4 Analytic Tableaux

In the previous sessions we have introduced formulas and a notion of truth. The question that remains is how can we use logic to prove that a formula is true – regardless of how we interpret variable symbols? That is, how do we prove that the formula is true entirely because of structural, or *logical* reasons.

Q: *What is logical truth?*
A formula is logically true, or a tautology, if it is true under all possible interpretations.

4.1 Evaluating formulas using Truth Tables

Q: *How can we evaluate whether a formula is a tautology?*
The simplest way to evaluate the truth of a formula `X` is to compute its value for all possible interpretations v_0 of the formula's variables. That is, we write down the graph of the function `Value(X,`v_0`)` for all possible values that can be given to v_0. In other words, we simply write down all possible combinations of values `t` and `f` for the variables of the formula – that would be the interpretation v_0 – and then, following the rules for computing `Value(X,`v_0`)` we compute `Value(Y,`v_0`)` for all subformulas `Y` of `X`. This approach has generally become known as *truth table method*.

Q: *Is anyone not familiar with that method?*
Let us review this method by considering a small example formula $(P \supset Q) \supset (\sim Q \supset \sim P)$.

Q: *How would we build the truth table for that formula?*

P	Q	P ⊃ Q	∼Q	∼P	∼Q ⊃ ∼P	(P ⊃ Q) ⊃ (∼Q ⊃ ∼P)
t	t	t	f	f	t	t
t	f	f	f	t	t	t
f	t	t	t	f	f	t
f	f	t	t	t	t	t

The formula is a tautology because every row ends with t. If only a single row would result in `f`, it would not be a tautology anymore. However, it could still be *satisfiable*.

Q: *Give me a formula in that table that is satisfiable but not a tautology*

The truth table method is simple, but has one severe drawback.

Q: *What drawback does the truth table method have?*
The tables grow incredibly fast even for relatively small formulas. The number of rows grows *exponentially in the number of variables* and additionally the number of columns is linear in the number of subformulas, which is roughly the number of connectives plus the

number of variables. The example of the formula $(P \vee (Q \wedge R)) \supset ((P \vee Q) \wedge (P \vee R))$ shows how quickly a truth table may grow.

P	Q	R	Q∧R	P∨(Q∧R)	P∨Q	P∨R	(P∨Q)∧(P∨R)	(P∨(Q∧R))⊃((P∨Q)∧(P∨R))
t	t	t	t	t	t	t	t	t
t	t	f	f	t	t	t	t	t
t	f	t	f	t	t	t	t	t
t	f	f	f	t	t	t	t	t
f	t	t	t	t	t	t	t	t
f	t	f	f	f	t	f	f	t
f	f	t	f	f	f	t	f	t
f	f	f	f	f	f	f	f	t

4.2 Analytic Tableaux

Truth tables were based on the definition that *a formula is a tautology if it is true under every interpretation*. However, using truth tables for evaluating the truth of a formula quickly becomes infeasible. For larger formulas, we should therefore look for a better method – something that is fairly schematic, but doesn't rely checking individual valuations anymore.

Instead, it is better to reason about truth and falsehood as such and to *analyze* the conditions for the truth of a formula under an interpretation based on what we know about its subformulas. For this purpose let us rephrase the axioms for boolean valuations in terms of truth and falsehood. [*Note that these are actually equivalences.*]

B_1:: If ∼X is true then X is false
If ∼X is false then X is true

B_2:: If (X∧Y) is true then X and Y are true
If (X∧Y) is false then X is false or Y is false

B_3:: If (X∨Y) is true then X is true or Y is true
If (X∨Y) is false then X and Y are false

B_4:: If (X⊃Y) is true then X is false or Y is true
If (X⊃Y) is false then X is true and Y is false

So, to prove a formula X true, we look at its outer structure, apply the appropriate axiom and then check the remaining conditions for the subformulas. Let us look at a simple example.

Q: *How would you prove* P⊃P*?*

Well, P⊃P is true if P is false or P is true. And this is the case for any interpretation v_0, since v_0 can only assign `f` or `t` to the variable P.

