

Propositional Logic

Building Blocks for Rational Thinking

CIS 32

Functionalia

Project I Demo: Monday

Project I Deliverables: Next Thursday

Write-Up Criteria

.ZIPPED Code (either in e-mail, or on USB Key in class)

Knowledge Representation - Write One Page summary on your Role (it will be due in HW 3)

Today:

Project I Code Changes

Midterm Review

Propositional Logic

Why Logic?

- “Weak” (search-based) problem-solving does not scale to real problems.
- To succeed, problem solving needs domain specific knowledge.
- In search, knowledge = heuristic.
- We need to be able to represent knowledge efficiently.
- W3C - Web Consortium is developing a Logic Based Web-Service Descriptions
(**FL**OWS - **F**irst-order **L**ogic **O**ntology for **W**eb-**S**ervices)
- One way to do this is to use logic.

What is Logic?

- When most people say “logic”, they mean either *propositional logic* or *first-order predicate logic*.
- However, the precise definition is quite broad, and literally hundreds of types of logics have been studied by philosophers, computer scientists and mathematicians.
- Any “formal system” can be considered a logic if it has:
 - a well-defined *syntax*;
 - a well-defined *semantics*; and
 - a well-defined *proof-theory*.

Components of Logic

- The **syntax** of a logic defines the syntactically acceptable objects of the language.

Properly called well-formed formulae (**wff**). (We shall just call them formulae.)

Colorless green ideas sleep furiously.

- The **semantics** of a logic associate each formula with a meaning.
- The **proof theory** is concerned with manipulating formulae according to certain rules.

Propositional Logic

- The simplest, and most abstract, logic we can study is called *propositional logic*.
- **Definition:** A *proposition* is a statement that can be either **true** or **false**; it must be one or the other, and it cannot be both.
- **EXAMPLES.** The following are propositions:
 - the reactor is on;
 - the wing-flaps are up;
 - It is raining outside.

whereas the following are not:

- are you going out somewhere?
- $2+3$

Propositional Logic

- It is possible to determine whether any given statement is a proposition by prefixing it with:

It is true that ...

and seeing whether the result makes grammatical sense.

- **Atomic propositions.** Intuitively, these are the set of smallest propositions.
- **Definition:** An *atomic proposition* is one whose truth or falsity does not depend on the truth or falsity of any other proposition.
- So all the above propositions are atomic.

- **Shortcut:** rather than write out propositions in full, we will abbreviate them by using propositional variables.
- It is standard practice to use the lower-case roman letters

p, q, r, \dots

to stand for propositions.

Sometimes, Greek letters are also used, e.g.:

ϕ	phi
Φ	capital Phi
ψ	psi
π	pi

- If we do this, we must define what we mean by writing something like:

Let p be *It is raining outside*.

- Another alternative is to write something like `it-is-raining-outside`, so that the interpretation of the propositional variable becomes obvious.

The Connectives

- Now, the study of atomic propositions is pretty boring. We therefore now introduce a number of *connectives* which will allow us to build up *complex propositions*.
- The connectives we introduce are:

\wedge and (& or .)

\vee or (| or +)

\neg not (\sim)

\Rightarrow implies (\supset or \rightarrow)

\Leftrightarrow iff (\leftrightarrow)

(Alternate symbols commonly used are in parentheses.)

And

- Any two propositions can be combined by the word “and” to form a third proposition called the **conjunction** of the original propositions.
- **Definition:** If p and q are arbitrary propositions, then the *conjunction* of p and q is written

$$p \wedge q$$

and will be true iff both p and q are true.

Truth Table

- We can summarise the operation of \wedge in a *truth table*.

The idea of a truth table for some formula is that it describes the behavior of a formula under all possible interpretations of the primitive propositions that are included in the formula.

- If there are n different atomic propositions in some formula, then there are 2^n different lines in the truth table for that formula. (This is because each proposition can take one of 2 values—i.e., *true* or *false*.)
- Let us write T for truth, and F for falsity.

Then the truth table for $p \wedge q$ is:

p	q	$p \wedge q$
F	F	F
F	T	F
T	F	F
T	T	T

Or

- Any two propositions can be combined by the word “or” to form a third proposition called the ***disjunction*** of the originals.
- **Definition:** If p and q are arbitrary propositions, then the *disjunction* of p and q is written

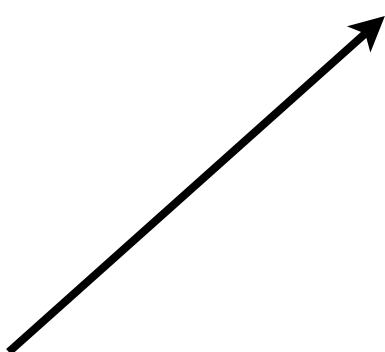
$$p \vee q$$

and will be true iff either p is true, or q is true, or both p and q are true.

Or Truth Table

- The operation of \vee is summarised in the following truth table:

p	q	$p \vee q$
F	F	F
F	T	T
T	F	T
T	T	T

- 
- **Note** that this “or” is a little different from the usual meaning we give to “or” in everyday natural language.

If...Then...

- Many statements, particularly in mathematics, are of the form:

if p is true then q is true.

Another way of saying the same thing is to write:

p implies q .

- In propositional logic, we have a connective that combines two propositions into a new proposition called the *conditional*, or *implication* of the originals, that attempts to capture the sense of such a statement.

- **Definition:** If p and q are arbitrary propositions, then the conditional of p and q is written

$$p \Rightarrow q$$

and will be true iff either p is **false** or q is **true**.

- The truth table for \Rightarrow is:

p	q	$p \Rightarrow q$
F	F	T
F	T	T
T	F	F
T	T	T

- The \Rightarrow operator is the hardest to understand of the operators we have considered so far, and yet it is extremely important.
- If you find it difficult to understand, just remember that the $p \Rightarrow q$ means “if p is **true**, then q is **true**”. (Most intuitive meaning)

Further to that, if p is **false**, then we don't care about q , and by default, we make $p \Rightarrow q$ evaluate to **T** in this case. (does not imply q is T)

Otherwise, $p \Rightarrow q$ is **false** when p is **true** and q is **false**.

