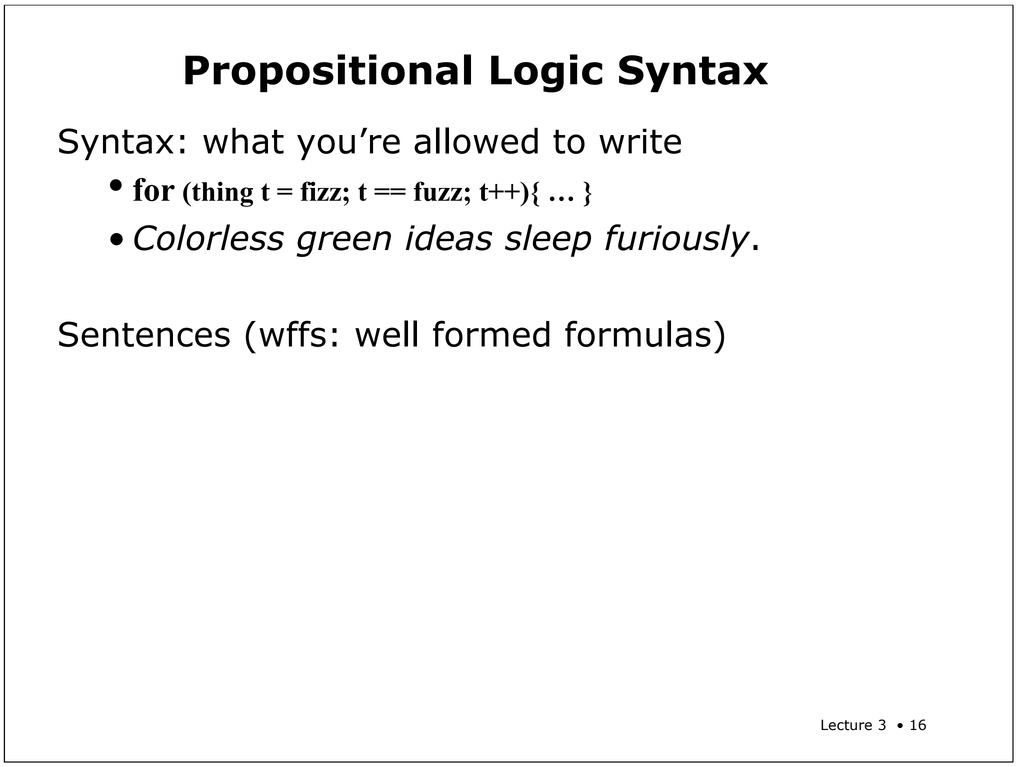


Propositional Logic Syntax

Syntax: what you're allowed to write

- **for (thing t = fizz; t == fuzz; t++){ … }**
- *Colorless green ideas sleep furiously*.

Another famous illustration of syntax is this one, due to the linguist Chomsky: "Colorless green ideas sleep furiously". The idea is that it doesn't mean anything really, but it's syntactically well-formed. It's got the nouns, the verbs, and the adjectives in the right place. If you scrambled the words up, you wouldn't get a sentence, right? You'd just get a string of words that didn't obey the rules of syntax. So, "furiously ideas green sleep colorless" is not OK.



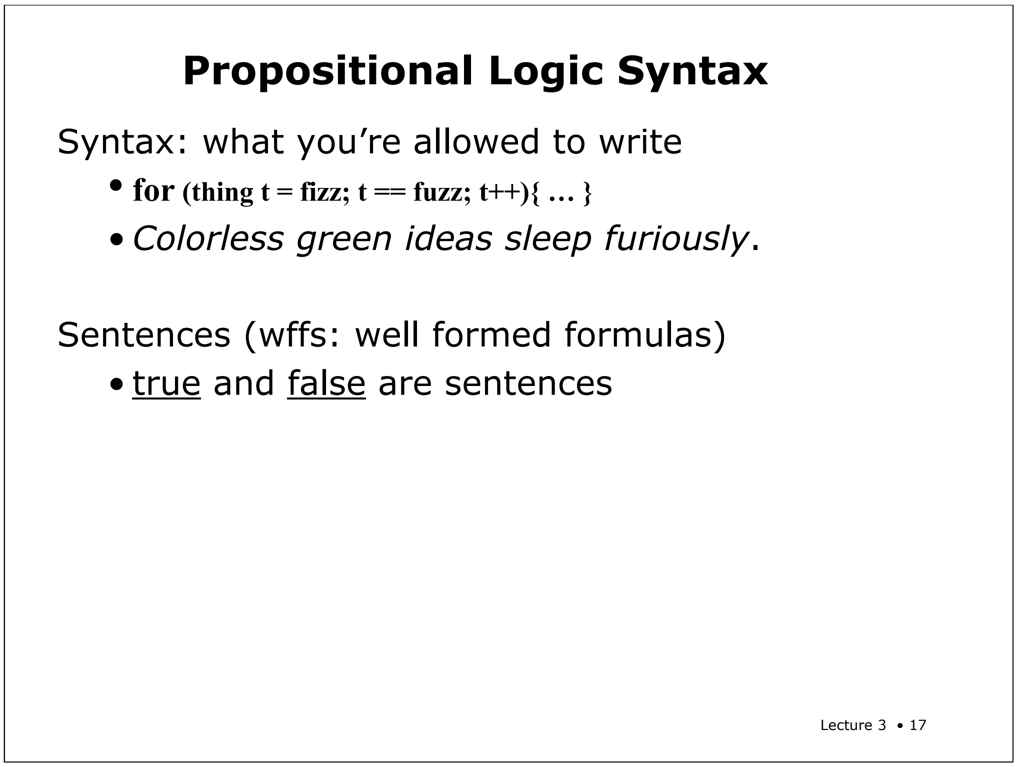
Propositional Logic Syntax

Syntax: what you’re allowed to write

- **for (thing t = fizz; t == fuzz; t++){ … }**
- *Colorless green ideas sleep furiously*.

Sentences (wffs: well formed formulas)

So let's define the syntax of propositional logic. We’ll call the legal things to write down "sentences". So if something is a sentence, it is a syntactically OK thing in our language. Sometimes sentences are called "WFFs" (which stands for “well-formed formulas” in other books).



Propositional Logic Syntax

Syntax: what you’re allowed to write

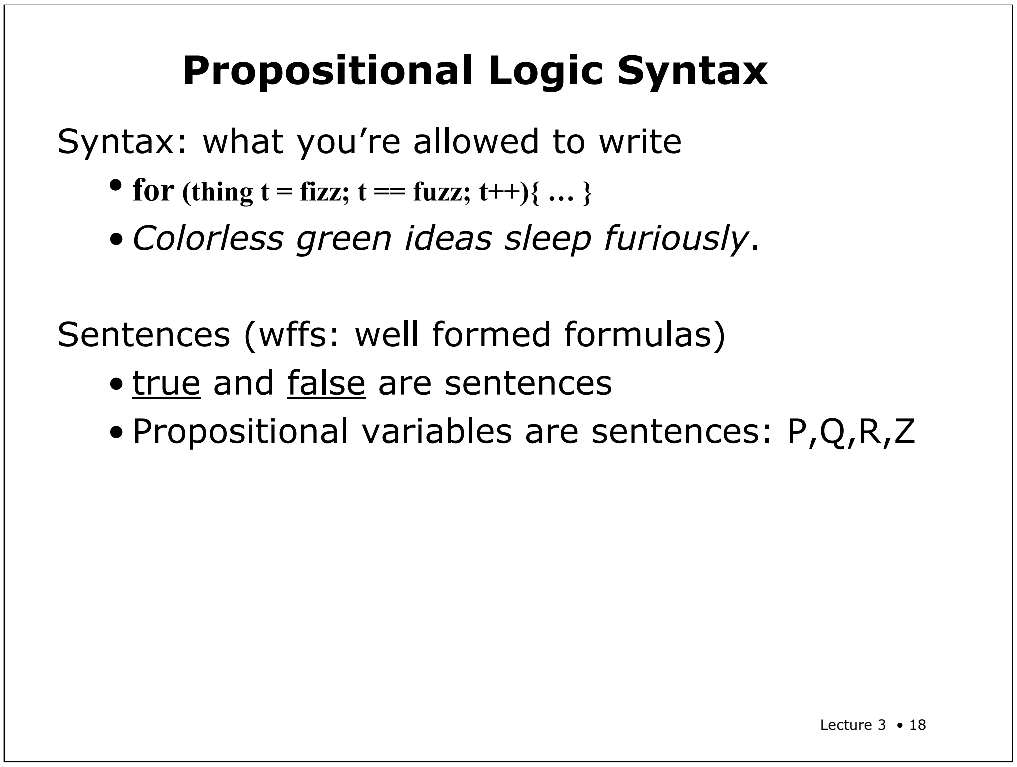
- **for (thing t = fizz; t == fuzz; t++){ … }**
- *Colorless green ideas sleep furiously*.

Sentences (wffs: well formed formulas)

- <u>true</u> and <u>false</u> are sentences

We're going to define the set of legal sentences recursively. So here are two base cases:

The words, "true" and "false", are sentences.



Propositional Logic Syntax

Syntax: what you're allowed to write

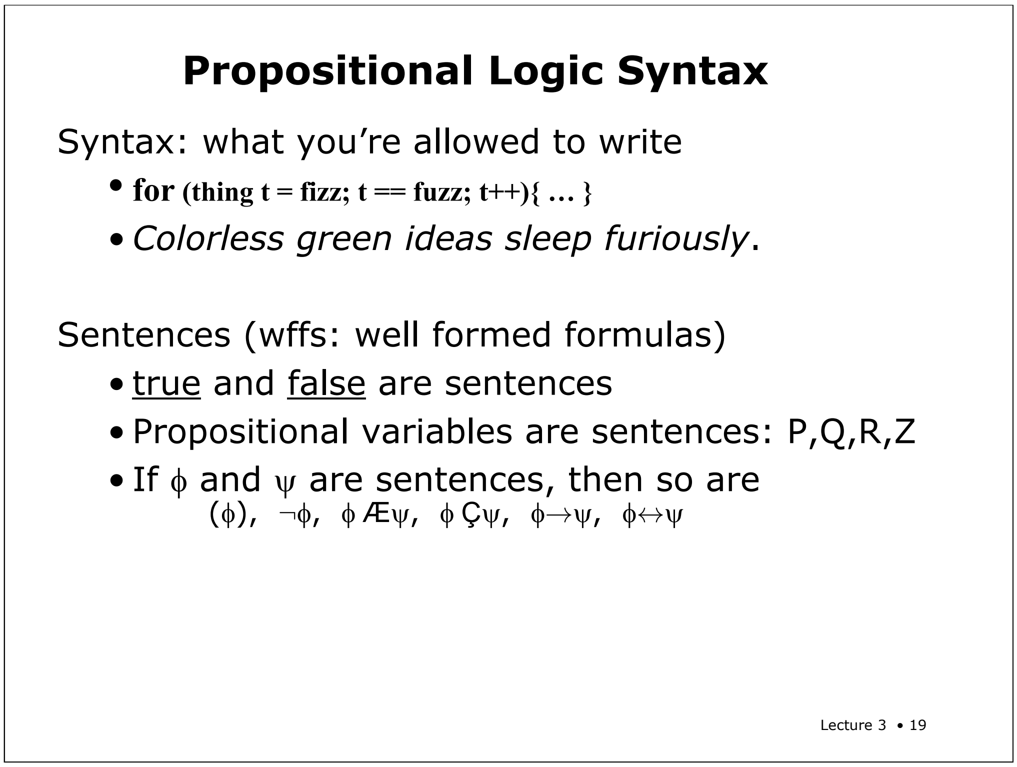
- **for (thing t = fizz; t == fuzz; t++){ … }**
- *Colorless green ideas sleep furiously*.

Sentences (wffs: well formed formulas)

- <u>true</u> and <u>false</u> are sentences
- Propositional variables are sentences: P,Q,R,Z

Propositional variables are sentences. I'll give you some examples. P, Q, R, Z.

We're not, for right now, defining a language that a computer is going to read. And so we don't have to be absolutely rigorous about what characters are allowed in the name of a variable. But there are going to be things called variables, and we'll just use uppercase letters for them. Those are sentences. It's OK to say "P" -- that's well-formed.



Propositional Logic Syntax

Syntax: what you're allowed to write

- **for (thing t = fizz; t == fuzz; t++){ … }**
- *Colorless green ideas sleep furiously*.

Sentences (wffs: well formed formulas)

- <u>true</u> and <u>false</u> are sentences
- Propositional variables are sentences: P,Q,R,Z
- If ϕ and ψ are sentences, then so are
  (ϕ), $\neg\phi$, ϕ Æψ, ϕ Çψ, $\phi\rightarrow\psi$, $\phi\leftrightarrow\psi$

Now, here's the recursive part. If \phi and \psi are sentences, then so are –

Wait! What, exactly, are \phi and \psi? They're called metavariables, and they range over expressions. This rule says that if \phi and \psi are things that you already know are sentences because of one of these rules, then you can make more sentences out of them.

\Phi with parentheses around it is a sentence.

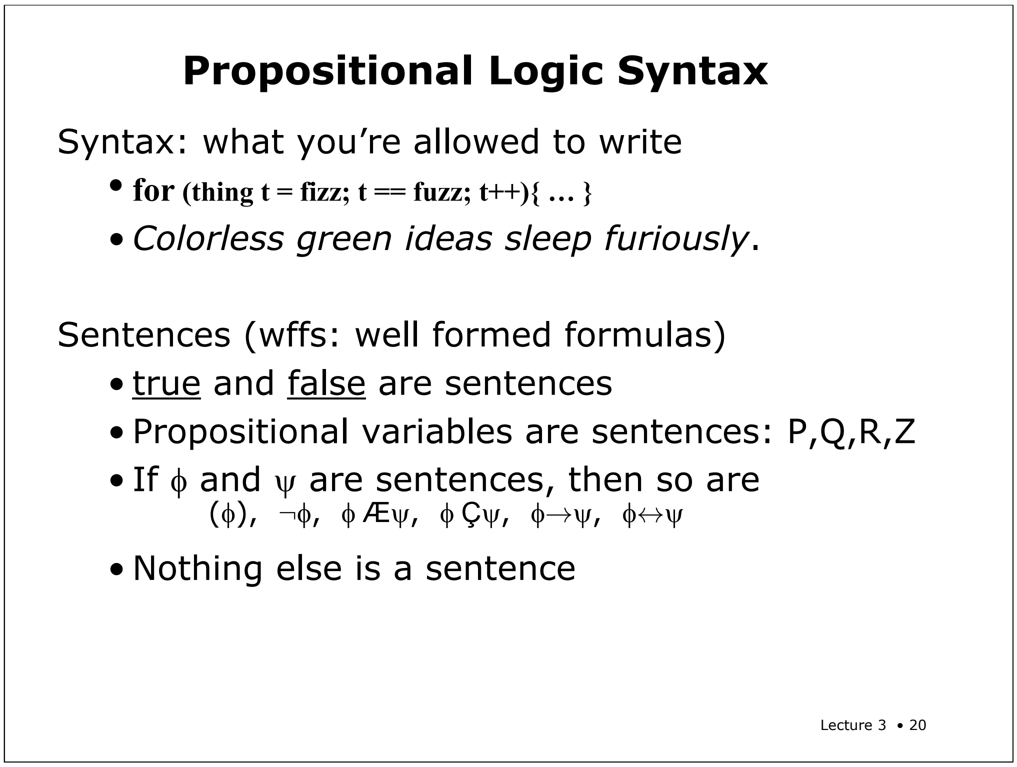
Not \Phi is a sentence (that little bent thing is our "not" symbol (but we're not really supposed to know that yet, because we're just doing syntax right now)).

