Last Tuesday we have looked into Gentzen systems as an alternative proof calculus, which focuses on constructing a proof for a given formula rather than trying to refute it and concluding from a failed refutation that the formula is valid. I went over the individual proof rules and tried to present an intuitive justification for them. To emphasize that Gentzen systems actually provide evidence for the truth of a formula, I described for each rule a method for constructing the corresponding evidence, using an ML-like notation for "abstract proof terms" to make this a bit more precise.

There were two small issues that I would like to clarify.

(1) If you look into Gentzen's original paper and Smullyan's account in chapter 11 of the book you will find that they write rules in a *synthetic style*, that is the premises of the rule are written on top of a horizontal line and the conclusion derived from these premises below. Our account of logical rules uses an *analytic style* – or *top-down*, just as we did for tableau systems. The rules in the handout first show the main goal and below that the subgoals derived by applying the rule.

Although the meaning is still that the subgoals are premises that imply the main goal (or conclusion), this *refinement style* better reflects the way *how proofs are developed*. You start with a formula that you want to prove, apply a rule, get subgoals that remain to be proven, and continue until there are no more unproven subgoals. Note that evidence is constructed bottom up once you have completed the proof.

With synthetic rules you would either have to develop a proof from bottom to top – and essentially do the same – or start with the leaves and build you proof, assuming you know exactly what you need. The synthetic style is only good for writing down proofs once you know them, but not for developing them.

Top down proof rules are designed for computer-supported, but interactive reasoning – which is particularly important if we go beyond propositional or even first-order logic, where fully automated proof methods aren't strong enough anymore. This is also the reason why Gentzen systems provide all the information relevant for continuing the proof process locally in a single sequent while in tableau proofs one has to look at the whole proof tree.

(2) The second is the issue of *reversible* rules. A rule is reversible, if applying it does not destroy information, that is if the main goal is equivalent to the conjunction of all the subgoals of the rule.

You can easily prove that all the rules of Gentzen's multi-conclusioned sequent calculus are reversible if you replace the comma on the left hand side by $\wedge$, on the right hand side by $\vee$, and the turnstyle by $\supset$, and consider the unknown the set of formulas as just one big formula. One of the reasons to allow multiple conclusions is exactly that ... we want the rules to be reversible. We will see irreversible rules once we begin looking at a logic that doesn't allow multiple conclusions. I will go into that either by the end of this lecture or next Tuesday.

10.1 Consistency and completeness of Gentzen Systems

One of the issues that I have touched on already but not yet completed is the question of consistency and completeness of Gentzen Systems. Given the intuitive explanations it seems clear that the rules must be correct correct but that is not enough to convince ourselves that we haven't overlooked a subtle mistake. Besides, this doesn't say anything about the completeness of Gentzen Systems.

Q: *What again is the distinction between consistency and completeness?*

So we could go on and proceed the same way as we did with tableaux: we extend the notion of boolean valuations to sequents, define the notions of satisfiability and tautology, and then go on proving that a sequent is a tautology if and only if there is a sequent proof for it. Given that we just spent more than an hour to do that for tableau systems this look too appealing – actually it would be quite boring to repeat all that again for another proof calculus.

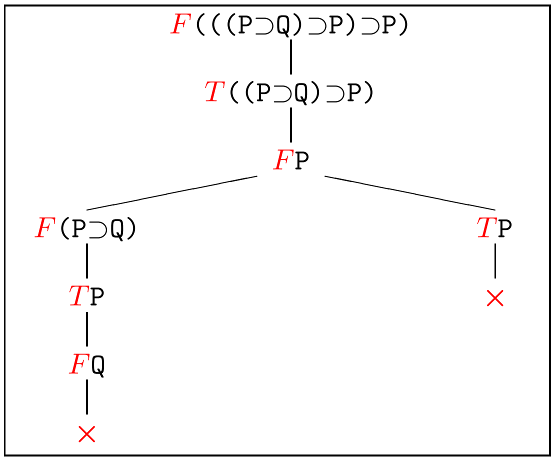
There is a much simpler way, because – as I have pointed out several times already – Gentzen systems and tableau are actually quite similar. When we prove a formula in either system we can more or less use the same kind and order of rules. When we write the proofs side by side (do this for $((P \wedge Q) \supset R) \supset P \supset Q \supset R$) we also observe that each sequent in the Gentzen proof contains exactly those formulas of the tableau that have not yet been decomposed ... with the once labelled F on the right and the ones labelled T on the left of the $\vdash$. In a sense, Gentzen systems and the tableau method are dual to each other and therefore we can prove consistency and completeness of Gentzen systems by providing a translation between Gentzen and tableau proofs, showing that the two are actually *isomorphic*.