- **Terminology:** if ϕ is the formula $p \Rightarrow q$, then p is the *antecedent* of ϕ and q is the *consequent*.

Iff

- Another common form of statement in maths is:

p is true if, and only if, q is true.

- The sense of such statements is captured using the *biconditional* operator.
- **Definition:** If p and q are arbitrary propositions, then the *biconditional* of p and q is written:

$$p \Leftrightarrow q$$

and will be true iff either:

1. p and q are both **true**; or
2. p and q are both **false**.

- The truth table for \Leftrightarrow is:

p	q	$p \Leftrightarrow q$
F	F	T
F	T	F
T	F	F
T	T	T

- If $p \Leftrightarrow q$ is true, then p and q are said to be *logically equivalent*.

They will be true under exactly the same circumstances.

Not

- All of the connectives we have considered so far have been *binary*: they have taken *two* arguments.
- The final connective we consider here is *unary*: It only takes one argument.
- Any proposition can be prefixed by the word 'not' to form a second proposition called the *negation* of the original.

- **Definition:** If p is an arbitrary proposition then the *negation* of p is written

$$\neg p$$

and will be true iff p is false.

- Truth table for \neg :

p	$\neg p$
F	T
T	F

Well Formed Formulae

- We can *nest* complex formulae as deeply as we want.
- We can use *parentheses* i.e., $)$, $($, to *disambiguate* formulae.
- **EXAMPLES.** If p, q, r, s and t are atomic propositions, then all of the following are formulae:

$$- p \wedge q \Rightarrow r$$

$$- p \wedge (q \Rightarrow r)$$

$$- (p \wedge (q \Rightarrow r)) \vee s$$

$$- ((p \wedge (q \Rightarrow r)) \vee s) \wedge t$$

whereas none of the following is:

$$- p \wedge$$

$$- p \wedge q)$$

$$- p \neg$$

Interpretation

- Given a particular formula, can you tell if it is true or not?
- No — you usually need to know the truth values of the component atomic propositions in order to be able to tell whether a formula is true.
- **Definition:** A *valuation* is a function which assigns a truth value to each primitive proposition.
- In C, we might write:

```
short Val( AtomicProp *p ) {  
    if ( *p )  
        return( 1 ); // true  
    else  
        return( 0 ); // false  
}
```

- Given a valuation, we can say for any formula whether it is **true** or **false**.
- A valuation is also known as an *interpretation*

- **EXAMPLE.** Suppose we have a valuation v , such that:

$$v(p) = F$$

$$v(q) = T$$

$$v(r) = F$$

Then the truth value of $(p \vee q) \Rightarrow r$ is evaluated by:

$$(v(p) \vee v(q)) \Rightarrow v(r) \tag{1}$$

$$= (F \vee T) \Rightarrow F \tag{2}$$

$$= T \Rightarrow F \tag{3}$$

$$= F \tag{4}$$

Line (3) is justified since we know that $F \vee T = T$.

Line (4) is justified since $T \Rightarrow F = F$.

If you can't see this, look at the truth tables for \vee and \Rightarrow .

- When we consider formulae in terms of interpretations, it turns out that some have interesting properties.

- **Definition:**

1. A formula is a **tautology** iff it is true under every valuation;
2. A formula is **consistent** iff it is true under *at least one* valuation;
3. A formula is **inconsistent** iff it is not made true under *any* valuation.

- A tautology is said to be *valid*.
- A consistent formula is said to be *satisfiable*.
- An inconsistent formula is said to be *unsatisfiable*.

- **Theorem:** ϕ is a tautology iff $\neg \phi$ is unsatisfiable.
- Each line in the truth table of a formula corresponds to a valuation.
- We can use truth tables to determine whether or not formulae are tautologies.
- If every line in the truth table has value **T**, the the formula is a tautology.
- Also use truth-tables to determine whether or not formulae are *consistent*.

- To check for consistency, we just need to find *one* valuation that satisfies the formula.
- If this turns out to be the first line in the truth-table, we can stop looking immediately! We have a certificate of satisfiability.
- To check for validity, we need to examine every line of the truth-table.

No short cuts.

- The lesson?
 - Checking satisfiability is easier than checking validity.

Syntax

- We have already informally introduced propositional logic; we now define it formally.
- To define the syntax, we must consider what symbols can appear in formulae, and the rules governing how these symbols may be put together to make acceptable formulae.
- **Definition:** Propositional logic contains the following symbols:
 1. A set of *primitive propositions*, $\Phi = \{p, q, r \dots\}$.
 2. The unary logical connective ' \neg ' (**not**), and binary logical connective ' \vee ' (**or**). (We will see about the others shortly.)
 3. The punctuation symbols ')' and '('.

- The primitive propositions will be used to represent statements such as:

- I am in Brooklyn
- It is raining
- It is Thursday the 12th of April.

These are primitive in the sense that they are *indivisible*; we cannot break them into smaller propositions.

- The remaining logical connectives (\wedge , \Rightarrow , \Leftrightarrow) will be introduced as abbreviations.

Grammar

- We now look at the rules for putting formulae together.
- **Definition:** The set WFF , of (well formed) formulae of propositional logic, is defined by the following rules:

1. If $p \in \Phi$, then $p \in WFF$.

2. If $\phi \in WFF$, then:

$$\neg \phi \in WFF$$

$$(\phi) \in WFF$$

3. If $\phi \in WFF$ and $\psi \in WFF$, then $\phi \vee \psi \in WFF$.

Connectives as Abbreviations

- The remaining connectives are defined by:

$$\phi \wedge \psi = \neg(\neg\phi \vee \neg\psi)$$

$$\phi \Rightarrow \psi = (\neg\phi) \vee \psi$$

$$\phi \Leftrightarrow \psi = (\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)$$

- These connectives are interpreted:

\wedge And

\Rightarrow Implies (if... then ...)

\Leftrightarrow If, and only if

- This concludes the formal definition of syntax.

Semantics

- We now look at the more difficult issue of *semantics*, or *meaning*.
- What does a proposition *mean*?
- That is, when we write

It is raining.

what does it **mean**?

From the point of view of logic, this statement is a *proposition*: something that is either \top or \perp .