\Phi "vee" \Psi is a sentence.

\Phi "wedge \Psi is a sentence.

\Phi "arrow" \Psi is a sentence.

\Phi "two-headed arrow" \Psi is a sentence.



Propositional Logic Syntax

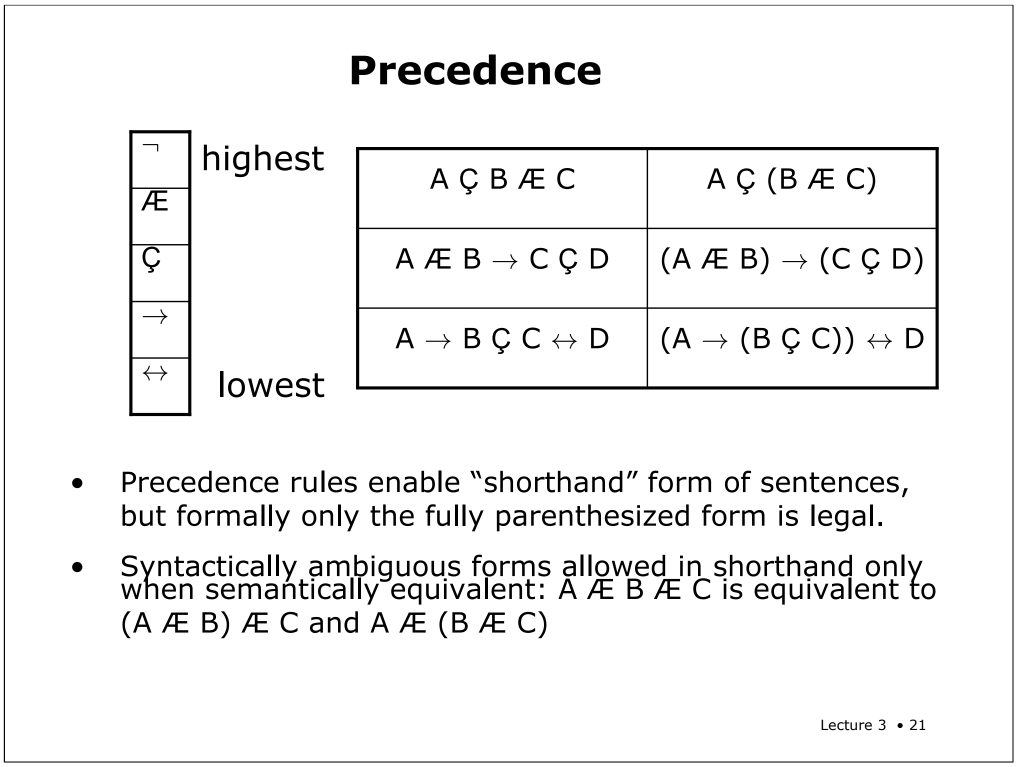
Syntax: what you're allowed to write

- **for (thing t = fizz; t == fuzz; t++){ … }**
- *Colorless green ideas sleep furiously*.

Sentences (wffs: well formed formulas)

- true and false are sentences
- Propositional variables are sentences: P,Q,R,Z
- If ϕ and ψ are sentences, then so are
  (ϕ), $\neg\phi$, ϕ Æψ, ϕ Çψ, $\phi\rightarrow\psi$, $\phi\leftrightarrow\psi$
- Nothing else is a sentence

And there's one more part of the definition, which says nothing else is a sentence. OK. That's the syntax of the language.



Precedence

¬	highest
Æ	
Ç	
→	
↔	lowest

A Ç B Æ C	A Ç (B Æ C)
A Æ B → C Ç D	(A Æ B) → (C Ç D)
A → B Ç C ↔ D	(A → (B Ç C)) ↔ D

- Precedence rules enable “shorthand” form of sentences, but formally only the fully parenthesized form is legal.
- Syntactically ambiguous forms allowed in shorthand only when semantically equivalent: A Æ B Æ C is equivalent to (A Æ B) Æ C and A Æ (B Æ C)

There’s actually one more issue we have to sort out. Precedence of the operations.

If we were being really careful, we’d require you to put parentheses around each new sentence that you made out of component sentences using negation, vee, wedge, or arrow. But it starts getting kind of ugly if we do that.

So, we allow you to leave out some of the parentheses, but then we need rules to figure out where the implicit parentheses really are. Those are precedence rules. Just as in arithmetic, we say that multiplication binds tighter than addition, we have similar rules in logic.

So, to add the parentheses to a sentence, you start with the highest precedence operator, which is negation. For every negation, you’d add an open paren in front of the negation sign and a close parenthesis after the next whole expression. This is exactly how minus behaves in arithmetic.

The next highest operator is wedge, which behaves like multiplication in arithmetic.

Next is vee, which behaves like addition in arithmetic.

Logic has two more operators, with weaker precedence. Next comes single arrow, and last is double arrow.

Also, wedge, vee, single arrow, and double arrow are associative.

Recitation Exercises: Part 1

Which of these are legal sentences?

$$P \rightarrow Q \rightarrow R$$

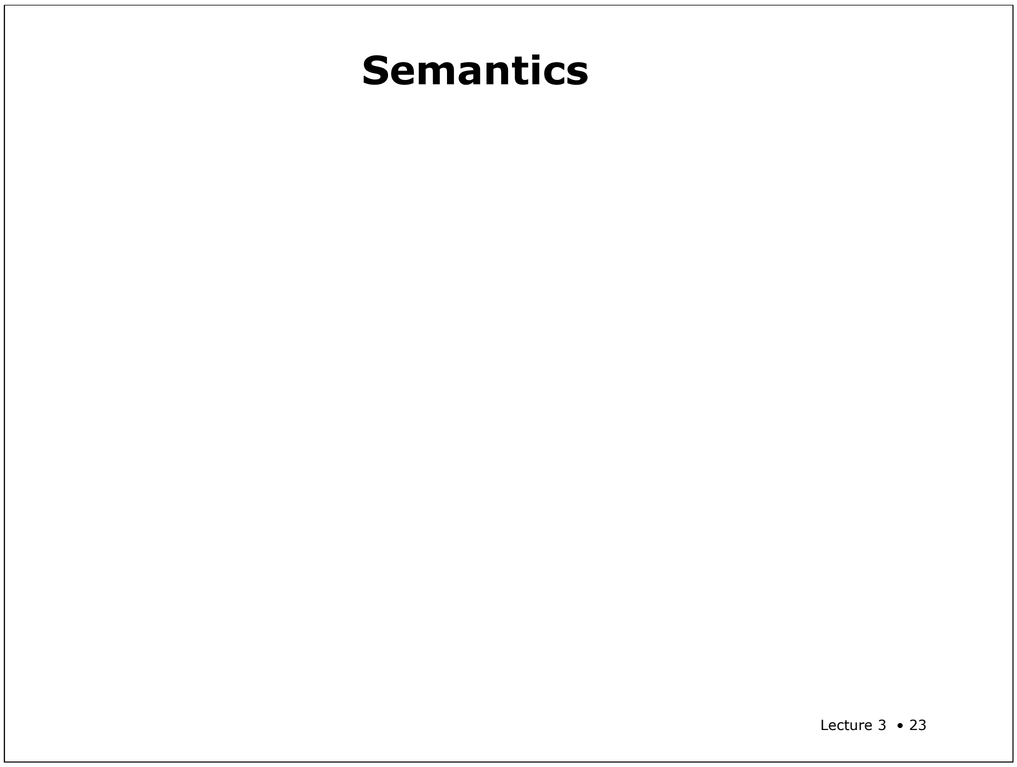
$$P, R \rightarrow Q$$

$$A \wedge (B \vee C \vee \neg D) \leftrightarrow \neg\neg Z$$

$$\neg P(Q)$$

Give fully parenthesized expressions for the legal sentences. (If there is more than one solution, just pick any one).

Here are some simple exercises. Please stop and do them now before going on with the rest of the lecture.



Semantics

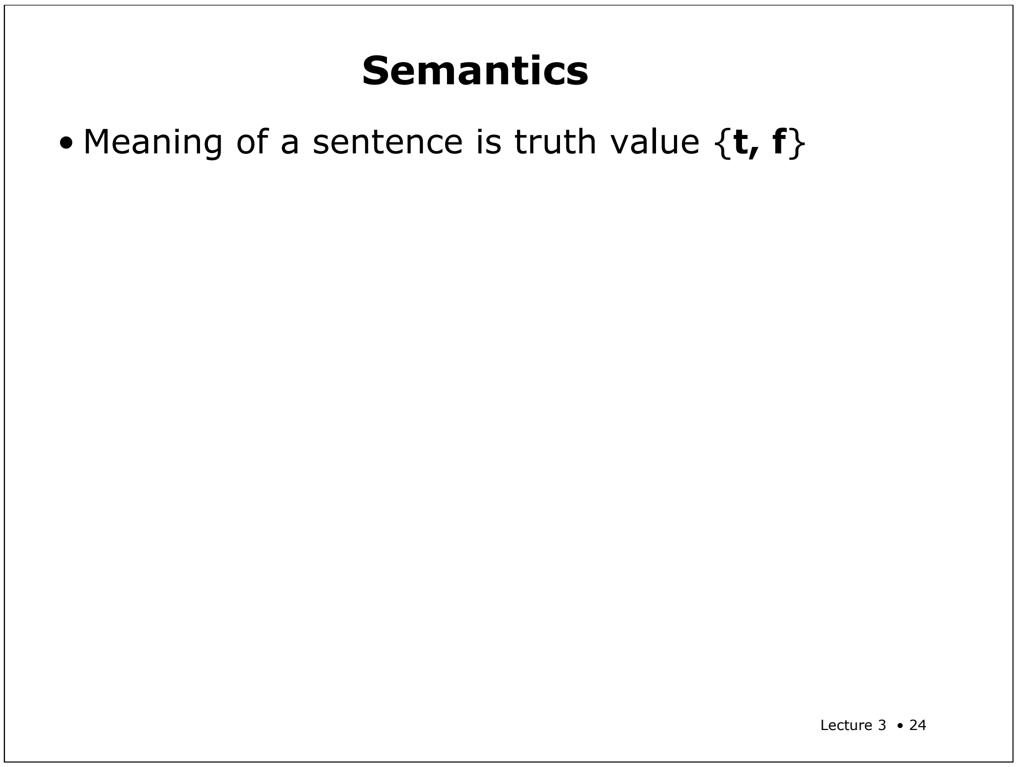
Lecture 3 • 23

So let's talk about semantics. The semantics of a sentence is its meaning. What does it say about the world?

We could just write symbols on the board and play with them all day long, and it could be fun, it could be like doing puzzles. But ultimately the reason that we want to be doing something with these kinds of logical sentences is because they somehow say something about the world. And it's really important to be clear about the connections between the things that we write on the board and what we think of them as meaning in the world, what they stand for.

And it's going to be something different every day. I remember once when I was a little kid I was on the school bus. And somebody's big sister or brother had started taking algebra and this kid told me, "You know what? My big sister's taking algebra and A equals 3!"

The reason that sounds so silly is that A is a variable. Our variables are going to be the same. They'll have different interpretations in different situations. So, in our study of logic, we're not going to assign particular values or meanings to the variables; rather, we're going to study the general properties of symbols and their potential meanings.

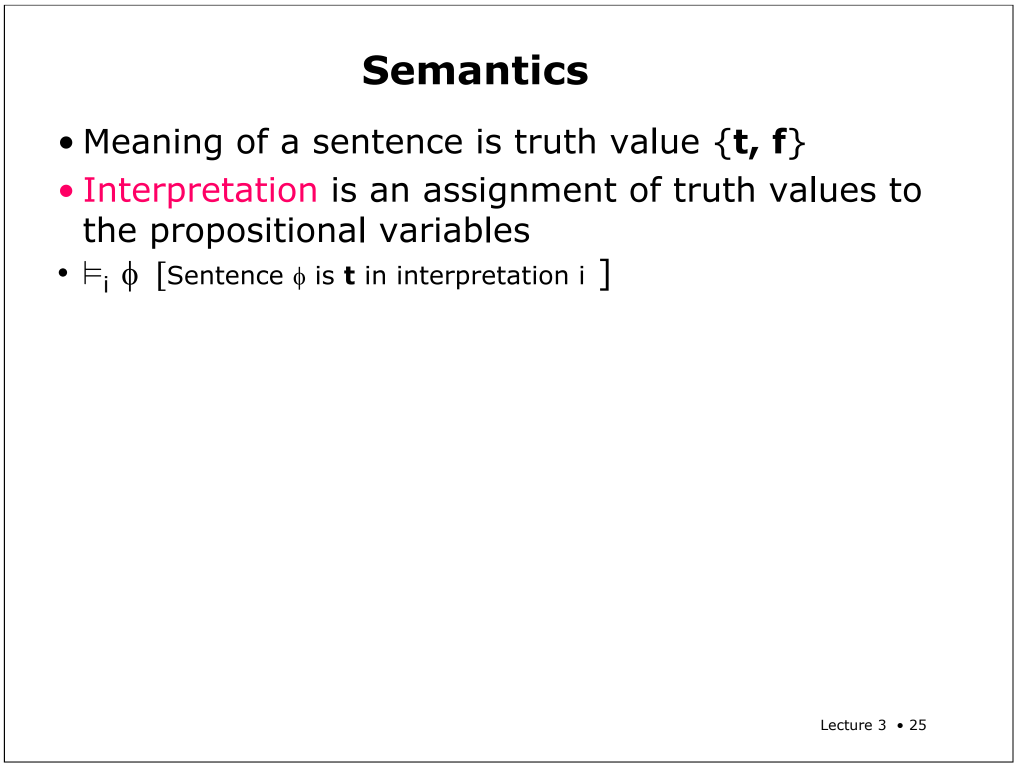


Semantics

- Meaning of a sentence is truth value {**t, f**}

Ultimately, the meaning of every sentence, in a situation, will be a truth value, T or F. Just as, in high-school algebra, the meaning of every expression is a numeric value.

Note that there's already a really important difference between underlined true and false, which are syntactic entities that we can write on the board, and the truth values T and F which stand for the abstract philosophical ideals of truth and falsity.



Semantics

- Meaning of a sentence is truth value {**t, f**}
- Interpretation is an assignment of truth values to the propositional variables
- $\models_i \phi$ [Sentence ϕ is **t** in interpretation i]

How could we decide whether A wedge B wedge C is true or not? Well, it has to do with what A and B and C stand for in the world. What A and B and C stand for in the world will be given by an object called an “interpretation”. An interpretation -- and I'm going to depart a little bit from Russell and Norvig, who tell a slightly more complicated story -- is an assignment of truth values to the propositional variables.