Ok, that was easy. But actually, instead of proving a formula true, it is even easier to prove that it cannot be false, that is to **assume that X is false and to derive a contradiction from that assumption**. Sometimes this is called an *indirect proof* or a *refutation proof*

Q: *How would that work for* $P \supset P$*?*

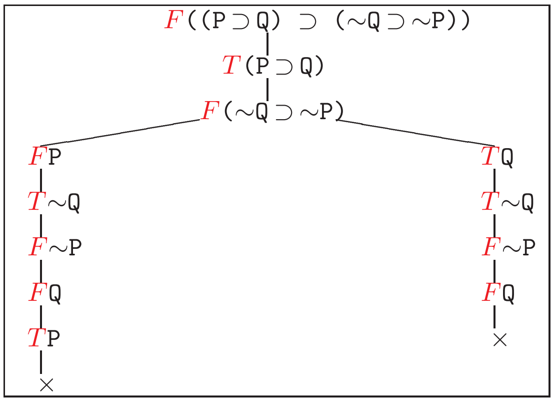
Well, if we assume that $P \supset P$ is false then P must be both true and false, and this certainly cannot be the case.

The *Tableaux Method* is schematic version of the technique we just applied. It replaces the statement "*X is true*" by the *signed formula* TX and the statement "*X is false*" by the signed formula FX. It replaces logical reasoning by the schematic application of syntactic maniulations to a signed formula and takes the rules for doing that from the above observations. Before we formulate these rules, let us look at an example.

Example: To prove $(P \supset Q) \supset (\sim Q \supset \sim P)$, we assume the formula to be false and see if we can derive a contradiction from that. So we start with the signed formula

$F((P \supset Q) \supset (\sim Q \supset \sim P))$

Keeping in mind that FX stands for "*X is false*", we use observation B_4 and derive $T(P \supset Q)$ and $F(\sim Q \supset \sim P)$, which we write below the initial signed formula. Smullyan calls these two formulas *direct consequences* of the signed formula $F((P \supset Q) \supset (\sim Q \supset \sim P))$. Look at the first of these two formulas. According to B_4 it is true, if either P is false or Q is true. This means we have to investigate two possibilities independently. We denote that by opening two *branches*, one for each argument that we have to follow.



Q: *Does anyone dare suggest what to do next?*

Next we use the second of the new formulas. Again we use B_4 and derive $T\sim Q$ and $F\sim P$, which we write as established facts into each of the two branches.

Let's look at the left branch and pick the first signed formula that we can still decompose: $T\sim Q$. Using B_1 we derive FQ and in the same way we derive TP from $F\sim P$. Let us look at the branch a little closer.

Q: *Any observations?*

We see both FP and TP in that branch, that is a signed formula and its *conjugate* (i.e. the same basic formula with a different sign – Smullyan denotes conjugates by $\bar{F}$). That means, we have shown than P is both false and true. Since both are consequences of our initial assumption, the branch is *contradictory* and we *close* it by marking it with a $\times$.

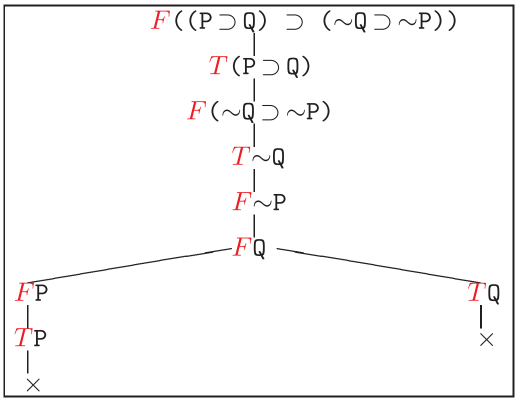
Q: *Does that mean we're done?*

But we're not done yet. We still have to look at the other branch, because this branch described just one of two possibilities. So we decompose $T\sim Q$ using B_1 and derive FQ. Now we also have a contradiction in this branch and check it off as well.