- The meaning of a primitive proposition is thus either \top or \perp .
- In the same way, the meaning of a formula of propositional logic is either \top or \perp .

- **QUESTION:** How can we tell whether a formula is \top or \perp ?
- For example, consider the formula

$$(p \wedge q) \Rightarrow r$$

Is this \top ?

- The answer must be: *possibly*. It depends on your *interpretation* of the primitive propositions p , q and r .
- The notion of an interpretation is easily formalised.
- Definition: An interpretation for propositional logic is a function

$$\pi : \Phi \mapsto \{T, F\}$$

which assigns T (true) or F (false) to every primitive proposition.

- But an interpretation only gives us the meaning of primitive propositions;

What about complex propositions — arbitrary formulae?

- We use some *rules* which tell us how to obtain the meaning of an arbitrary formulae, given some interpretation.

- Before presenting these rules, we introduce a symbol: \models

If π is an interpretation, and ϕ is a formula, then the expression

$$\pi \models \phi$$

will be used to represent the fact that ϕ is \top under the interpretation π .
Alternatively, if $\pi \models \phi$, then we say that:

- π satisfies ϕ ; or
- π models ϕ

- The symbol \models is called the *semantic turnstile* (or *entailment*).

Entailment

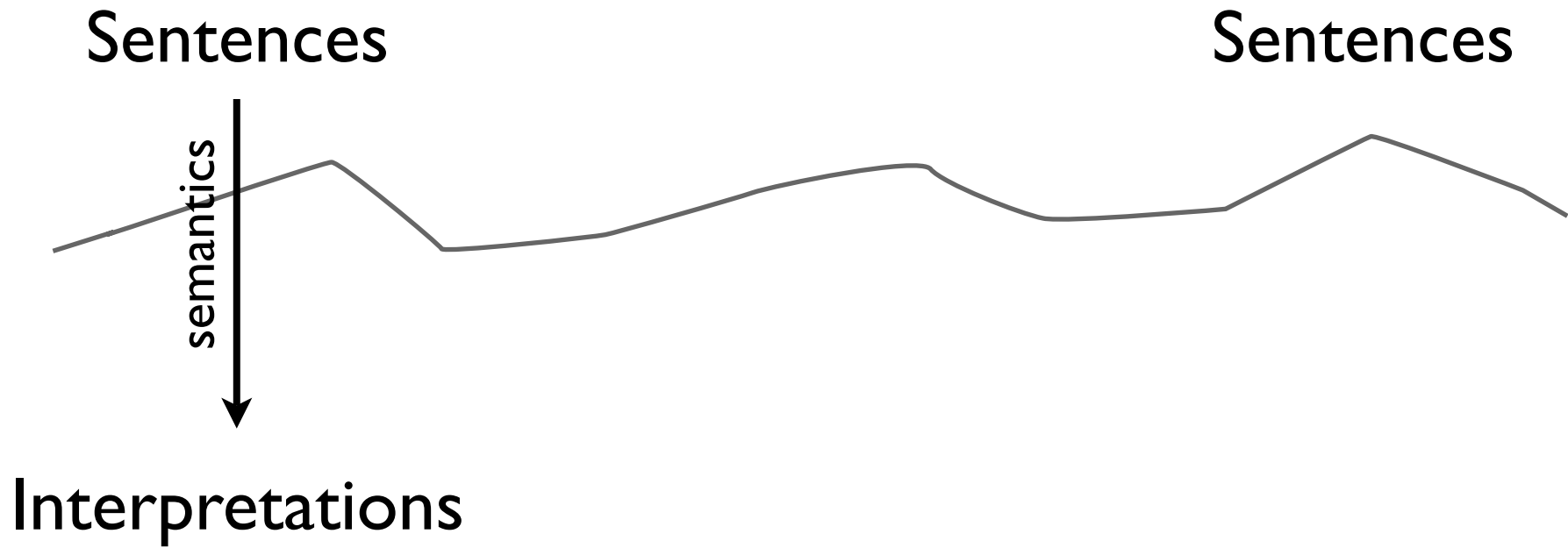
Sentences

Sentences



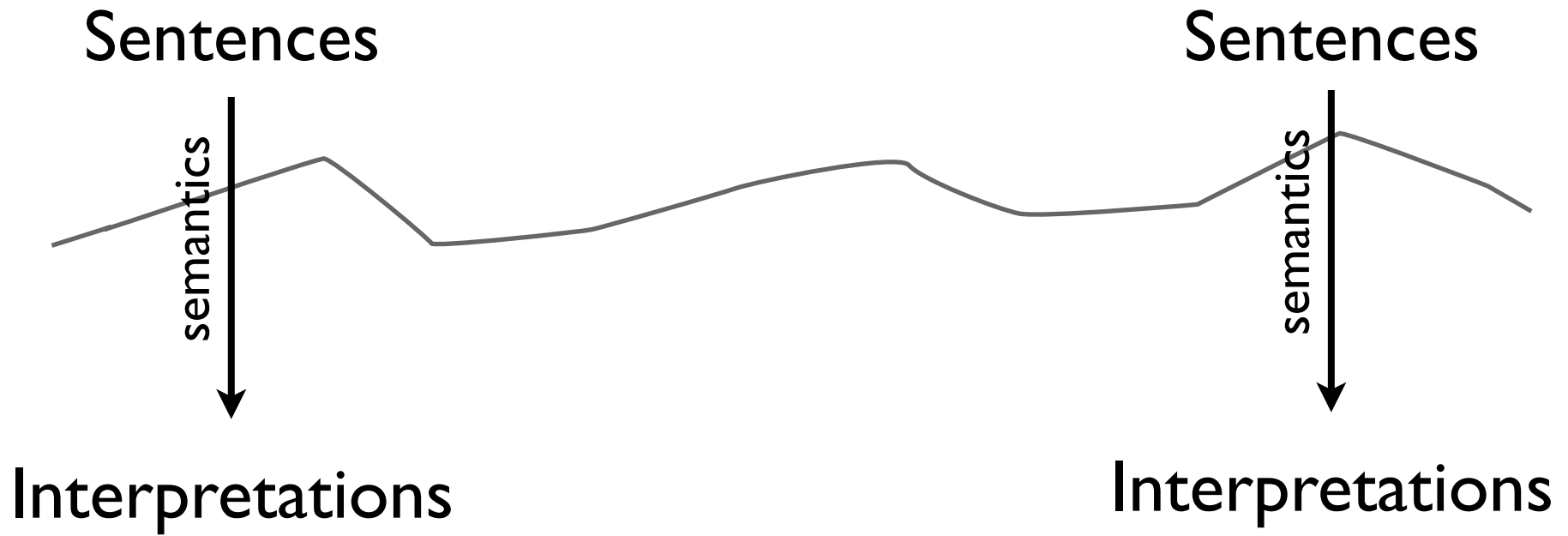
Would like to tell whether one set of Sentences *entails* the other?
(If one set of Sentences are true then the other set will be true too)