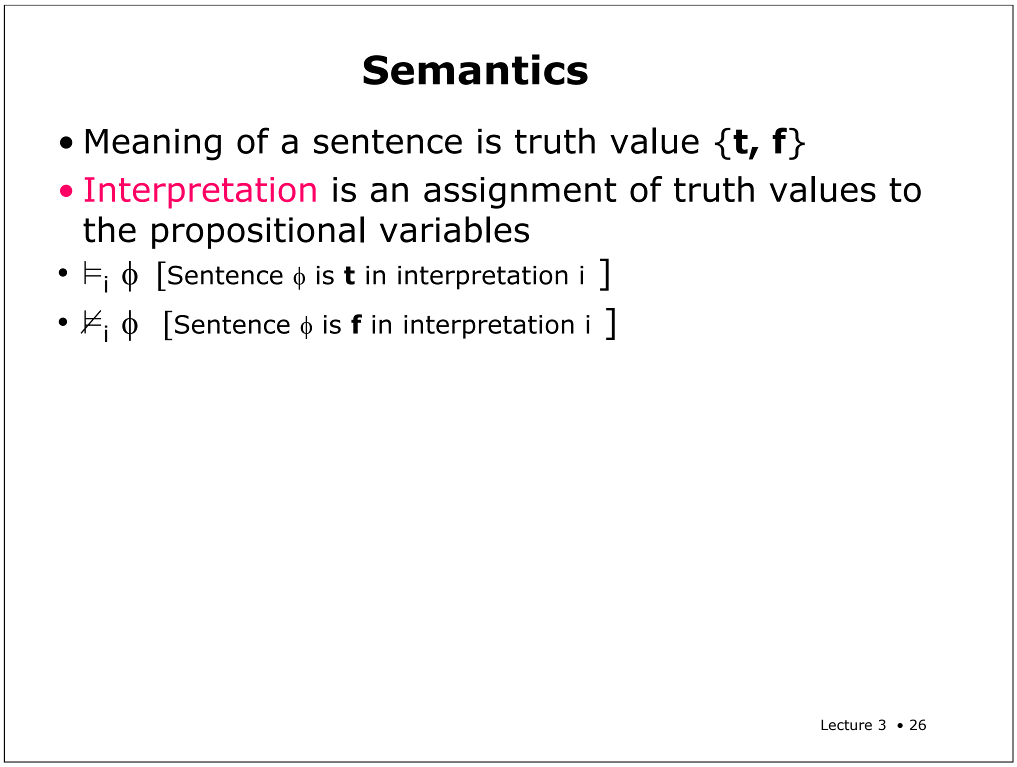
An interpretation is an assignment of truth values to the variables. You can think of it as a possible way the world could be.

So if our set of variables is P, Q, R, and V, then P true, Q false, R true, V true, that would be an interpretation.

So then, given an interpretation, we can ask the question, is this sentence true in that interpretation? We will write “turnstile sub I of Phi” (turnstile is the name logicians give to the symbol with one vertical bar and then two horizontal bars coming out to the right) to mean “sentence \Phi is true in interpretation I”. The turnstile symbol is not part of our language. It's part of the way logicians write things on the board when they're talking about what they're doing.

This is a really important distinction. If you can think of our sentences like expressions in a programming language, then you can think of these expressions with turnstiles as being about whether programs work in a certain way or not.

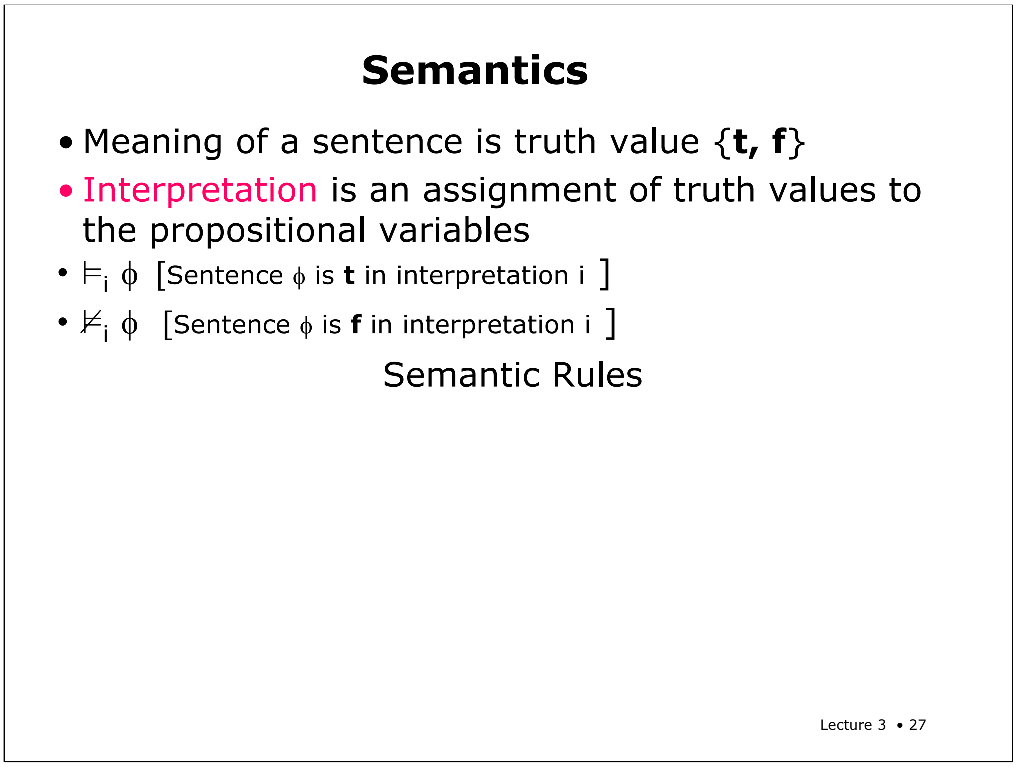
In order to even think about whether \Phi is true in interpretation I, \Phi **has** to be a sentence. If it’s not a well-formed sentence, then it doesn’t even make sense to ask whether it’s true or false.



Semantics

- Meaning of a sentence is truth value {**t, f**}
- Interpretation is an assignment of truth values to the propositional variables
- $\models_i \phi$ [Sentence ϕ is **t** in interpretation i]
- $\not\models_i \phi$ [Sentence ϕ is **f** in interpretation i]

Similarly, we'll use a turnstile with a slash through it to say that a sentence is **not** true in an interpretation. And since the meaning of every sentence is a truth value and there are only two truth values, then if a sentence \Phi is not true (does not have the truth value T) in an interpretation, then it has truth value F in that interpretation and we'll say it's false in that interpretation.



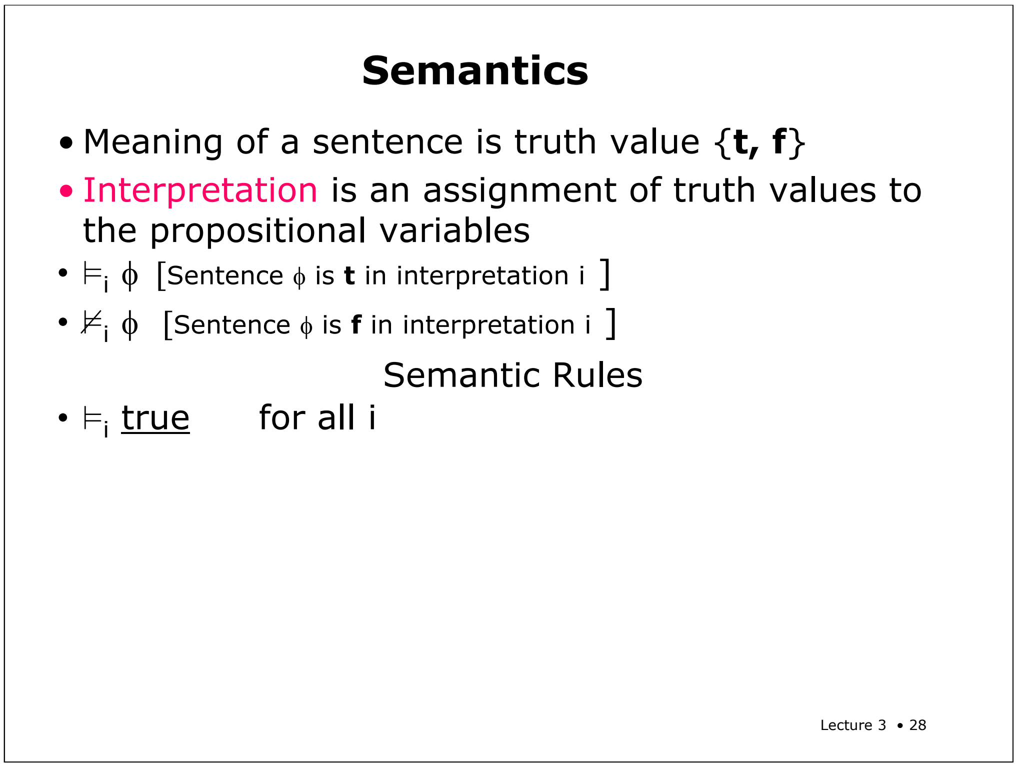
Semantics

- Meaning of a sentence is truth value {**t, f**}
- Interpretation is an assignment of truth values to the propositional variables
- $\models_i \phi$ [Sentence ϕ is **t** in interpretation i]
- $\not\models_i \phi$ [Sentence ϕ is **f** in interpretation i]

Semantic Rules

So now we can write down the rules of the semantics. We can write down, on the board, when sentence Phi is true in interpretation I.

I'm going to write the semantics down in a way that's parallel to the way we specified the syntax.



Semantics

- Meaning of a sentence is truth value {**t, f**}
- Interpretation is an assignment of truth values to the propositional variables
- $\models_i \phi$ [Sentence ϕ is **t** in interpretation i]
- $\not\models_i \phi$ [Sentence ϕ is **f** in interpretation i]

Semantic Rules

- $\models_i$ <u>true</u> for all i

First, the sentence consisting of the symbol "true" is true in all interpretations.

Semantics

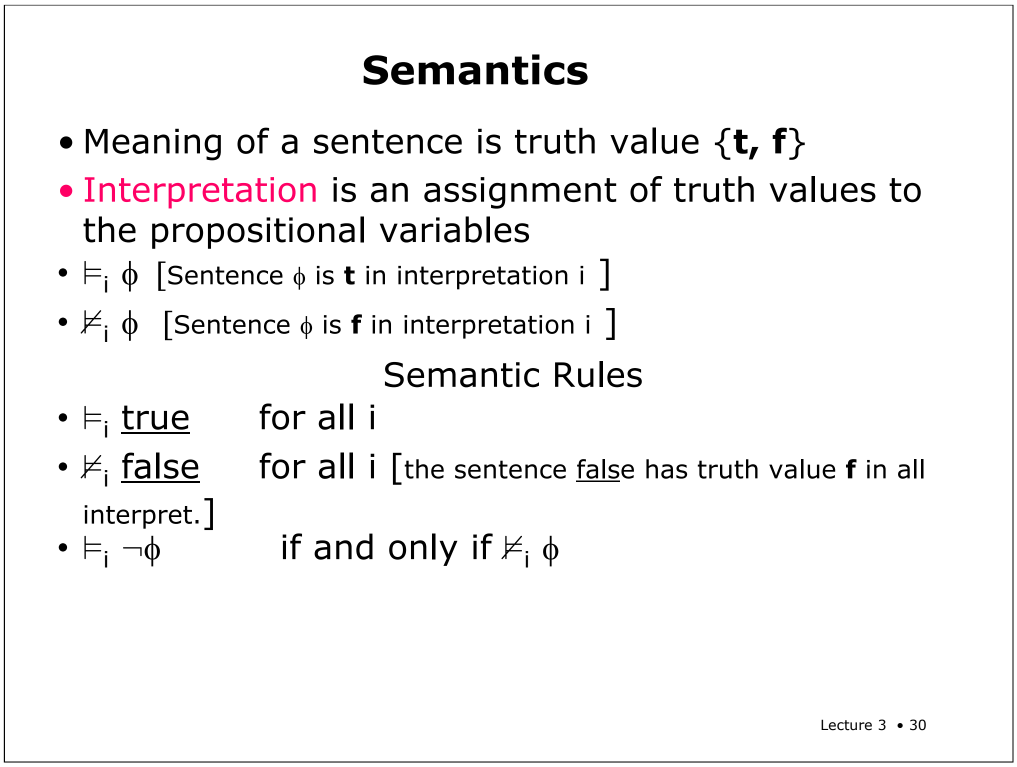
- Meaning of a sentence is truth value {**t, f**}
- Interpretation is an assignment of truth values to the propositional variables
- $\models_i \phi$ [Sentence ϕ is **t** in interpretation i]
- $\not\models_i \phi$ [Sentence ϕ is **f** in interpretation i]

Semantic Rules

- $\models_i$ <u>true</u> for all i
- $\not\models_i$ <u>false</u> for all i [the sentence <u>fals</u>e has truth value **f** in all interpret.]

The sentence consisting of a symbol "false" has truth value "F" in all interpretations.

All right, now we can do the connectives. We'll leave out the parentheses. The truth value of a sentence with top-level parentheses is the same as the truth value of the sentence with the parentheses removed.



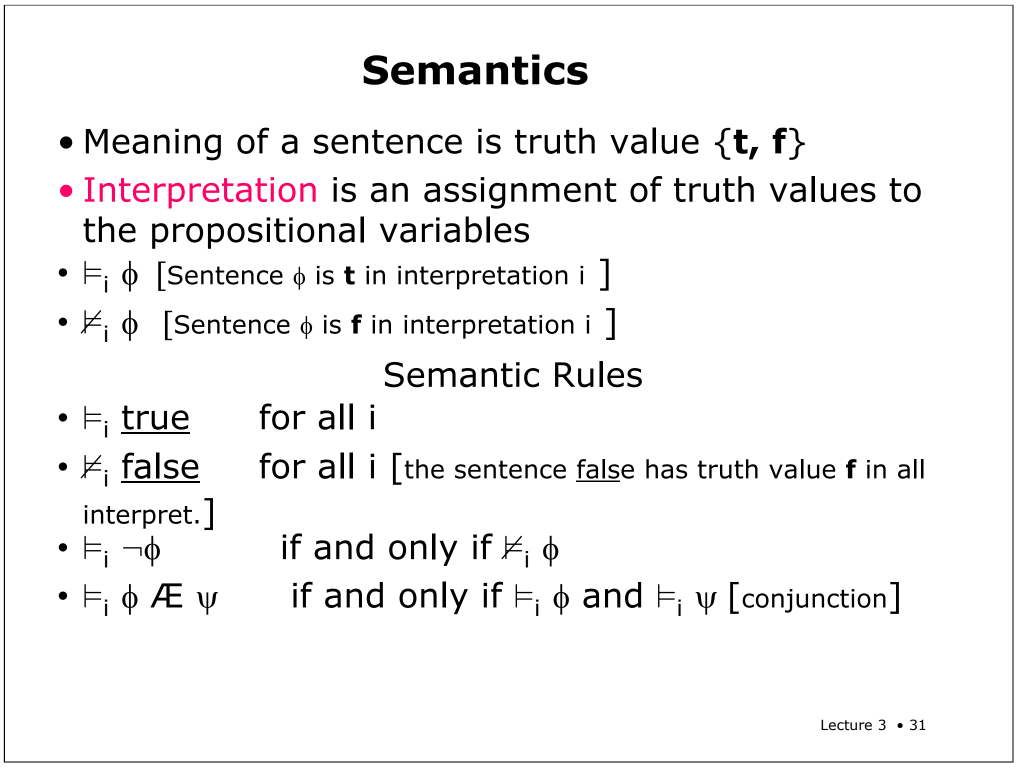
Semantics

- Meaning of a sentence is truth value {**t, f**}
- Interpretation is an assignment of truth values to the propositional variables
- $\models_i \phi$ [Sentence ϕ is **t** in interpretation i]
- $\not\models_i \phi$ [Sentence ϕ is **f** in interpretation i]

Semantic Rules

- $\models_i$ <u>true</u> for all i
- $\not\models_i$ <u>false</u> for all i [the sentence <u>false</u> has truth value **f** in all interpret.]
- $\models_i \neg\phi$ if and only if $\not\models_i \phi$

Now, let’s think about the negation sign. When is \negation \Phi true in an interpretation I? Whenever \Phi is false in that interpretation.