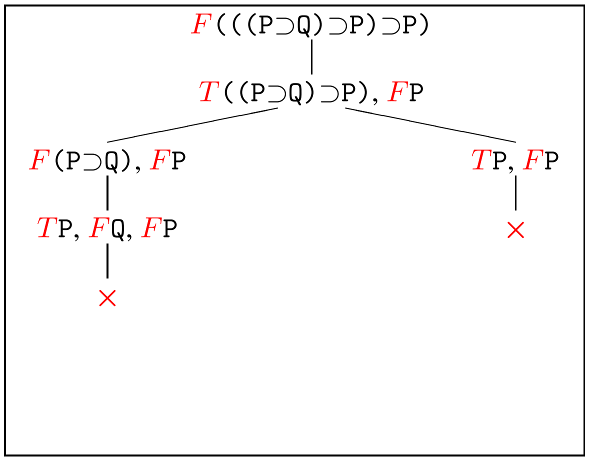
In order to turn this informal observation into something more precise I have to go back to tableau systems and formalize the concept of "formulas that have not been decomposed yet". Smullyan calls the modification of the original tableau method that handles these formulas explicitly in a proof node Block Tableaux.

10.2 Block Tableaux

One of the disadvantages of the original tableau method from the perspective of a human user is that it is difficult to keep track of those formulas in the proof tree that still can be decomposed, thus generating new signed formulas that may be essential for closing the current branch. One essentially has to look at the whole path in the proof tree between the root and the current proof node and determine which of these nodes haven't been used yet.

For interactive use it would be much better to redesign the calculus in a way that it supports *local* reasoning. Essentially this means that we add all the formulas to a proof node that are above it in the tree but haven't been decomposed yet and can still be analyzed. So what we essentially do is to use a tableau calculus that operates on *sets of signed formulas* instead of a single one and doesn't require a user to look anywhere else but at the current proof node. The following example illustrates the difference


$F(((P\supset Q)\supset P)\supset P)$
$T((P\supset Q)\supset P)$
FP
$F(P\supset Q)$
TP
TP
FQ
×
×

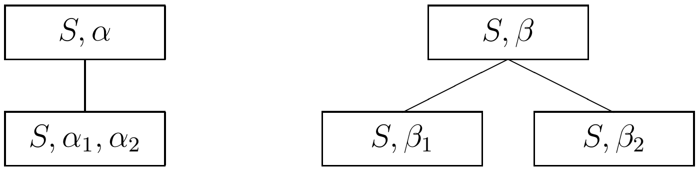

$F(((P\supset Q)\supset P)\supset P)$
$T((P\supset Q)\supset P)$, FP
$F(P\supset Q)$, FP
TP, FP
TP, FQ, FP
×
×

As we can see, the block tableau proof is even shorter because it keeps the two formulas generated by most α-rules within the formula set of a single successor node. Apart from the the proofs are essentially the same.

The formal modifications for the block tableau calculus are simple.

(1) The root of a tableau tree is now a finite set of formulas, usually the set consisting of the one original formula to be refuted.

(2) α and β rules operate on sets with a distinguished formula



In these rules, the comma stands for set union, so the α and β might be elements of S

(3) A block tableau is *closed* if each end point contains a formula and its complement and it is *atomically closed* if it contains a (signed) variable and its complement.