Entailment



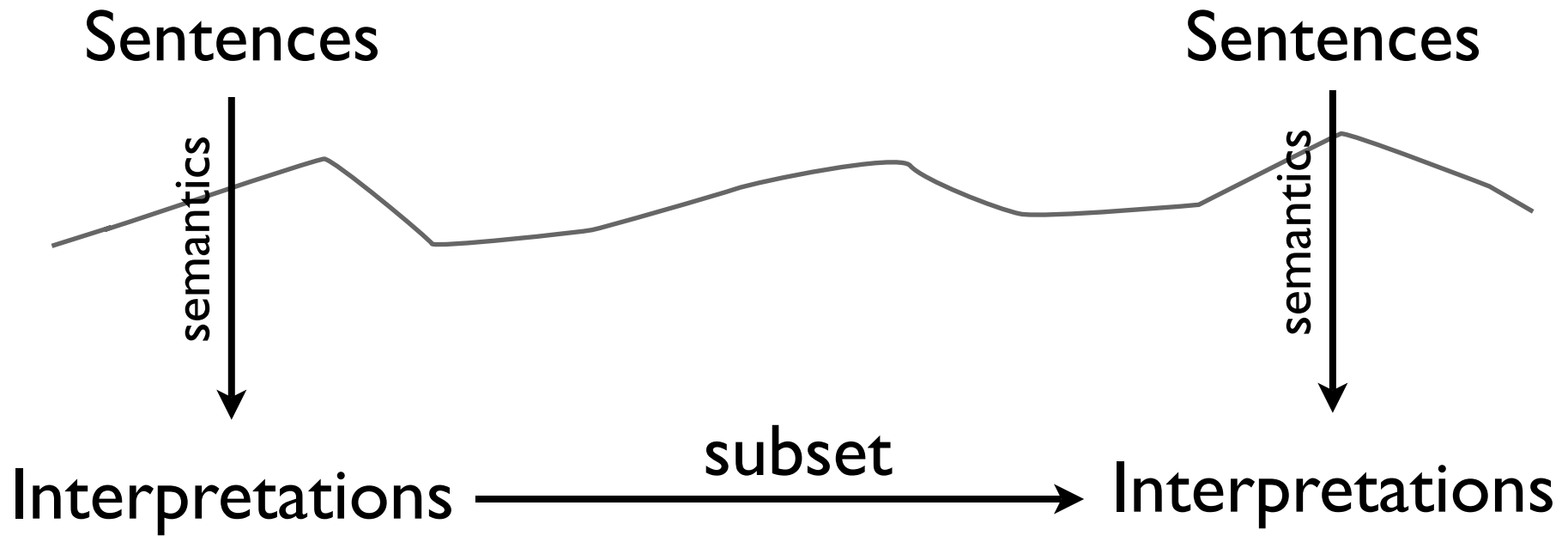
Through semantics arrive at a set of interpretations that satisfies the sentence.

Entailment



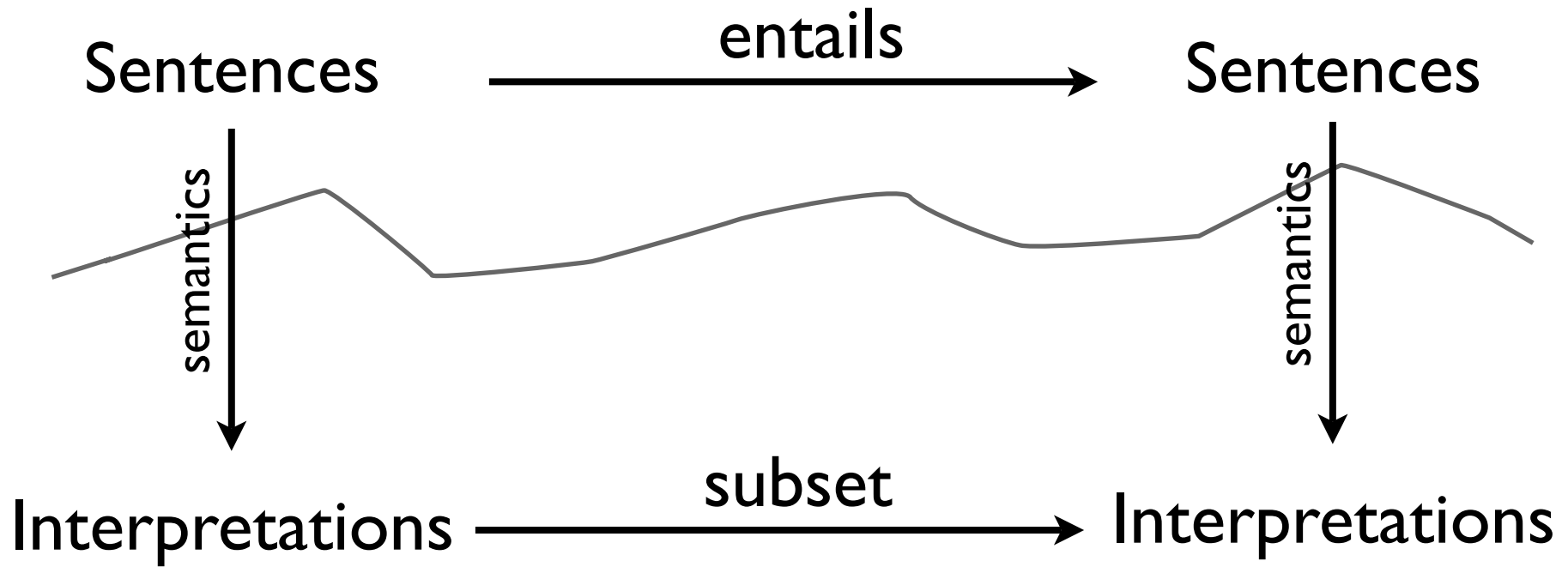
Likewise.