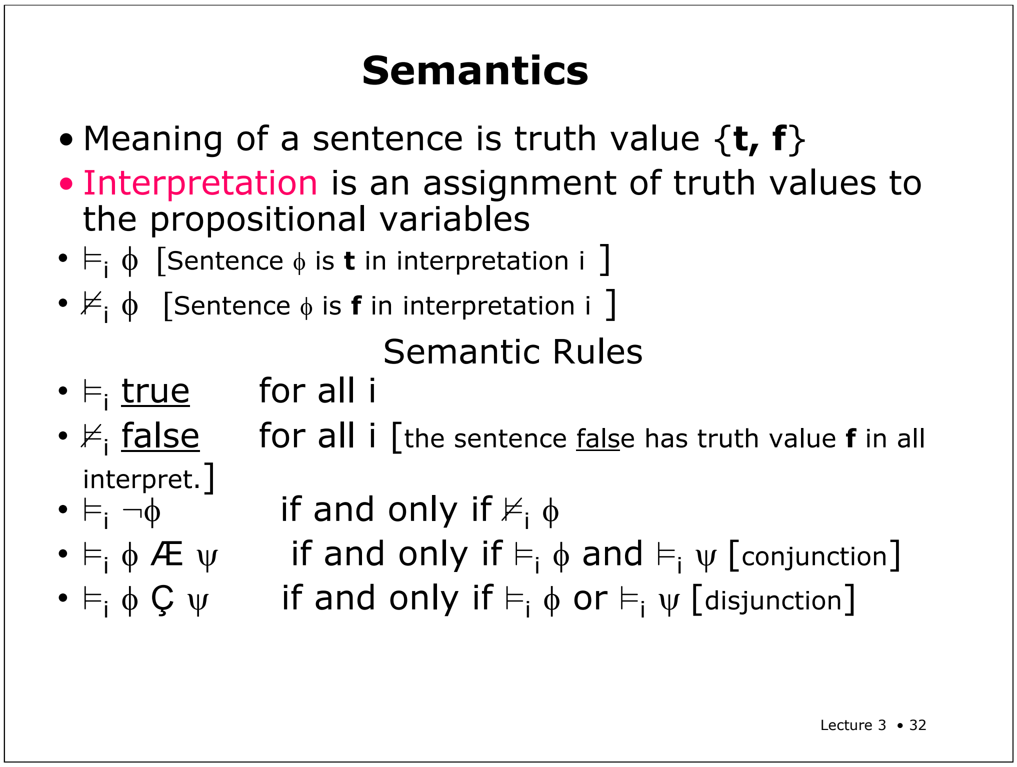
Semantics

- Meaning of a sentence is truth value {**t, f**}
- Interpretation is an assignment of truth values to the propositional variables
- $\models_i \phi$ [Sentence ϕ is **t** in interpretation i]
- $\not\models_i \phi$ [Sentence ϕ is **f** in interpretation i]

Semantic Rules

- $\models_i$ <u>true</u> for all i
- $\not\models_i$ <u>false</u> for all i [the sentence <u>fals</u>e has truth value **f** in all interpret.]
- $\models_i \neg\phi$ if and only if $\not\models_i \phi$
- $\models_i \phi$ Æ ψ if and only if $\models_i \phi$ and $\models_i \psi$ [conjunction]

When is Phi wedge Psi true in an interpretation I? Whenever both Phi **and** Psi are true in I. This is called “conjunction”. And we’ll start calling that symbol “and” instead of “wedge”, now that we know what it means.



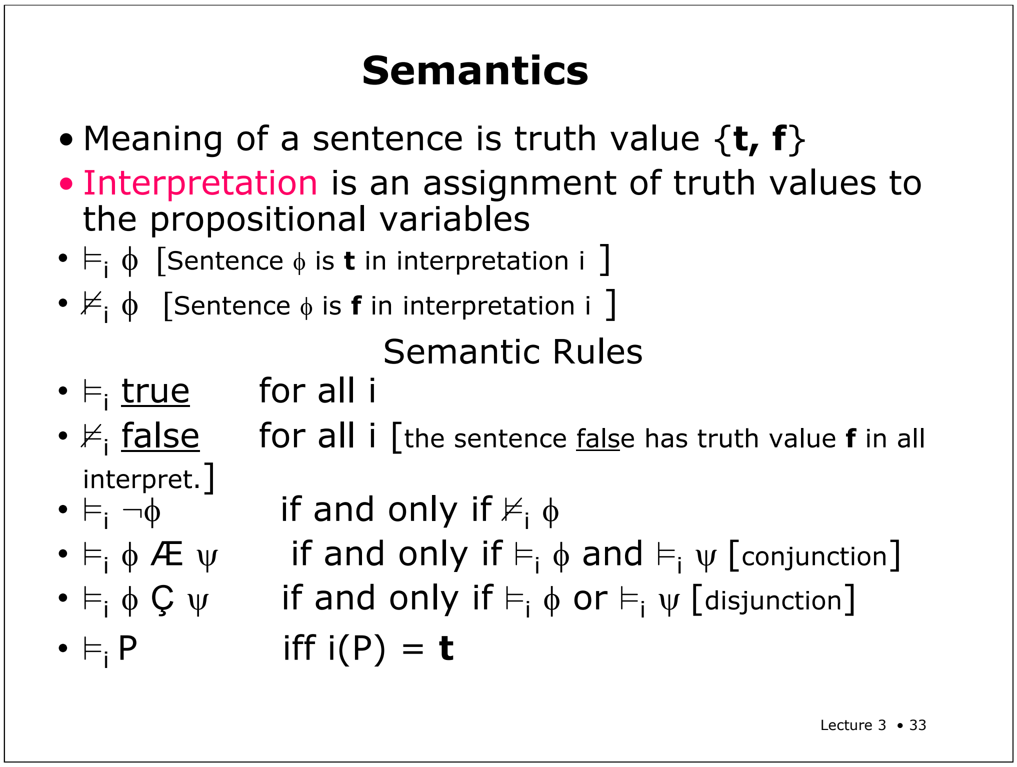
Semantics

- Meaning of a sentence is truth value {**t, f**}
- Interpretation is an assignment of truth values to the propositional variables
- $\models_i \phi$ [Sentence ϕ is **t** in interpretation i]
- $\not\models_i \phi$ [Sentence ϕ is **f** in interpretation i]

Semantic Rules

- $\models_i$ <u>true</u> for all i
- $\not\models_i$ <u>false</u> for all i [the sentence <u>fals</u>e has truth value **f** in all interpret.]
- $\models_i \neg\phi$ if and only if $\not\models_i \phi$
- $\models_i \phi$ Æ ψ if and only if $\models_i \phi$ and $\models_i \psi$ [conjunction]
- $\models_i \phi$ Ç ψ if and only if $\models_i \phi$ or $\models_i \psi$ [disjunction]

When is Phi vee Psi true in an interpretation I? Whenever **either** Phi or Psi is true in I. This is called “disjuction”, and we’ll call the vee symbol “or”. It is not an exclusive or; so that if both Phi and Psi are true in I, then Phi vee Psi is also true in I.



Semantics

- Meaning of a sentence is truth value {**t, f**}
- Interpretation is an assignment of truth values to the propositional variables
- $\models_i \phi$ [Sentence ϕ is **t** in interpretation i]
- $\not\models_i \phi$ [Sentence ϕ is **f** in interpretation i]

Semantic Rules

- $\models_i$ <u>true</u> for all i
- $\not\models_i$ <u>false</u> for all i [the sentence <u>false</u> has truth value **f** in all interpret.]
- $\models_i \neg\phi$ if and only if $\not\models_i \phi$
- $\models_i \phi$ Æ ψ if and only if $\models_i \phi$ and $\models_i \psi$ [conjunction]
- $\models_i \phi$ Ç ψ if and only if $\models_i \phi$ or $\models_i \psi$ [disjunction]
- $\models_i P$ iff i(P) = **t**

Lecture 3 • 33

Now we have one more clause. I'm going to do it by example. Imagine that we have a sentence P. P is one of our propositional variables. How do we know whether it is true in interpretation I?

Well, since I is a mapping from variables to truth values, I can simply look P up in I and return whatever truth value was assigned to P by I.

Some important shorthand

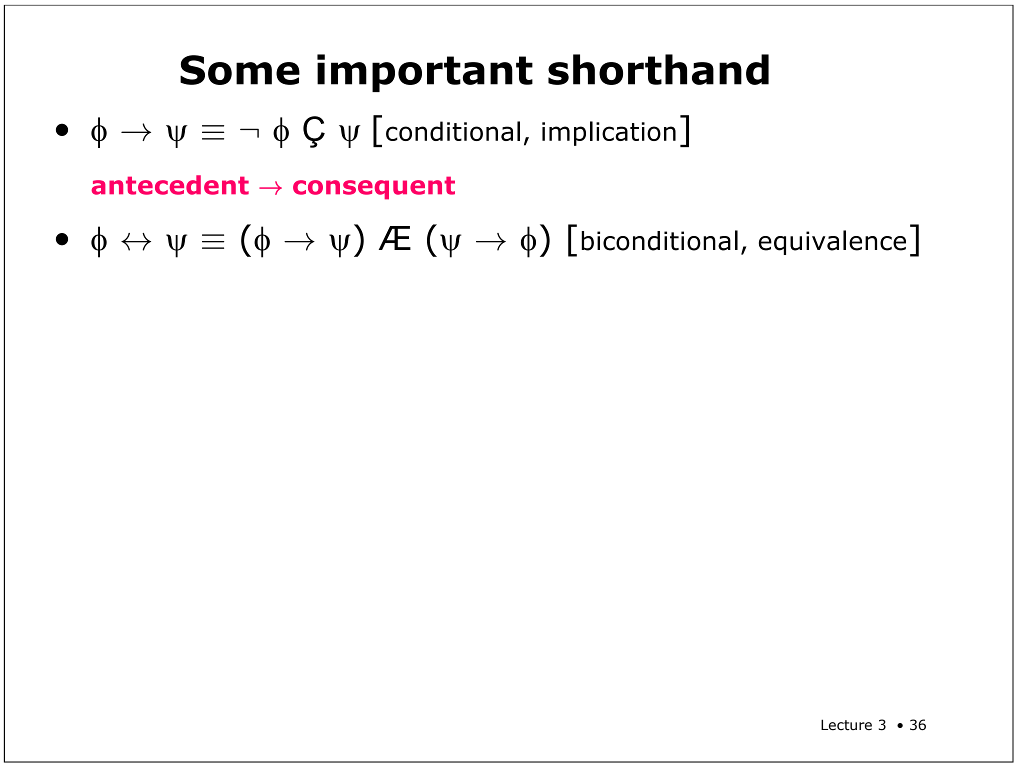
It seems like we left out the arrows in the semantic definitions of the previous slide. But the arrows are not strictly necessary; that is, it's going to turn out that you can say anything you want to without them, but they're a convenient shorthand. (In fact, you can also do without either "or" or "and", but we'll see that later).

Some important shorthand

- $\phi \rightarrow \psi \equiv \neg\, \phi$ Ç ψ [conditional, implication]

  antecedent → consequent

So, we can define Phi arrow Psi as being equivalent to not Phi or Psi. That is, no matter what Phi and Psi are, and in every interpretation, (Phi arrow Psi) will have the same truth value as (not Phi or Psi). We will now call this arrow relationship “implication”. We’ll say that Phi implies Psi. We may also call this a conditional expression: Psi is true if Phi is true. In such a statement, Phi is referred to as the antecedent and Psi as the consequent.



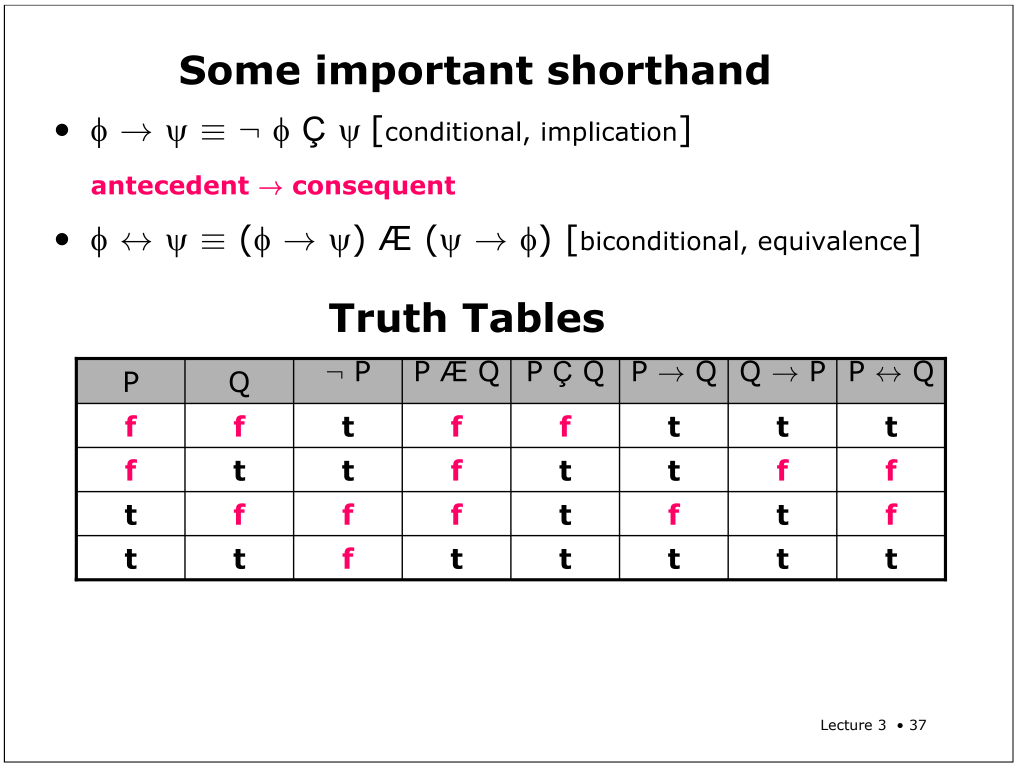
Some important shorthand

- ϕ → ψ ≡ ¬ ϕ Ç ψ [conditional, implication]

  antecedent → consequent

- ϕ ↔ ψ ≡ (ϕ → ψ) Æ (ψ → ϕ) [biconditional, equivalence]

Finally, the double arrow just means that we have single arrows going both ways. This is sometimes called a “bi-conditional” or “equivalence” statement. It means that in every interpretation, Phi and Psi have the same truth value.