Q: *So what have we shown?*

We now have shown that the assumption that $((P \supset Q) \supset (\sim Q \supset \sim P))$ is false definitely leads to a contradiction, so $((P \supset Q) \supset (\sim Q \supset \sim P))$ must be true. ★

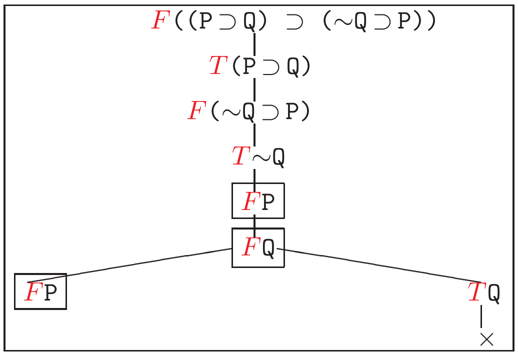
Note: we have used a top down order to decompose formulas. This is not necessary. We could also decompose the formula $F(\sim Q \supset \sim P)$ first because we use its consequences in both branches. For the same reason we would then decompose $T\sim Q$ before we finally branch by decomposing $T(P \supset Q)$.
The resulting proof would look like this



Example: Let us look at another example, where we change the formula just a little bit to $((P \supset Q) \supset (\sim Q \supset P))$.

Q: *Do you think the formula is still true – why not?*

Let us see what our method reveals. We begin as before just wipe out the ∼ before P and pretty much get the same tableau. However, we cannot close off the left branch, since we cannot decompose FP (while before we could decompose $F\sim$P).



Q: *What does that mean?*

Since we decomposed all the formulas, there is nothing else we can do, which means that our initial assumption is not contradictory. But there is more information in the tableau.

Q: *Does anyone see?*

Well, all the signed formulas in the branch are consequences of the assumption that $((P \supset Q) \supset (\sim Q \supset P))$ is false. That means that $(P \supset Q)$ must be true, $(\sim Q \supset P)$ must be false, $\sim Q$ must be true, and P and Q must be false.

So the tableau does not only show that the formula is not a tautology, but also gives us a *counterexample*, i.e. an interpretation of the variables that makes the formula false. ★

4.3 Rules for the construction of Tableaux

Let us review what we have just done. We have applied a schematic method that decomposes signed formulas according to the 4 axioms of boolean valuations. No semantic arguments were involved – just a "stupid" application of *syntactic rules*.

Smullyan states all these rules on page 17 of the book: for each connective there are two rules, one for the sign T and one for F.

1)
$$\frac{T{\sim}X}{FX} \qquad \frac{F{\sim}X}{TX}$$

2)
$$\frac{T(X \wedge Y)}{\begin{array}{c}TX\\TY\end{array}} \qquad \frac{F(X \wedge Y)}{FX \mid FY}$$

3)
$$\frac{T(X \vee Y)}{TX \mid TY} \qquad \frac{F(X \vee Y)}{\begin{array}{c}FX\\FY\end{array}}$$

4)
$$\frac{T(X \supset Y)}{FX \mid TY} \qquad \frac{F(X \supset Y)}{\begin{array}{c}TX\\FY\end{array}}$$

These rules express exactly the same as the axioms B_1 to B_4, but now in schematic form and as instructions for performing *inferences*.

The first rule, for instance, states that from $T{\sim}X$ we can directly infer FX and *add this signed formula to any branch passing through $T{\sim}X$*.

The rule for $T(X \wedge Y)$ states that we can infer both TX and TY and thus add both formulas to any branch passing through $T(X \wedge Y)$.

In contrast to that, the rule for $F(X \wedge Y)$ states that either FX or FY can be inferred. Thus the tableau has to branch if we apply the rule: to any branch passing through $F(X \wedge Y)$ we may add two new branches at the leaf – one containing FX and the other FY.

So for the propositional calculus we have two types of rules – one that adds *direct consequences* and one that *branches*. Smullyan assigns two *types* to the corresponding signed formulas. Type A (or α) to those that result in direct consequences (also called *conjunctive type*) and type B (or β) to those that cause branching (also called *disjunctive type*).