Entailment



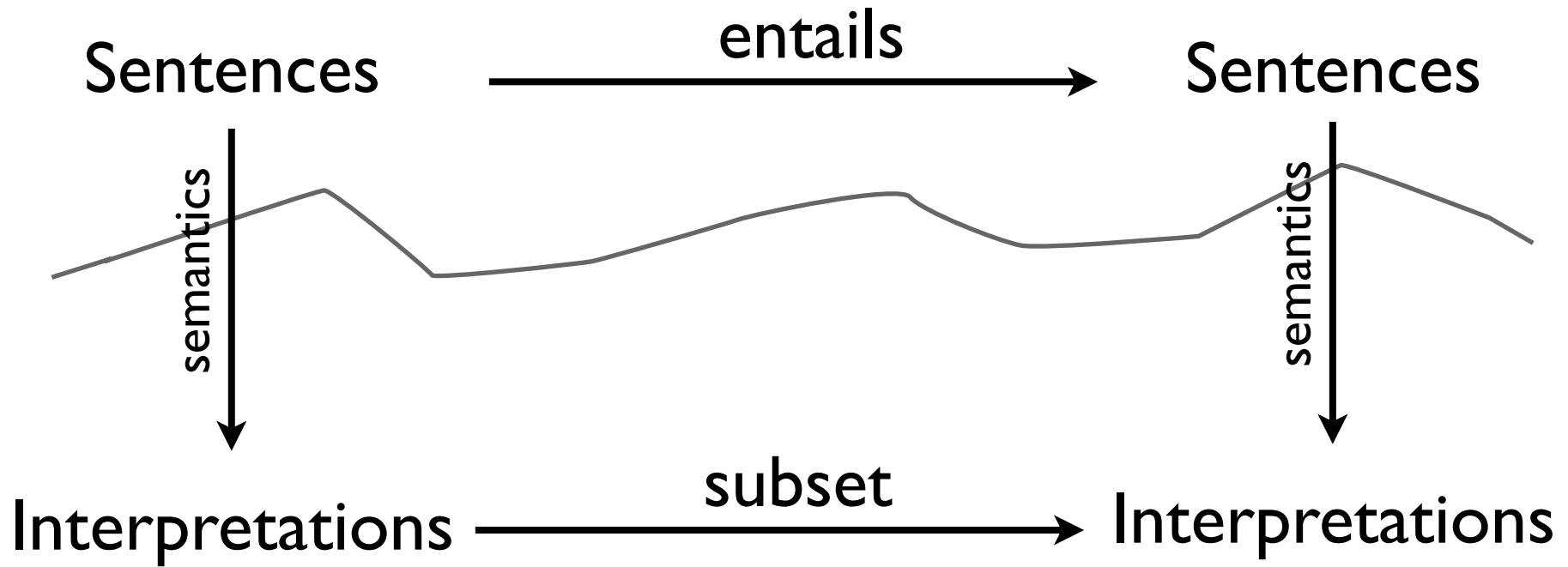
Now the question is whether the first set of interpretations is a subset of the the other set.

Entailment



In all interpretations of the first set of sentences that are true are also true for the second set of sentences.