Some important shorthand

- ϕ → ψ ≡ ¬ ϕ Ç ψ [conditional, implication]

  antecedent → consequent

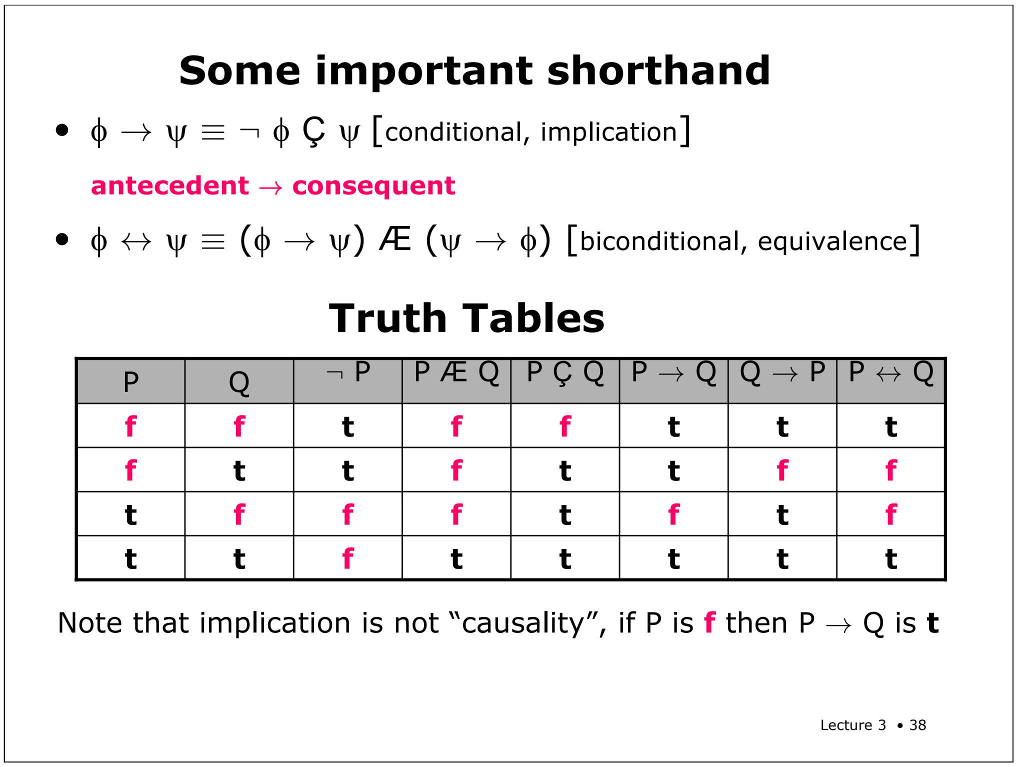
- ϕ ↔ ψ ≡ (ϕ → ψ) Æ (ψ → ϕ) [biconditional, equivalence]

Truth Tables

P	Q	¬ P	P Æ Q	P Ç Q	P → Q	Q → P	P ↔ Q
f	**f**	**t**	**f**	**f**	**t**	**t**	**t**
f	**t**	**t**	**f**	**t**	**t**	**f**	**f**
t	**f**	**f**	**f**	**t**	**f**	**t**	**f**
t	**t**	**f**	**t**	**t**	**t**	**t**	**t**

Just so you can see how all of these operators work, here are the truth tables. Consider a world with two propositional variables, P and Q. There are four possible interpretations in such a world (one for every combination of assignments to the variables; in general, in a world with n variables, there will be 2^n possible interpretations). Each row of the truth table corresponds to a possible interpretation, and we've filled in the values it assigns to P and Q in the first two columns.

Once we have chosen an interpretation (a row in the table), then the semantic rules tell us exactly what the truth value of every single legal sentence must be. Here we show the truth values for six different sentences made up from P and Q.



Some important shorthand

- $\phi \rightarrow \psi \equiv \neg \phi$ Ç ψ [conditional, implication]

  antecedent $\rightarrow$ consequent

- $\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi)$ Æ $(\psi \rightarrow \phi)$ [biconditional, equivalence]

Truth Tables

P	Q	¬ P	P Æ Q	P Ç Q	P → Q	Q → P	P ↔ Q
f	**f**	**t**	**f**	**f**	**t**	**t**	**t**
f	**t**	**t**	**f**	**t**	**t**	**f**	**f**
t	**f**	**f**	**f**	**t**	**f**	**t**	**f**
t	**t**	**f**	**t**	**t**	**t**	**t**	**t**

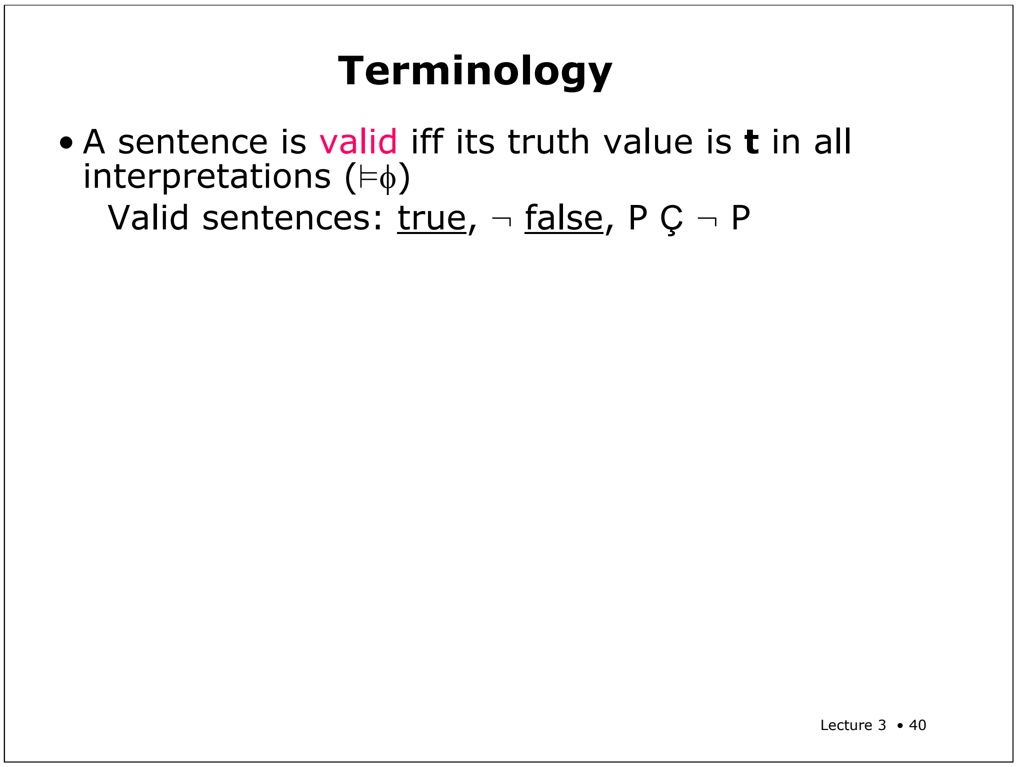
Note that implication is not "causality", if P is **f** then P → Q is **t**

Most of them are fairly obvious, but it's worth studying the truth table for implication fairly closely. In particular, note that (P implies Q) is true whenever P is false. You can see that this is reasonable by thinking about an English sentence like "If pigs can fly then …". Once you start with a false condition, you can finish with anything, and the sentence will be true.

Implication doesn't mean causes. It doesn't mean is related in any kind of way that you imagine, it is just a bare, formal definition of not P or Q.

Terminology

Now we'll define some terminology on this slide and the next, then do a lot of examples.



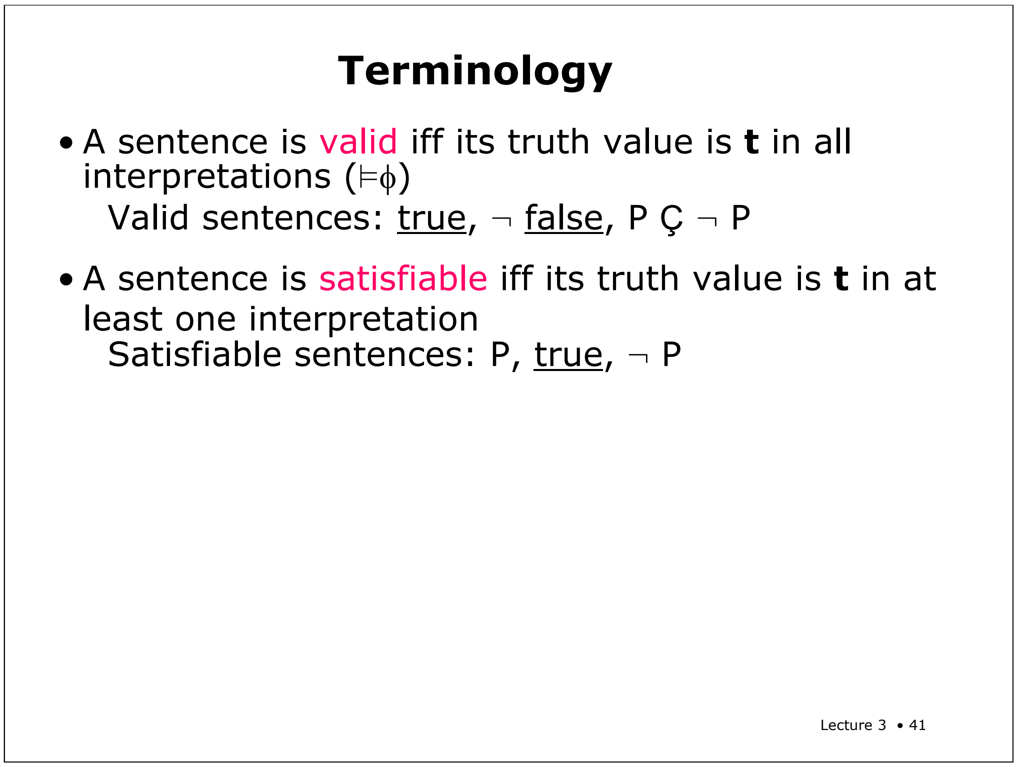
Terminology

- A sentence is valid iff its truth value is **t** in all interpretations (⊨ϕ)
  Valid sentences: true, ¬ false, P Ç ¬ P

Lecture 3 • 40

A sentence is valid if and only if it is true in all interpretations. OK. We have already seen one example of a valid sentence. What was it? True. Another one is “not false”.

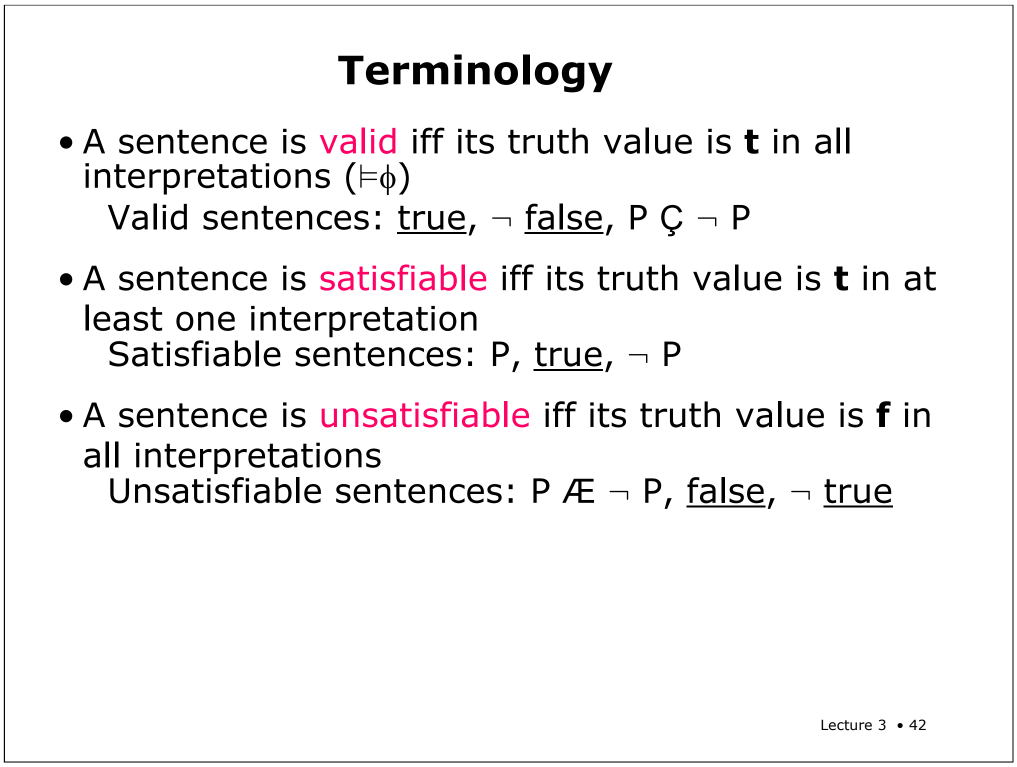
A more interesting one is “P or not P”. No matter what truth value is assigned to P by the interpretation, “P or not P” is true.



Terminology

- A sentence is valid iff its truth value is **t** in all interpretations (⊨ϕ)
  Valid sentences: <u>true</u>, ¬ <u>false</u>, P Ç ¬ P
- A sentence is satisfiable iff its truth value is **t** in at least one interpretation
  Satisfiable sentences: P, <u>true</u>, ¬ P

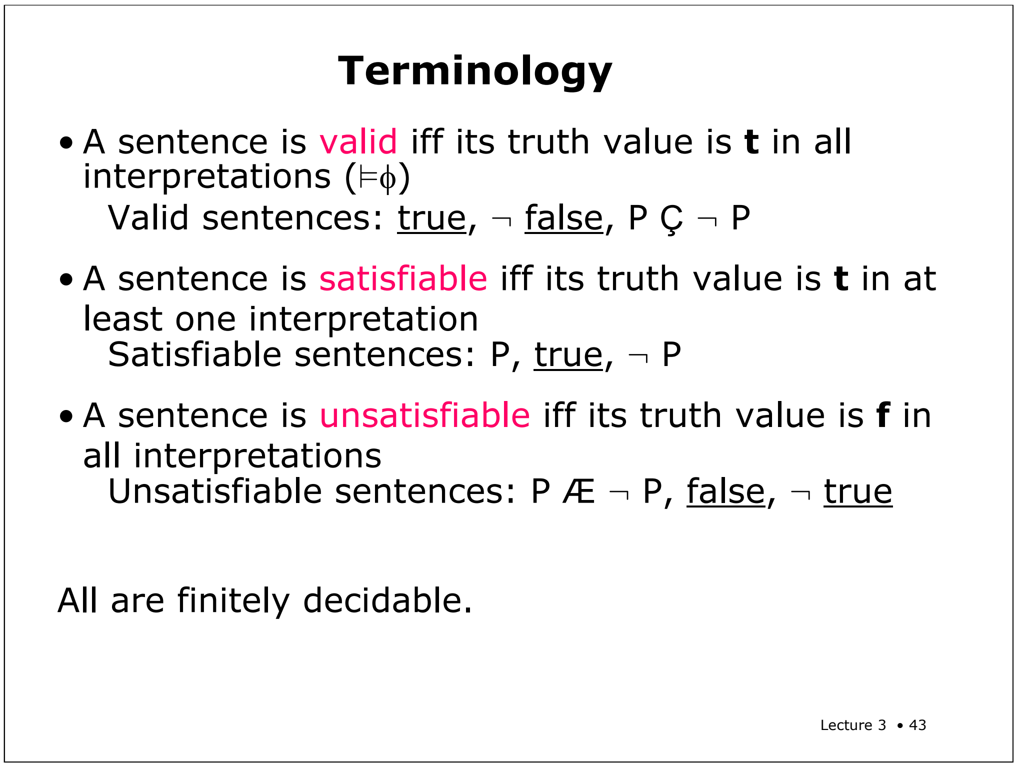
A sentence is satisfiable if and only if it's true in at least one interpretation. The sentence P is satisfiable. The sentence True is satisfiable. Not P is satisfiable.



Terminology

- A sentence is valid iff its truth value is **t** in all interpretations (⊨φ)
  Valid sentences: true, ¬ false, P Ç ¬ P
- A sentence is satisfiable iff its truth value is **t** in at least one interpretation
  Satisfiable sentences: P, true, ¬ P
- A sentence is unsatisfiable iff its truth value is **f** in all interpretations
  Unsatisfiable sentences: P Æ ¬ P, false, ¬ true

A sentence is unsatisfiable if and only if it’s false in every interpretation. Some unsatisfiable sentences are: false, not true, P and not P.



