LING4400: Lecture Notes 3
Propositional Logic

Contents

So far we looked at a general framework for logic based on entities, truth values and functions.

Today we'll look at some functions that define a very basic kind of logic, just over truth values.

3.1 Basic propositional connectives [Boole, 1847]

1. One useful function is **negation**: Not or $\neg$.

   It maps each truth value to the opposite truth value.

   Here's the type: $\langle t, t \rangle$.

   And here's the truth table:

   $[\![\mathsf{Not}]\!]^M =$

input		output
False	:	**True**
True	:	**False**

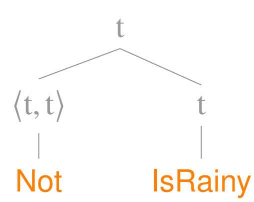
   So for example, if we have some propositions in our world model M:

   $$[\![\mathsf{IsRainy}]\!]^M = \textbf{False}$$

   Then we can negate them:

   $$[\![\mathsf{Not\ IsRainy}]\!]^M = \textbf{True}$$
   $$[\![\neg\ \mathsf{IsRainy}]\!]^M = \textbf{True}$$

   Here's the derivation tree:

2. Another useful function is **conjunction**: And or $\wedge$.

   It maps truth values to functions that map other truth values to still other truth values.

   Here's the type: $\langle t, \langle t, t \rangle\rangle$.

And here's the truth table:

$[\![\text{And}]\!]^M =$

<table>
<tr><th>input</th><th>output</th></tr>
<tr><td>False :</td><td><table><tr><th>input</th><th>output</th></tr><tr><td>False :</td><td>False</td></tr><tr><td>True :</td><td>False</td></tr></table></td></tr>
<tr><td>True :</td><td><table><tr><th>input</th><th>output</th></tr><tr><td>False :</td><td>False</td></tr><tr><td>True :</td><td>True</td></tr></table></td></tr>
</table>

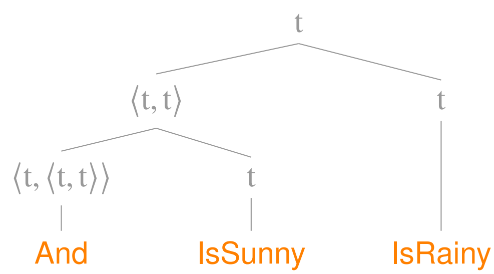
So for example, if we have some propositions in our world model M:

$$[\![\text{IsSunny}]\!]^M = \textbf{True}$$
$$[\![\text{IsRainy}]\!]^M = \textbf{True}$$

Then we can conjoin them:

$$[\![\text{And IsSunny IsRainy}]\!]^M = \textbf{True}$$

Here's the derivation tree:



Practice 3.1:

What is the interpretation of the expression And True?

Above we assume function application is **left-to-right associative**, so it's interpreted like:

$$[\![\underbrace{(\text{And IsSunny})}_{\text{this output is the second function}}\ \text{IsRainy}]\!]^M = \textbf{True}$$

This means a function in the output of another function acts like a two-argument function.

This is called a **Curried** function (named after Haskell Curry, who did it a lot).

These kinds of functions are also often written in between their arguments:

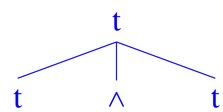
$$[\![p \wedge q]\!]^M = [\![\text{And } p\ q]\!]^M$$

For example:

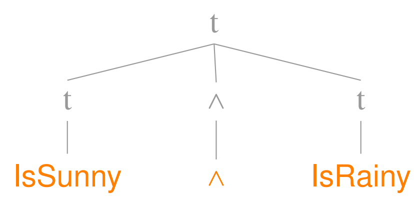
$$[\![\text{IsSunny} \wedge \text{IsRainy}]\!]^M = \textbf{True}$$

This is called **infix** notation.