Entailment

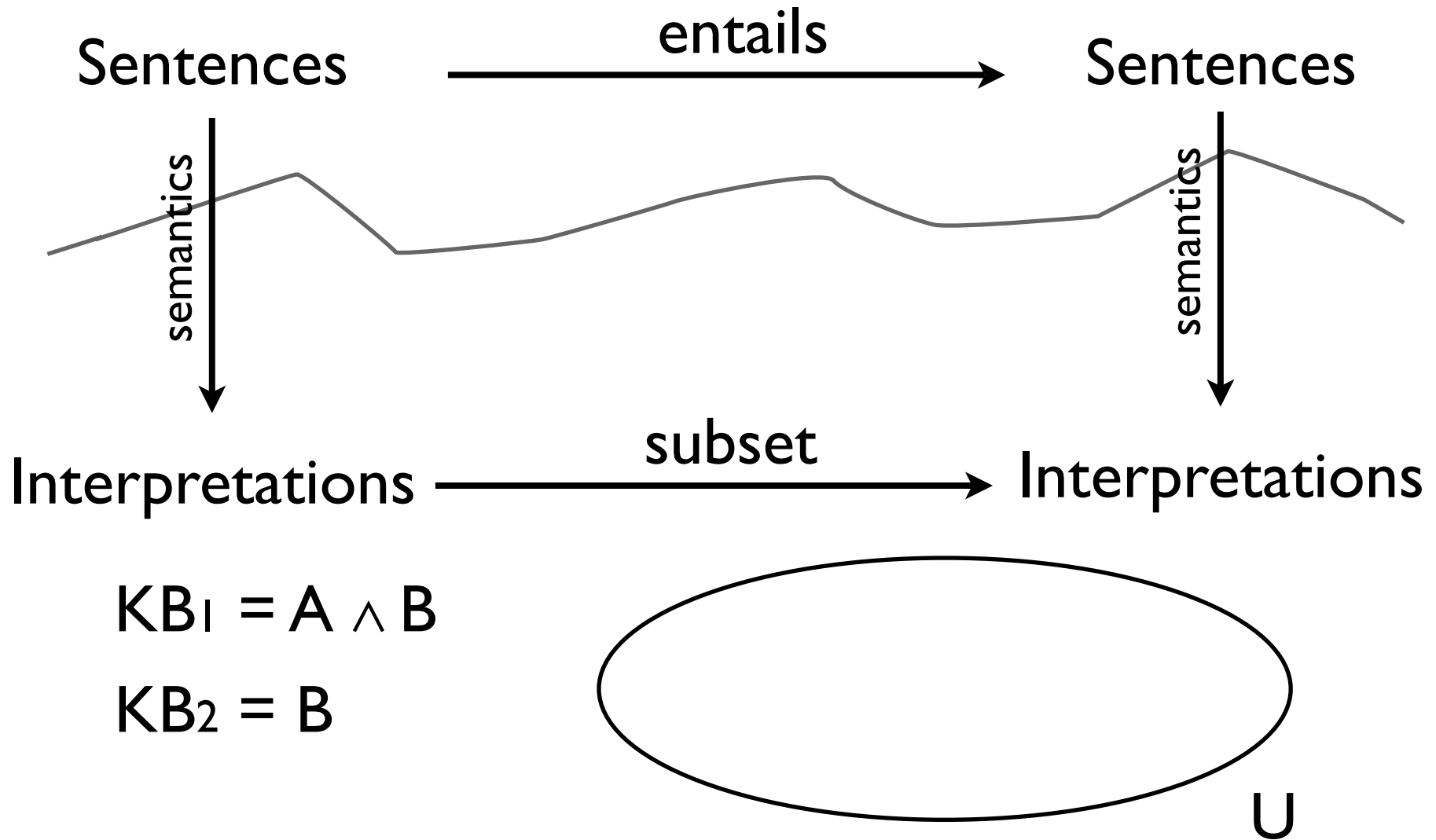


$$KB_1 = A \wedge B$$

$$KB_2 = B$$

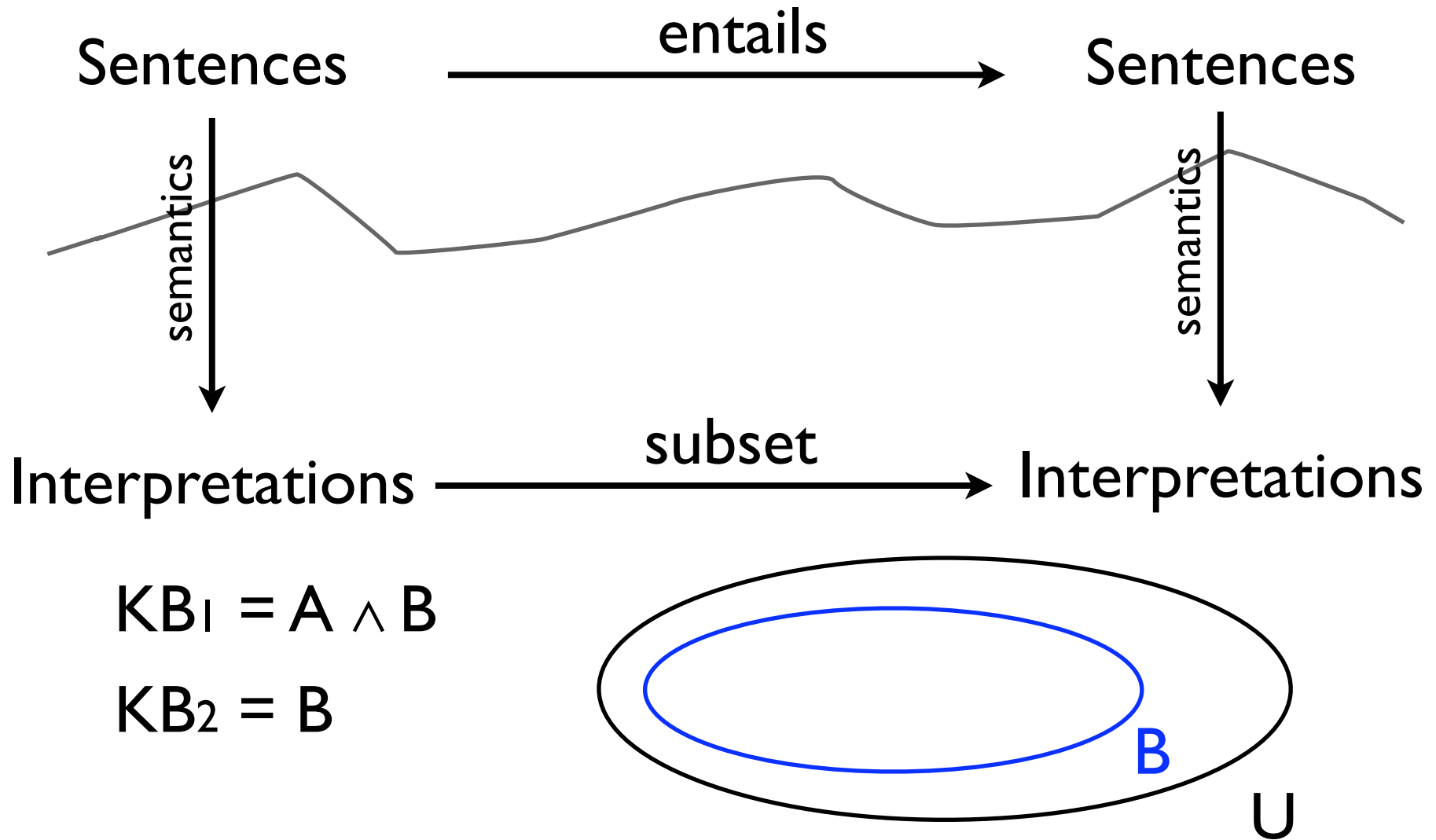
Viewed textually. Two sets of sentences (KB_1 and KB_2)