Terminology

- A sentence is valid iff its truth value is **t** in all interpretations (⊨ϕ)
  Valid sentences: true, ¬ false, P Ç ¬ P
- A sentence is satisfiable iff its truth value is **t** in at least one interpretation
  Satisfiable sentences: P, true, ¬ P
- A sentence is unsatisfiable iff its truth value is **f** in all interpretations
  Unsatisfiable sentences: P Æ ¬ P, false, ¬ true

All are finitely decidable.

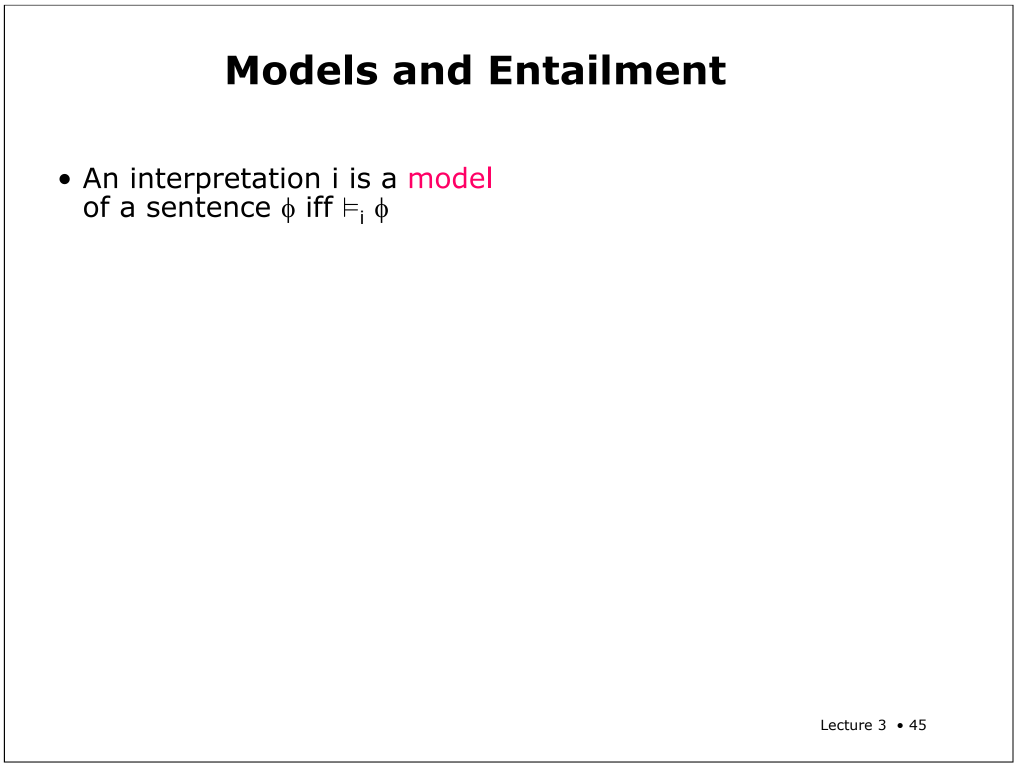
We can use the method of truth tables to check these things, right? If I wanted to know if a particular sentence was valid, or if I wanted to know if it was satisfiable or unsatisfiable. I could just make a truth table. Write down all the interpretations, figure out the value of the sentence in each interpretation, and if they're all true, it's valid. If they're all false, it's unsatisfiable. If it's somewhere in between, it's satisfiable. So there's a way; there's just a completely dopey, tedious, mechanical way to figure out if a sentence is one of these things. That's not true in all logics. This is a useful, special property of propositional logic. It might take you a lot of time, but it's a finite amount of time and you can decide any of these questions.

Models and Entailment

So now, another way to think about what's going on here involves a kind of complicated diagram.

We talked about the relationship between sentences and interpretations, right? That's the semantics. And you can really think of a sentence as standing for, or as meaning a set of interpretations, right? There is some set of interpretations that make this sentence true.

(I really apologize for the amount of terminology. It’s rare that we'll have so much terminology, but I think I should do it anyway, just because you'll run into these words.)



Models and Entailment

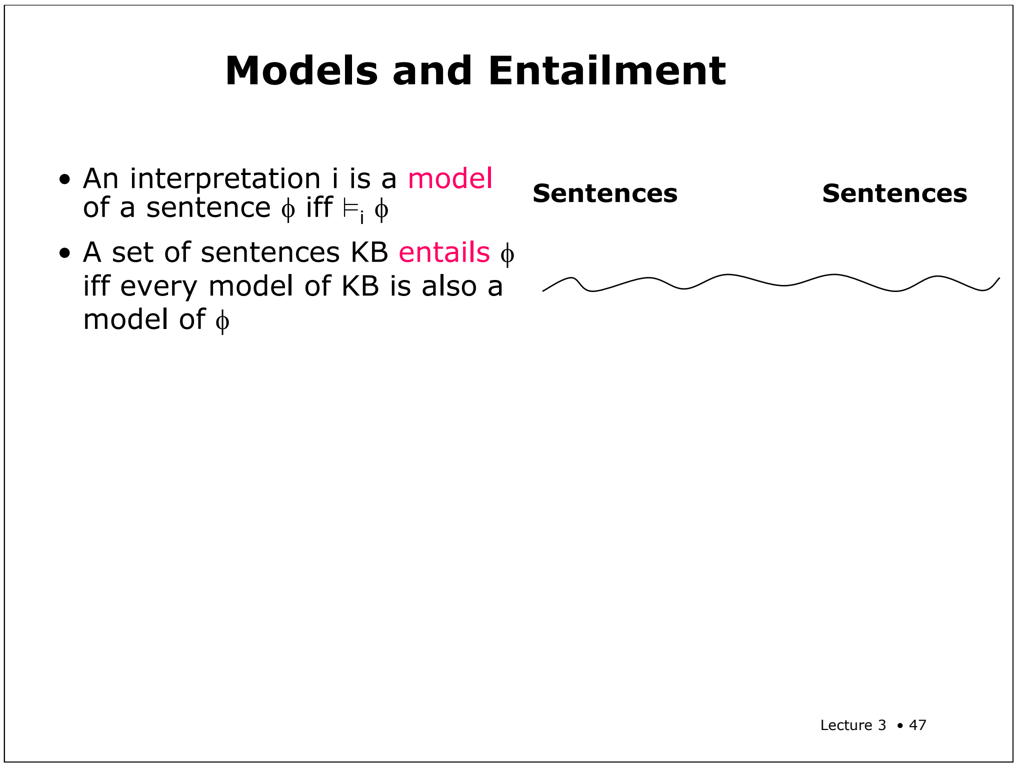
- An interpretation i is a model of a sentence ϕ iff $\models_i \phi$

So we'll say an interpretation is a **model** of a sentence if the sentence is true in the interpretation (this is a way to attach a word to the turnstyle character). .

Models and Entailment

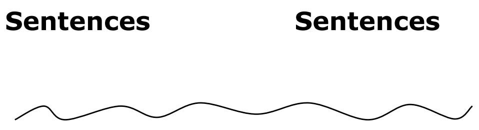
- An interpretation i is a model of a sentence ϕ iff $\models_i \phi$
- A set of sentences KB entails ϕ iff every model of KB is also a model of ϕ

Now consider a set of sentences – we'll call it KB for Knowledge Base. We'll say KB **entails** \Phi if and only if every model of KB is also a model of \Phi.



Models and Entailment

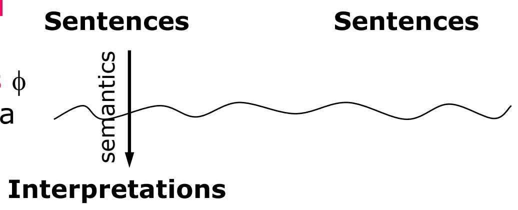
- An interpretation i is a model of a sentence ϕ iff $\models_i \phi$
- A set of sentences KB entails ϕ iff every model of KB is also a model of ϕ



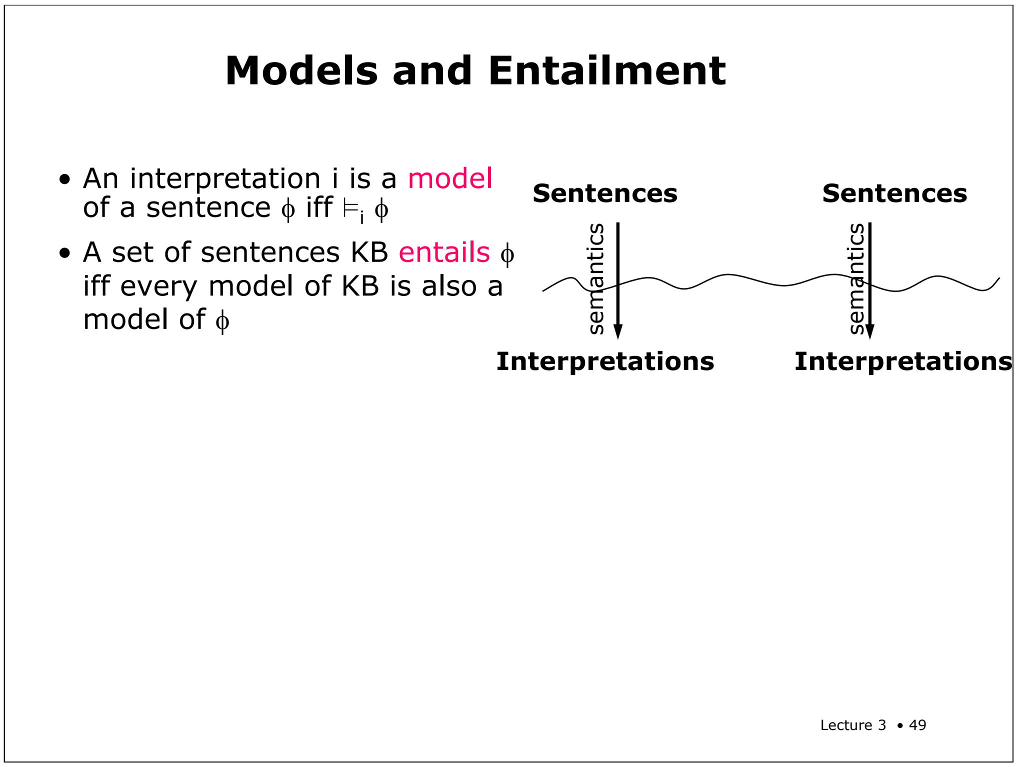
So, here's the picture. If we consider two groups of sentences, we might like to say that one set of sentences entails the other. That is, if the first set of sentences are true, then the second set will also be true. Entailment is a relationship between sets of sentences. But we define it using semantics.

Models and Entailment

- An interpretation i is a model of a sentence ϕ iff $\models_i \phi$
- A set of sentences KB entails ϕ iff every model of KB is also a model of ϕ



We take the first set of sentences through the definition of semantics and get a set of interpretations that satisfy those sentences.

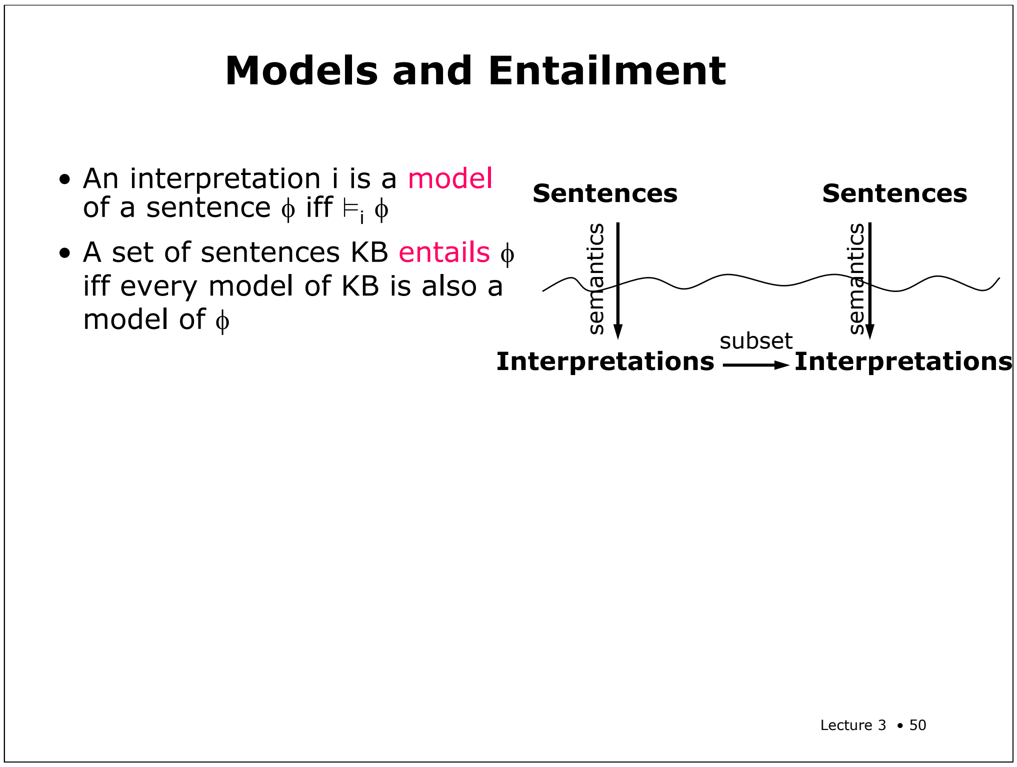


Models and Entailment

- An interpretation i is a model of a sentence ϕ iff $\models_i \phi$
- A set of sentences KB entails ϕ iff every model of KB is also a model of ϕ

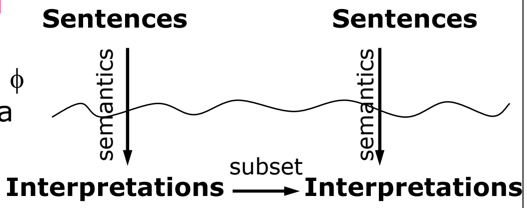
Sentences
semantics
Interpretations
Sentences
semantics
Interpretations

Then, we do the same for the second set of sentences.



Models and Entailment

- An interpretation i is a model of a sentence ϕ iff $\models_i \phi$
- A set of sentences KB entails ϕ iff every model of KB is also a model of ϕ



Now, we can ask whether the first set of interpretations is a subset of the second set.

Models and Entailment

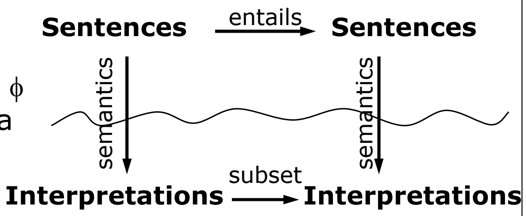
- An interpretation i is a model of a sentence ϕ iff $\models_i \phi$
- A set of sentences KB entails ϕ iff every model of KB is also a model of ϕ

$$\begin{tikzcd}
\textbf{Sentences} \arrow[r, "\text{entails}"] \arrow[d, "\text{semantics}"'] & \textbf{Sentences} \arrow[d, "\text{semantics}"'] \\
\textbf{Interpretations} \arrow[r, "\text{subset}"] & \textbf{Interpretations}
\end{tikzcd}$$

If so, we say that the first set of sentences entails the second. In all the worlds in which the first set of sentences are true, the second set will also be true.