We can draw trees for expressions in infix notation using flattened rules:



For example:



Practice 3.2:

Draw a derivation tree showing types for the expression $\lambda_{p:t}\ \lambda_{q:t}$ Not (And p q).

3.2 Derived propositional connectives

From these basic propositional functions, we can derive two more useful functions:

1. We can derive **disjunction** (Or or $\vee$) as a negated conjunction of negated propositions:

   $$[\![\text{Or}]\!]^M = [\![\lambda_{p:t}\ \lambda_{q:t}\ \text{Not (And (Not } p\text{) (Not } q\text{))}]\!]^M$$

   (In other words, it's *not* true that *neither* of p and q is true. At least one is true.)

   This is equivalent to a $\langle t, \langle t, t\rangle\rangle$ function with the following truth table:

   $[\![\text{Or}]\!]^M =$

input	output	
	input	output
False :	**False** :	**False**
	True :	**True**
	input	output
True :	**False** :	**True**
	True :	**True**

2. We can derive **implication** (If or →) as a disjunction with the first proposition negated:

$$[\![\text{If}]\!]^M = [\![\lambda_{p:t}\ \lambda_{q:t}\ \text{Or}\ (\text{Not}\ p)\ q]\!]^M$$

(In other words, if p is true then q is true; if p is false then q doesn't matter.)

This is equivalent to a $\langle t, \langle t, t \rangle \rangle$ function with the following truth table:

$[\![\text{If}]\!]^M =$

input	output	
	input	output
False :	**False** :	**True**
	True :	**True**
	input	output
True :	**False** :	**False**
	True :	**True**

These are **vacuously true** for false premises in the sense that rules can be tacitly obeyed:

(1) *If Mali is coastal, you have to do four thousand push-ups.*

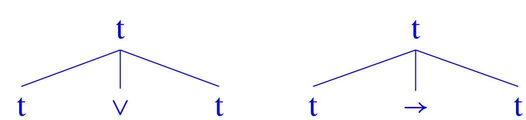
These functions are also often written using infix notation:

$$[\![p \vee q]\!]^M = [\![\text{Or}\ p\ q]\!]^M$$
$$[\![p \rightarrow q]\!]^M = [\![\text{If}\ p\ q]\!]^M$$

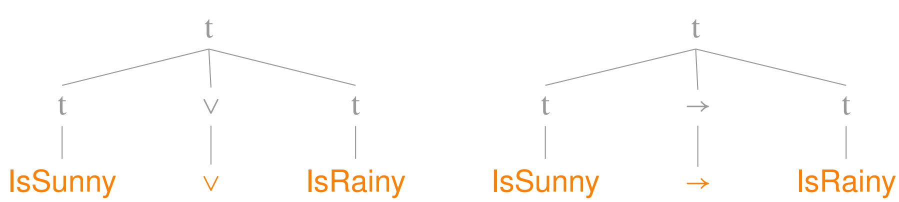
For example:

$$[\![\text{IsSunny} \vee \text{IsRainy}]\!]^M = \textbf{True}$$
$$[\![\text{IsSunny} \rightarrow \text{IsRainy}]\!]^M = \textbf{True}$$

Again, we can draw trees for expressions in infix notation using flattened rules:



For example:



Logic with just these four functions is called **propositional logic**.

Practice 3.3:

Write an expression to produce the following truth table using conjunction and negation:

<table>
<tr><th>input</th><th>output</th></tr>
<tr><td>False :</td><td><table><tr><th>input</th><th>output</th></tr><tr><td>False :</td><td>False</td></tr><tr><td>True :</td><td>True</td></tr></table></td></tr>
<tr><td>True :</td><td><table><tr><th>input</th><th>output</th></tr><tr><td>False :</td><td>False</td></tr><tr><td>True :</td><td>False</td></tr></table></td></tr>
</table>

References

[Boole, 1847] Boole, G. (1847). *The mathematical analysis of logic: being an essay towards a calculus of deductive reasoning*. Cambridge: Macmillan, Barclay, & Macmillan.