Entailment



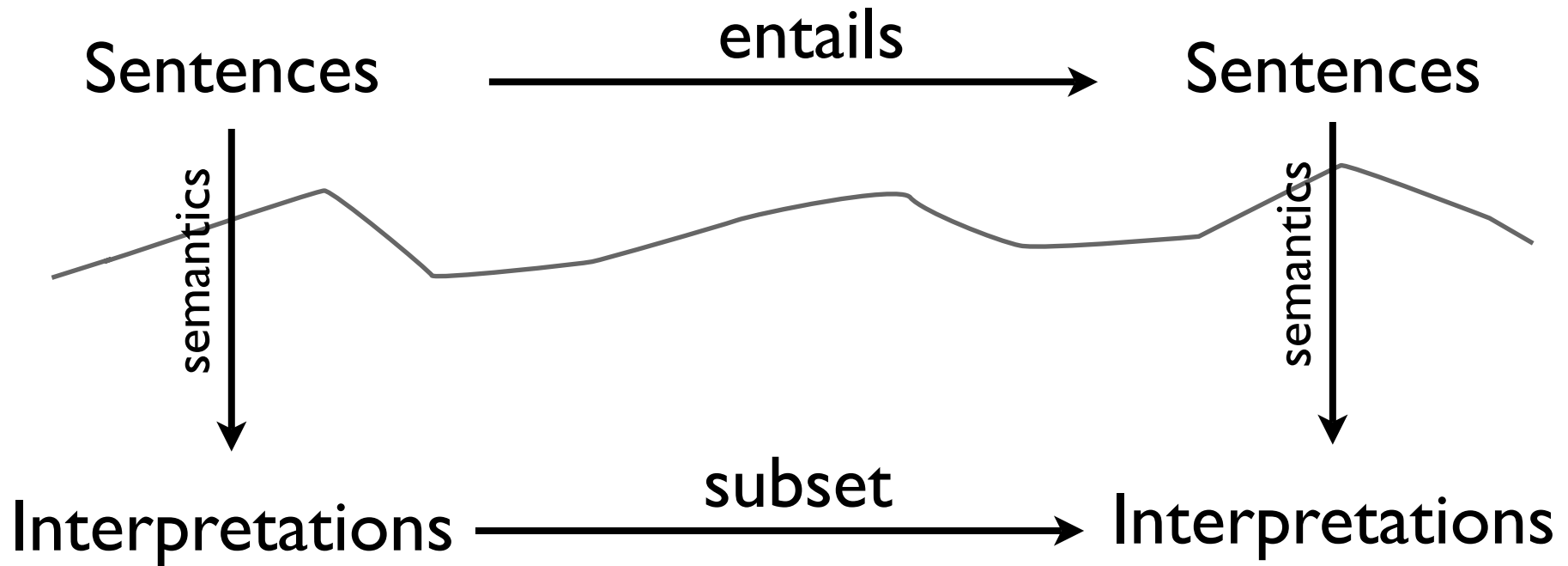
Venn Diagrams. U the *universe of discourse*

Entailment



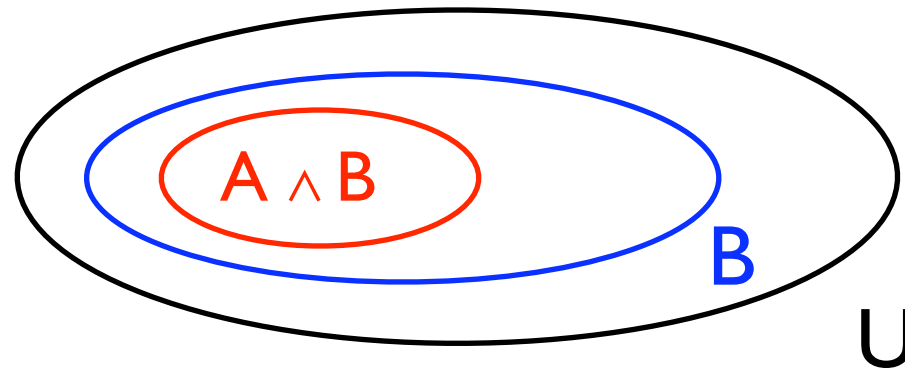
B (or rather the interpretations that make B true) lies within this domain.

Entailment



$$KB_1 = A \wedge B$$

$$KB_2 = B$$



So $A \wedge B$ is a *model* of B , as all interpretations that lie within A **and** B lie also within B .

- The rule for primitive propositions is quite simple. If $p \in \Phi$ then

$$\pi \models p \text{ iff } \pi(p) = T.$$

- The remaining rules are defined *recursively*.
- The rule for \neg :

$$\pi \models \neg\phi \text{ iff } \pi \not\models \phi$$

(where $\not\models$ means ‘does not satisfy’.)

- The rule for \vee :

$$\pi \models \phi \vee \psi \text{ iff } \pi \models \phi \text{ or } \pi \models \psi$$

- Since these are the only connectives of the language, these are the only semantic rules we need.
- Since:

$$\phi \Rightarrow \psi$$

is defined as:

$$(\neg\phi) \vee \psi$$

it follows that:

$$\pi \models \phi \Rightarrow \psi \text{ iff } \pi \not\models \phi \text{ or } \pi \models \psi$$

- And similarly for the other connectives we defined.

- If we are given an interpretation π and a formula ϕ , it is a simple (if tedious) matter to determine whether $\pi \models \phi$
- We just apply the rules above, which eventually bottom out of the recursion into establishing if some proposition is true or not.
- So for:

$$(p \vee q) \wedge (q \vee r)$$

we first establish if $p \vee q$ or $q \vee r$ are true and then work up to the compound proposition.

Proof Theory

- What is logic used for? A number of things, but most importantly, it is a *language for representing the properties of things*.
- But also, we hope it will give us *a method for establishing the properties of things*.
- To see how logic may be used to establish the properties of things, it helps to look at its history.
- Logic was originally developed to make the notion of an *argument* precise.

(We do not mean argument as in fighting here!)

- Here is a classic argument:

All men are mortal

Socrates is a man

Socrates is mortal

- This example serves to illustrate a number of features of arguments:
 - The argument has a number of *premises* — these are the statements that appear before the horizontal line;
 - The argument has a *conclusion* — this is the statement that appears after the horizontal line;
 - The argument has the form

If

you accept that
the premises are true

then

you must accept that
the conclusion is true.

Soundness

- In mathematics, we are concerned with when arguments are sound.
- To formalise the notion of a sound argument, we need some extra terminology...

- **Definition:** If $\phi \in \text{WFF}$, then:

1. if there is some interpretation π such that

$$\pi \models \phi$$

then ϕ is said to be *satisfiable*, otherwise ϕ is *unsatisfiable*.

2. if

$$\pi \models \phi$$

for *all* interpretations π , then ϕ is said to be *valid*.