Models and Entailment

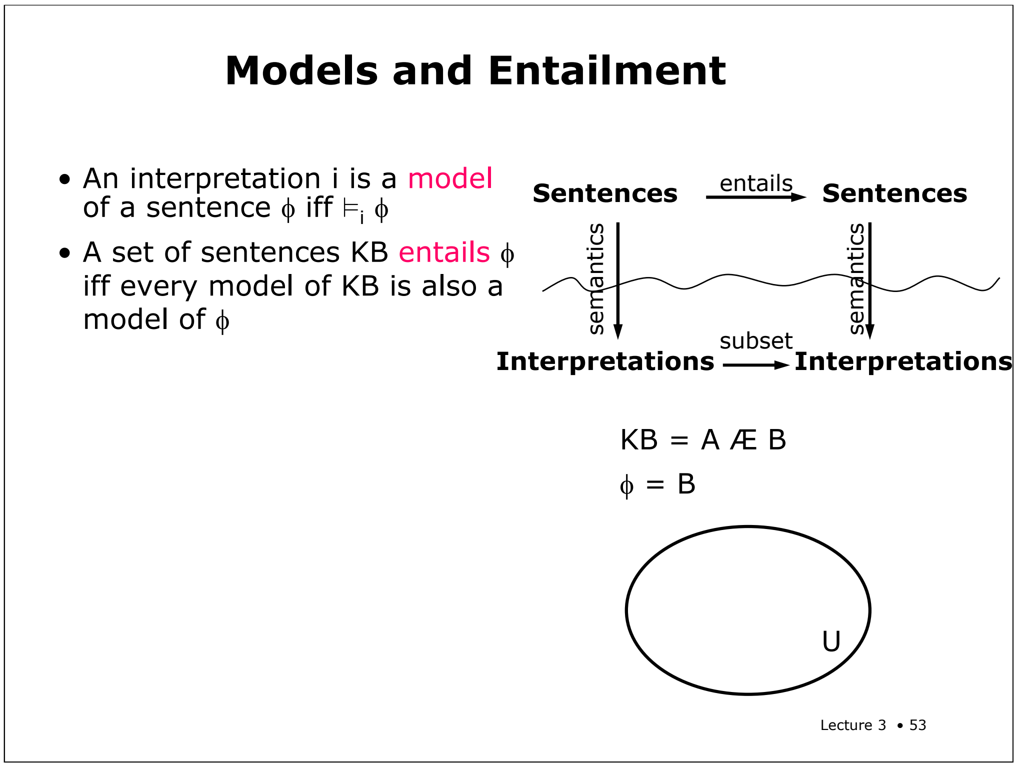
- An interpretation i is a **model** of a sentence ϕ iff $\models_i \phi$
- A set of sentences KB **entails** ϕ iff every model of KB is also a model of ϕ



KB = A Æ B

ϕ = B

We can go through this again with an example. Let's let KB be two sentences, A and B. And Let's let \Phi be B.



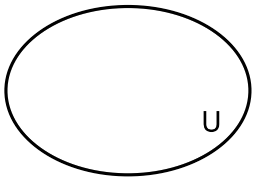
Models and Entailment

- An interpretation i is a **model** of a sentence ϕ iff $\models_i \phi$
- A set of sentences KB **entails** ϕ iff every model of KB is also a model of ϕ

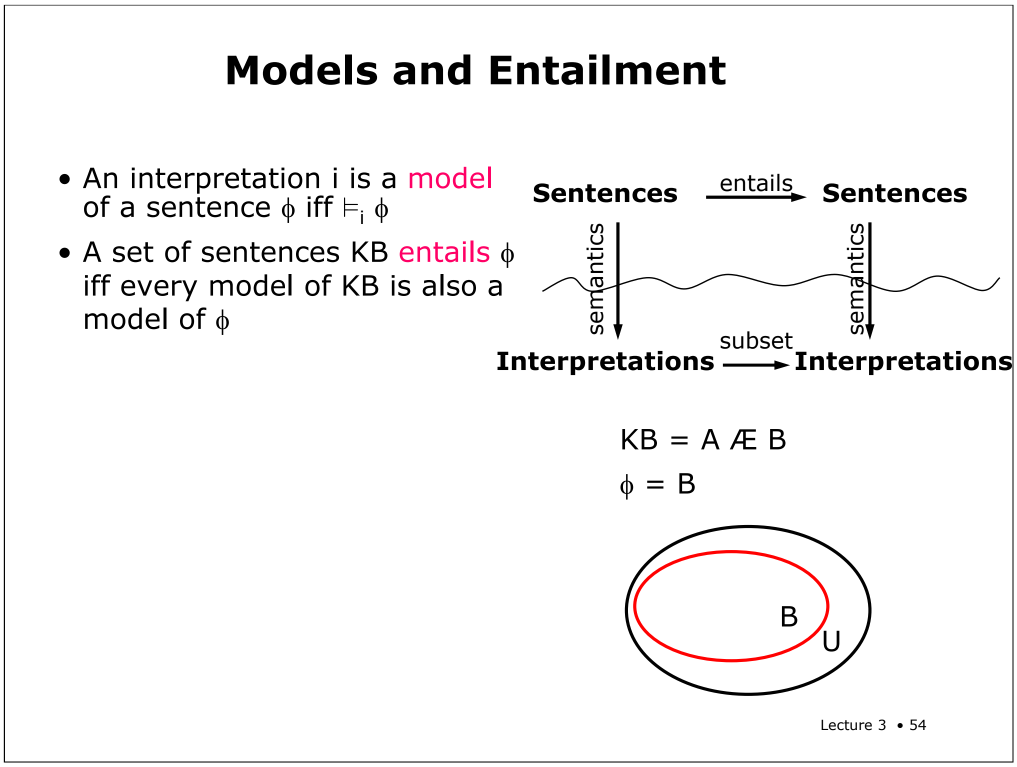
$$\begin{tikzcd}
\textbf{Sentences} \arrow[r, "\text{entails}"] \arrow[d, "\text{semantics}"'] & \textbf{Sentences} \arrow[d, "\text{semantics}"] \\
\textbf{Interpretations} \arrow[r, "\text{subset}"] & \textbf{Interpretations}
\end{tikzcd}$$

KB = A Æ B

ϕ = B



Now, we can use a Venn diagram to think about the interpretations. Let U be the set of all possible interpretations.



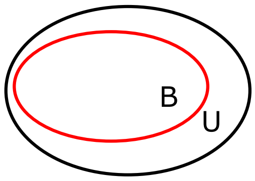
Models and Entailment

- An interpretation i is a model of a sentence ϕ iff $\models_i \phi$
- A set of sentences KB entails ϕ iff every model of KB is also a model of ϕ

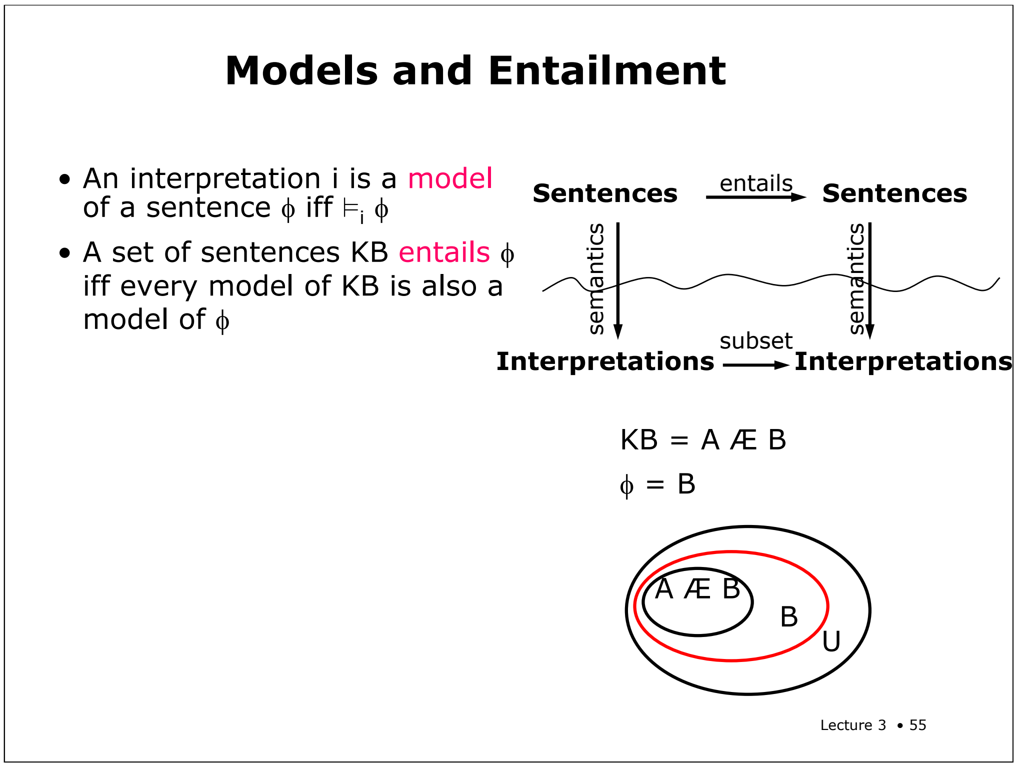
$$\begin{tikzcd}
\textbf{Sentences} \arrow[r, "\text{entails}"] \arrow[d, "\text{semantics}"'] & \textbf{Sentences} \arrow[d, "\text{semantics}"'] \\
\textbf{Interpretations} \arrow[r, "\text{subset}"] & \textbf{Interpretations}
\end{tikzcd}$$

KB = A Æ B

ϕ = B



There is a set of interpretations that are models of B (interpretations that make B true); it’s a subset of U, and we’ll label it B in our diagram.



Models and Entailment

- An interpretation i is a model of a sentence ϕ iff $\models_i \phi$
- A set of sentences KB entails ϕ iff every model of KB is also a model of ϕ

Sentences
entails
Sentences
semantics
semantics
Interpretations
subset
Interpretations

KB = A Æ B

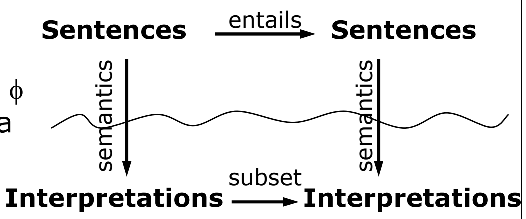
ϕ = B

A Æ B
B
U

Now, what about interpretations that are models of A and B? Clearly that's a subset of the models of B (because any interpretation that is a model of A and B has to be a model of A and a model of B).

Models and Entailment

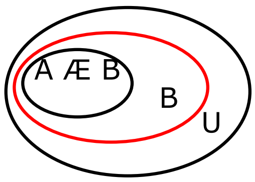
- An interpretation i is a model of a sentence ϕ iff ⊨$_i$ ϕ
- A set of sentences KB entails ϕ iff every model of KB is also a model of ϕ



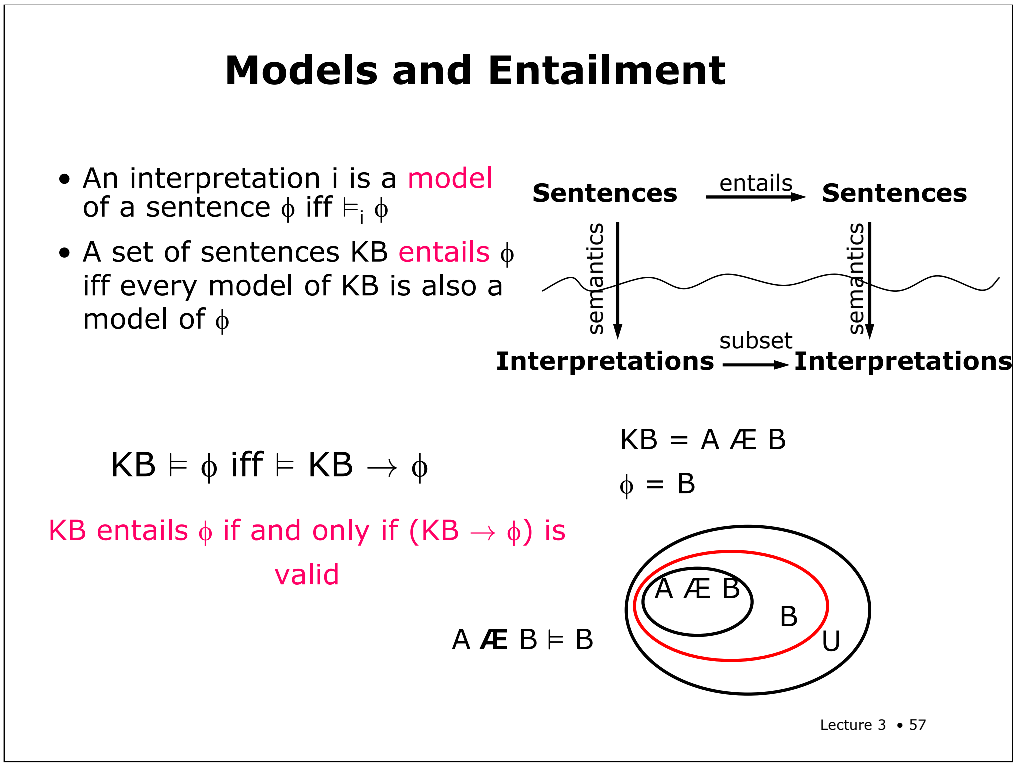
KB = A Æ B

ϕ = B

A Æ B ⊨ B



So, we find that A and B entails B. If we know that A and B are true, then B has to be true.



Models and Entailment

- An interpretation i is a model of a sentence ϕ iff $\models_i \phi$
- A set of sentences KB entails ϕ iff every model of KB is also a model of ϕ

$$\begin{tikzcd}
\text{Sentences} \arrow[r, "\text{entails}"] \arrow[d, "\text{semantics}"'] & \text{Sentences} \arrow[d, "\text{semantics}"] \\
\text{Interpretations} \arrow[r, "\text{subset}"] & \text{Interpretations}
\end{tikzcd}$$

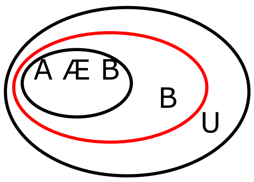
$$\text{KB} \models \phi \text{ iff } \models \text{KB} \rightarrow \phi$$

KB entails ϕ if and only if (KB $\rightarrow \phi$) is valid

KB = A Æ B

ϕ = B

A Æ B $\models$ B



We'll finish with an important theorem of logic (it's **about** logic, not **in** propositional logic): KB entails \Phi if and only if the sentence (KB implies \Phi) is valid.

That is, if the sentence \Phi is true in all models where the Knowledge Base is true, then the sentence -- KB implies \Phi is valid (true in all interpretations, so we use a turnstile without the subscript).

Why is this important? It says that if we have a way of deciding whether sentences are valid, then we have a way of checking whether one set of sentences entails another. That is, whether the truth of one set of sentences semantically requires the truth of another. This is going to lead into techniques for **proof**, which is the process of testing for validity. Next time we'll talk about proof, which is a mechanical way to grind sentences into sentences that satisfies the entailment relationship. That lets you draw valid conclusions from assumptions.