Q: *Why does he do that?*

Assigning types to signed formulas helps reducing inference rules to their essential effect, namely adding direct consequences or branching. When we later prove that the tableaux method actually works correctly, it makes quite a difference whether we have to consider 2 cases or 8. When you try to implement the method on a computer, it makes a difference whether you have to program 2 basic methods for manipulating tableaux or 8 methods, among which many are almost identical.

So Smullyan introduces a *uniform notation* for signed formulas, denoting formulas of type A by the symbol α and formulas of type B by the symbol β. For each of these formulas he defines component formulas α_1, α_2 and β_1, β_2 respectively as follows see tables page 21

α	$T(X \wedge Y)$	$F(X \vee Y)$	$F(X \supset Y)$	$T{\sim}X$	$F{\sim}X$
α_1	TX	FX	TX	FX	TX
α_2	TY	FY	FY	–	–

β	$F(X \wedge Y)$	$T(X \vee Y)$	$T(X \supset Y)$
β_1	FX	TX	FX
β_2	FY	TY	TY

These are exactly the formulas that we find in the corresponding rules, so we we know that – under any interpretation – a signed formula α is true iff both α_1 and α_2 are true and β is true iff at least one of β_1, β_2 is true. Thus we can summarize the 8 rules by the following two:

$$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad \frac{\beta}{\beta_1 \mid \beta_2}$$

Q: *So what is the point here?*

Well, logic is much about abstraction, reducing logical reasoning to the very essentials. Using an abstract language of formulas instead of informal text is one step. Using schematic rules to create proofs is the next. Finding an elegant representation of the proof system of tableaux leads us even further.

There are more steps that one could go if one were to build an efficient proof system in a computer, but we'll get to that later. The current form is the most appropriate one for another important task: showing that the tableaux method is correct and sufficient – i.e. that only true formulas can be proven and that we can in fact find a tableaux proof for every true formula. We will look into these issues in one of the next lectures.

4.4 A definition of tableaux

Now that we have explained how the tableaux method works, let us define tableaux precisely.

Definition: An *analytic tableau $\mathcal{T}$ for a signed formula X* is a dyadic ordered tree, whose points are formulas and that satisfies the following conditions.

1. The root of $\mathcal{T}$ is X.
2. If a node y has one successor then there is some α on the path P_y and the successor of y is α_1 or α_2.
3. If a node y has two successors then there is some β on the path P_y and β_1, β_2 are the successors of y.

A branch θ of a tableau $\mathcal{T}$ is *closed* if it contains a formula and its conjugate (or its negation, for unsigned formulas). A *tableau $\mathcal{T}$ is closed* if all its branches are closed.

A *proof* of an unsigned formula X is a closed tableau for FX (or $\sim X$).

A tableau $\mathcal{T}_2$ is a *direct extension* of $\mathcal{T}_1$ if $\mathcal{T}_2$ is the result of applying the α-rule or the β-rule to one of the branches of $\mathcal{T}_1$.

A branch θ of a tableau $\mathcal{T}$ is *complete* if for every α on θ both α_1 and α_2 occur on θ and if for every β on θ at least one of β_1, β_2 occur on θ. A *tableau $\mathcal{T}$ is complete* if all its branches are either closed or complete.

4.5 Proof Strategies

A tableaux proof for a formula X is constructed by iteratively extending the root tableau FX until it is complete. If the resulting tableau is closed, the formula is true, otherwise it is not. In one of the next lectures we will prove this fact.

A tableau can be extended by applying either an α-rule or a β-rule. We have a certain degree of freedom here, since we may pick an arbitrary branch and an arbitrary α- or a β-formula on this branch. Once a formula has been used, it doesn't make sense to use it again on the same branch. Since every rule decomposes a formula into smaller components (i.e. reduces the degree of the sub-components), the extension process must eventually terminate.

There are several proof strategies one may follow. The simplest one is to just *decompose the formulas in top down order* and mark them off as used. However, as our initial examples showed, it is usually more effective to *decompose α-formulas first*, since this will keep the proof from branching too early.