- Valid formulae of propositional logic are called *tautologies*.

- **Theorem:**

1. If ϕ is a valid (*tautology*) formula, then $\neg \phi$ is unsatisfiable;

2. If $\neg \phi$ is unsatisfiable, then ϕ is valid.

- We indicate that a formula ϕ is valid by writing

$$\models \phi.$$

- We can now define the *logical consequence*.

- **Definition:** If

$$\{\phi_1, \dots, \phi_n, \phi\} \subseteq WFF$$

then ϕ is said to be a *logical consequence* of $\{\phi_1, \dots, \phi_n\}$ iff ϕ is satisfied by all interpretations that satisfy

$$\phi_1 \wedge \dots \wedge \phi_n.$$

- We indicate that ϕ is a logical consequence of ϕ_1, \dots, ϕ_n by writing

$$\{\phi_1, \dots, \phi_n\} \models \phi.$$

- An expression like this is called a *semantic sequent*.

- Theorem:

$$\{\phi_1, \dots, \phi_n\} \models \phi.$$

iff

$$\models (\phi_1 \wedge \dots \wedge \phi_n) \Rightarrow \phi.$$

- So we have a method for determining whether ϕ is a logical consequence of ϕ_1, \dots, ϕ_n :

we use a truth table to see whether $\phi_1 \wedge \dots \wedge \phi_n \Rightarrow \phi$ is a tautology.

If it is, then ϕ is a logical consequence of ϕ_1, \dots, ϕ_n .

- *Our main concern in proof theory is thus to have a technique for determining whether a given formula is valid, as this will then give us a technique for determining whether some formula is a logical consequence of some others.*

- **EXAMPLE.** Show that

$$p \wedge q \models p \vee q.$$

To do this, we construct a truth-table for

$$(p \wedge q) \Rightarrow (p \vee q).$$

Here it is:

p	q	(1) $p \wedge q$	(2) $p \vee q$	$(1) \Rightarrow (2)$
F	F	F	F	T
F	T	F	T	T
T	F	F	T	T
T	T	T	T	T

Since

$$(p \wedge q) \Rightarrow (p \vee q).$$

is true under every interpretation, we have that $p \vee q$ is a logical consequence of $p \wedge q$.

Truth Tables are Exhaustive

- The notion of logical consequence we have defined above is acceptable for a definition of a sound argument, but is not very helpful for checking whether a particular argument is sound or not.
- The problem is that we must look at all the possible interpretations of the primitive propositions. While this is acceptable for, say, 4 primitive propositions, it will clearly be unacceptable for 100 propositions, as it would mean checking 2^{100} interpretations.

(Moreover, for first-order logic, there will be an infinite number of such interpretations.)

- What we require instead is an alternative version of logical consequence, that does not involve this kind of checking.

This leads us to the idea of *syntactic proof*.

'Syntactic' Proof

- The idea of syntactic proof is to replace the semantic checking to determine whether a formula is valid by a procedure that involves purely *syntactic* manipulation.
- The kinds of techniques that we shall use are similar to those that we use when solving problems in algebra.
- The basic idea is that to show that ϕ is a logical consequence of ϕ_1, \dots, ϕ_n we use a set of *rules* to manipulate formulae.

If we can derive ϕ from ϕ_1, \dots, ϕ_n by using these rules, then ϕ is said to be *proved* from ϕ_1, \dots, ϕ_n which we indicate by writing:

$$\phi_1, \dots, \phi_n \vdash \phi.$$

- The symbol \vdash is called the *syntactic turnstile*.
- An expression of the form

$$\phi_1, \dots, \phi_n \vdash \phi.$$

is called a *syntactic sequent*.

- A *rule* has the general form:

$$\frac{\vdash \phi_1; \dots; \vdash \phi_n}{\vdash \phi} \text{ rule name}$$

Such a rule is read:

If

ϕ_1, \dots, ϕ_n are proved

then

ϕ is proved.

- **EXAMPLE.** Here is an example of such a rule:

$$\frac{\vdash \phi; \vdash \psi}{\vdash \phi \wedge \psi} \quad \wedge\text{-I}$$

This rule is called ***and-introduction***. It says that if we have proved ϕ , and we have also proved ψ , then we can prove $\phi \wedge \psi$.

- **EXAMPLE.** Here is another rule:

$$\frac{\vdash \phi \wedge \psi}{\vdash \phi; \vdash \psi} \quad \wedge\text{-E}$$

This rule is called ***and elimination***. It says that if we have proved $\phi \wedge \psi$, then we can prove both ϕ and ψ ; it allows us to eliminate the \wedge symbol from between them.

Definition of Proof

- Let us now try to define precisely what we mean by *proof*.

- **Definition:** (*Proof*) If

$$\{\phi_1, \dots, \phi_m, \phi\} \subseteq WFF$$

then there is a proof of ϕ from ϕ_1, \dots, ϕ_m iff there exists some sequence of formulae:

$$\psi_1, \dots, \psi_n$$

such that $\psi_n = \phi$, and each formula ψ_k , for $1 \leq k < n$ is either one of the formula ϕ_1, \dots, ϕ_m , or else is the conclusion of a rule whose antecedents appeared earlier in the sequence.

- If there is a proof of ϕ from ϕ_1, \dots, ϕ_m , then we indicate this by writing:

$$\phi_1, \dots, \phi_m \vdash \phi.$$

- It should be clear that the symbols \vdash and \models are related. We now have to state exactly *how* they are related.

- There are two properties of \vdash to consider:

- *soundness*;

- *completeness*.

- Intuitively, \vdash is said to be *sound* if it is correct, in that it does not let us derive something that is not true.

- Intuitively, *completeness* means that \vdash will let us prove anything that is true.

• **Definition:** (*Soundness*) A proof system \vdash is said to be *sound* with respect to semantics \models iff

$$\phi_1, \dots, \phi_n \vdash \phi$$

implies

$$\phi_1, \dots, \phi_n \models \phi.$$

• **Definition:** (*Completeness*) A proof system \vdash is said to be *complete* with respect to semantics \models iff

$$\phi_1, \dots, \phi_n \models \phi$$

implies

$$\phi_1, \dots, \phi_n \vdash \phi.$$