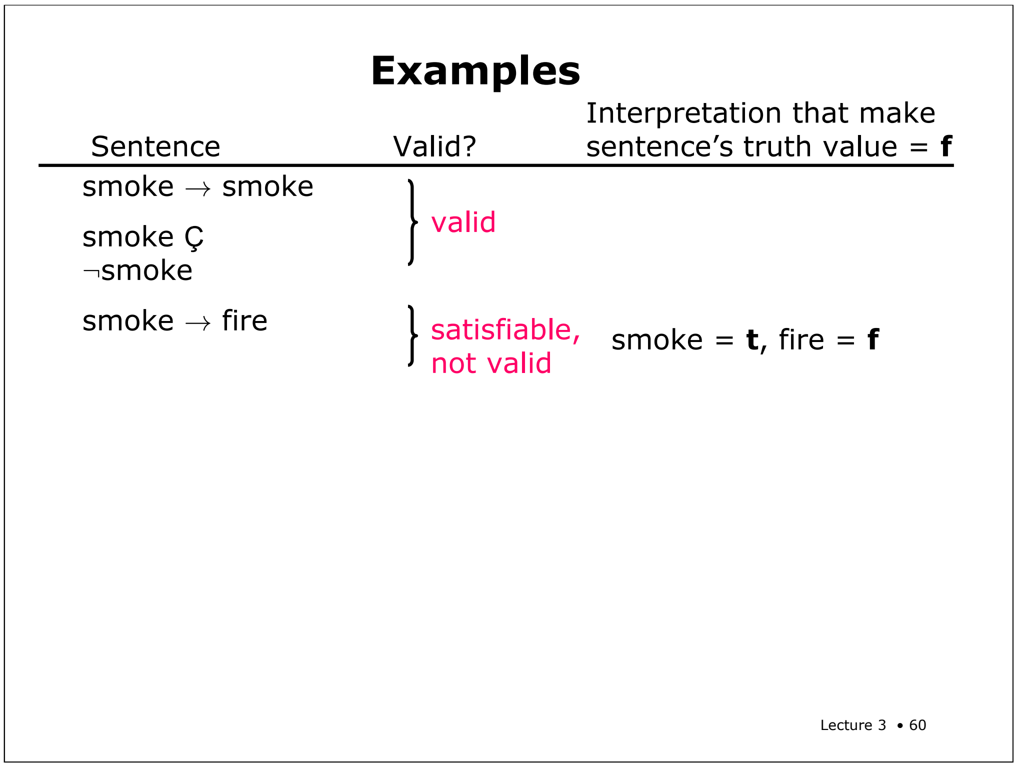
Examples

Let’s work through some examples. We can think about whether they're valid or unsatisfiable or satisfiable.

Examples

Sentence	Valid?	Interpretation that make sentence’s truth value = **f**
smoke → smoke	$\left.\right\}$ valid	
smoke Ç ¬smoke		

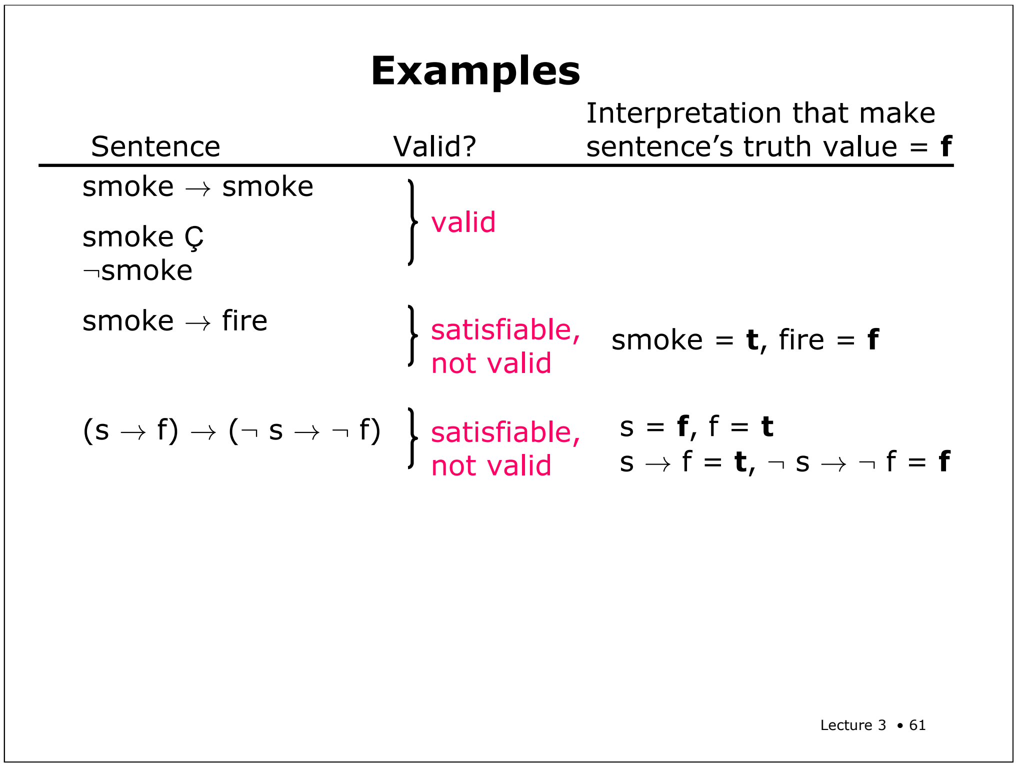
What about “smoke implies smoke”? Rather than doing a whole truth table it might be easier if we can convert it into smoke or not smoke, right? The definition of A implies B is not A or B. And we decided that smoke or not smoke was valid already.



Examples

Sentence	Valid?	Interpretation that make sentence’s truth value = **f**
smoke → smoke	valid	
smoke Ç ¬smoke	valid	
smoke → fire	satisfiable, not valid	smoke = **t**, fire = **f**

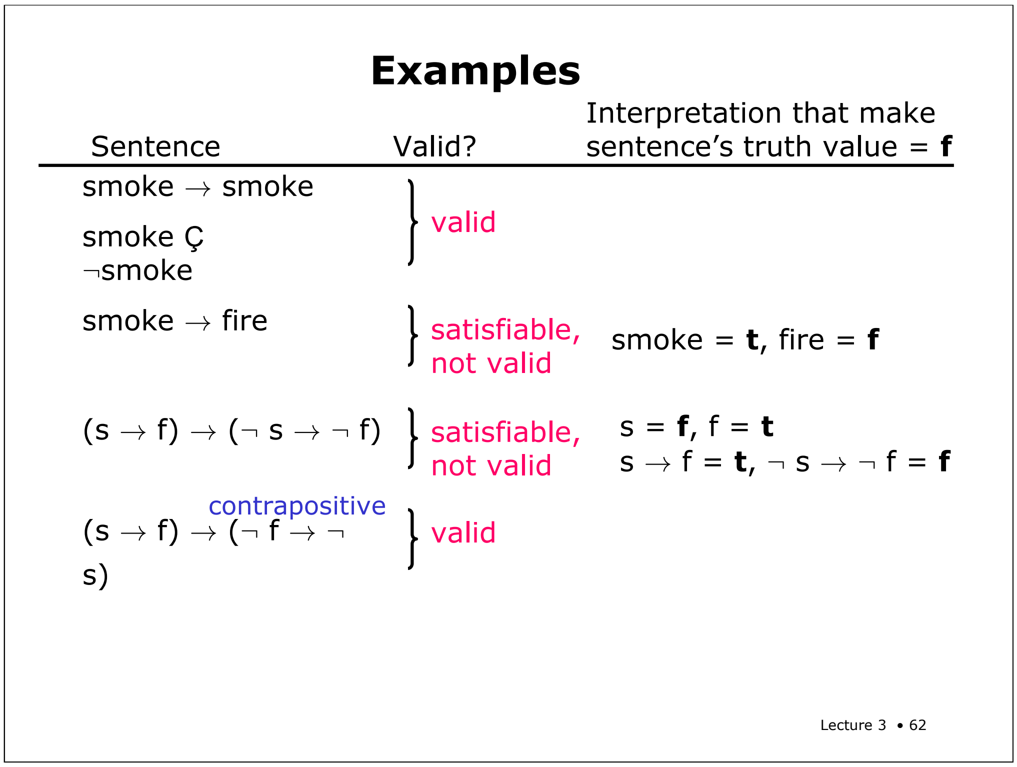
What about “smoke implies fire”? It’s satisfiable, because there's an interpretation of these two symbols that makes it true. There are other interpretations that make it false. I should say, everything that's valid is also satisfiable.



Examples

Sentence	Valid?	Interpretation that make sentence’s truth value = **f**
smoke → smoke	valid	
smoke Ç ¬smoke		
smoke → fire	satisfiable, not valid	smoke = **t**, fire = **f**
(s → f) → (¬ s → ¬ f)	satisfiable, not valid	s = **f**, f = **t** s → f = **t**, ¬ s → ¬ f = **f**

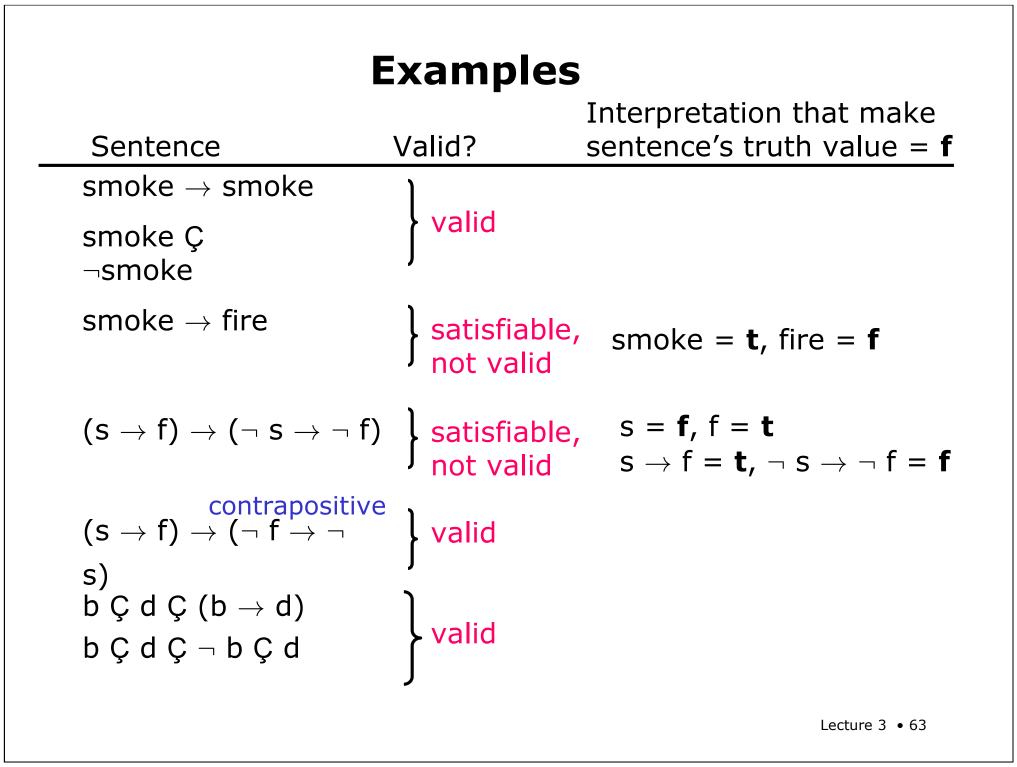
Here is a form of reasoning that you hear people do a lot, but the question is, is it OK? “Smoke implies fire implies not smoke implies not fire.” It’s invalid. We could show that by drawing out the truth table (and you should do it as an exercise if the answer is not obvious to you). Another way to show that a sentence is not valid is to give an interpretation that makes the sentence have the truth value F. In this case, if we give “smoke” the truth value F and and “fire” the truth value “T”, then the whole sentence has truth value “F”.



Examples

Sentence	Valid?	Interpretation that make sentence's truth value = **f**
smoke → smoke smoke Ç ¬smoke	valid	
smoke → fire	satisfiable, not valid	smoke = **t**, fire = **f**
(s → f) → (¬ s → ¬ f)	satisfiable, not valid	s = **f**, f = **t** s → f = **t**, ¬ s → ¬ f = **f**
contrapositive (s → f) → (¬ f → ¬ s)	valid	

Reasoning in the other direction is okay, though. So the sentence "smoke implies fire implies not fire implies not smoke" is valid. And for those of you who love terminology, this thing is called the contrapositive. So, if there's no fire, then there's no smoke.



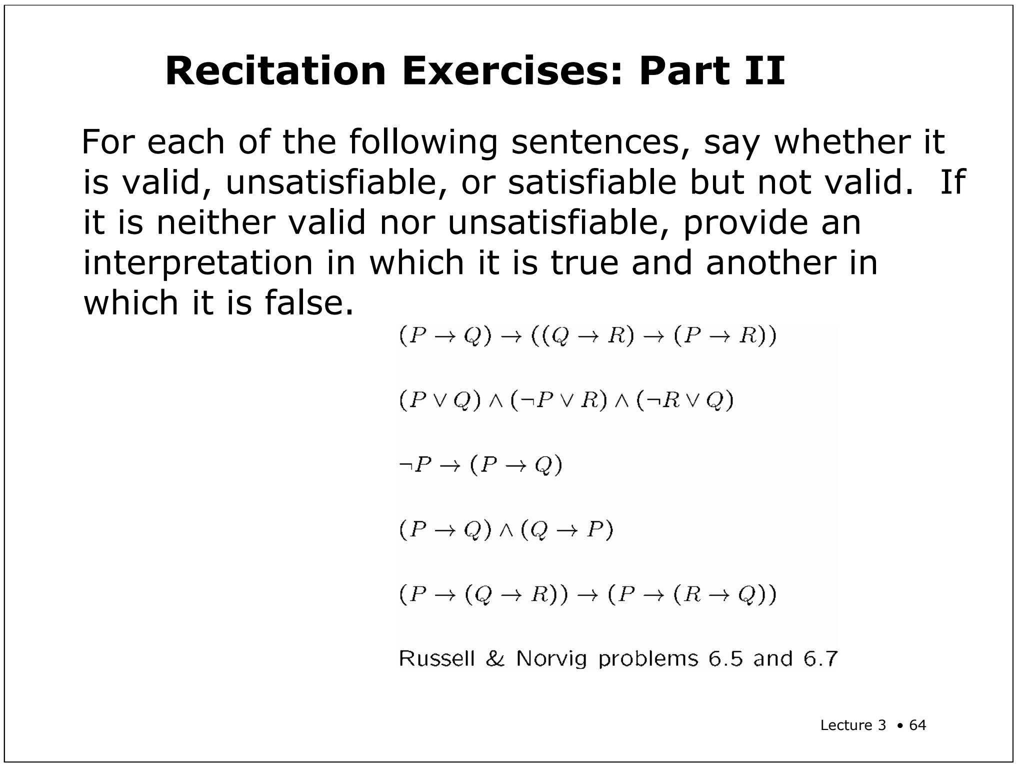
Examples

Sentence	Valid?	Interpretation that make sentence's truth value = **f**
smoke → smoke smoke Ç ¬smoke	valid	
smoke → fire	satisfiable, not valid	smoke = **t**, fire = **f**
(s → f) → (¬ s → ¬ f)	satisfiable, not valid	s = **f**, f = **t** s → f = **t**, ¬ s → ¬ f = **f**
(s → f) → (¬ f → ¬ s)	valid	
b Ç d Ç (b → d) b Ç d Ç ¬ b Ç d	valid	

contrapositive

Lecture 3 • 63

What about "b or d or (b implies d)"? We can rewrite that (using the definition of implication) into "b or d or not b or d", which is valid, because in every interpretation either b or not b must be true.



Recitation Exercises: Part II

For each of the following sentences, say whether it is valid, unsatisfiable, or satisfiable but not valid. If it is neither valid nor unsatisfiable, provide an interpretation in which it is true and another in which it is false.

$(P \rightarrow Q) \rightarrow ((Q \rightarrow R) \rightarrow (P \rightarrow R))$

$(P \vee Q) \wedge (\neg P \vee R) \wedge (\neg R \vee Q)$

$\neg P \rightarrow (P \rightarrow Q)$

$(P \rightarrow Q) \wedge (Q \rightarrow P)$

$(P \rightarrow (Q \rightarrow R)) \rightarrow (P \rightarrow (R \rightarrow Q))$

Russell & Norvig problems 6.5 and 6.7

Here are some problems to do for recitation.

Talk to you later!