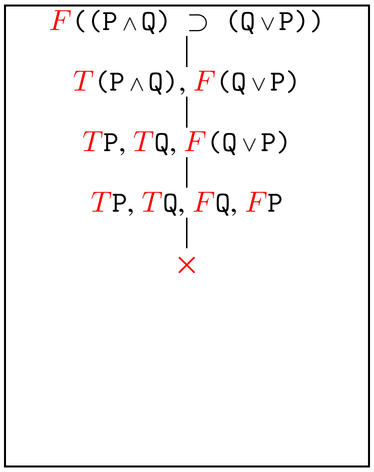
Smullyan uses two α-rules, one that adds only α_1 and the other adds only α_2. But this is not really necessary because adding the other α-successor does not affect whether a branch can be closed or not.
If we spell out these rules for each logical connective and write the rules in a less graphical fashion, but simply use one line for each generated branch, we get the table of rules that I gave you in the handout. The condition for closing a tableau can also be formulated as rule that closes a branch, because it creates no successor.

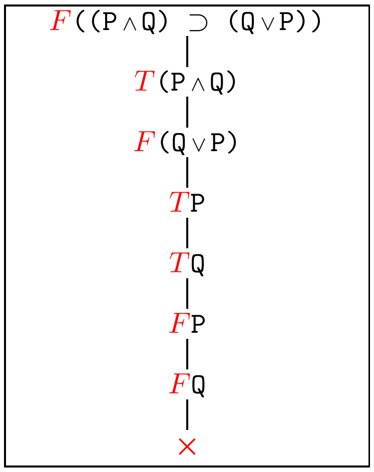
	T	F	
α	S, TA∧B	S, FA∧B	β
	S, TA, TB	S, FA	
		S, FB	
β	S, TA∨B	S, FA∨B	α
	S, TA	S, FA , FB	
	S, TB		
β	S, TA⊃B	S, FA⊃B	α
	S, FA	S, TA, FB	
	S, TB		
α	S, T∼A	S, F∼A	
	S, TA	S, FA	α
*	S, TA, FA		

As the above example demonstrated, a conventional tableau for a single formula an immediately be converted into a block tableau for that formula. One just collects all the formulas that have been generated before and not yet been decomposed into a set of formulas for the block tableau. Apart from that and the fact that α-rules in the block tableau generate only one node where the conventional tableau creates two, the block tableau is a *one-to-one simulation* of the analytic tableau, because we apply the same rules to the same nodes. If the convential tableau is closed, so is the block tableau. If it is open, so is the block tableau. Thus *every analytic standard tableau can be simulated by a block tableau* and as a result of that *block tableaux are complete*

Q: $\boxed{\textit{Why?}}$
because due to completeness of analytic tableau every valid formula has a closed analytic tableau and hence a closed block tableau.

Let us look at the other direction and show that *every block tableau can be simulated by an analytic standard tableau*. Let us look at another example





Again we use the same rules at the same nodes in both tableau and notice that -rules in analytic standard create two successors instead of the one generated in the block tableau. Thus *block tableaux are consistent* since every formula that has a closed block tableau also has a closed analytic tableau and thus, by consistency of analytic tableau, is valid.

Thus we have shown that *the block tableau calculus is consistent and complete* and thus provided a foundation for proving Gentzen systems consistent and complete.

10.3 Gentzen systems are essentially block tableaux

Let us come back to the abovementioned similarity between Gentzen systems and tableaux.

We know that a sequent $\{X_1, ...X_n\}\vdash\{Y_1, ...Y_m\}$ pretty much expresses the same as the formula $(X_1 \wedge ...X_n)\supset(Y_1 \vee ...Y_m)$. In order to prove this formula, the tableau method would begin decomposing $F(X_1 \wedge ...X_n)\supset(Y_1 \vee ...Y_m)$ by using the α-rule for $\supset$, which yields $T(X_1 \wedge ...X_n)$ and $F(Y_1 \vee ...Y_m)$. After applying the α-rules for $\wedge$ and $\vee$ we would eventually get the set of signed formulas $\{TX_1, ...\ TX_n, FY_1,...FY_m\}$. In a block tableau, this set would be a single proof node, since none of the X_i and Y_j have been decomposed yet.

Now if we convert every formula X in the hypotheses of the initial sequent into TX and every formula Y in the conclusion into FY and drop the turnstyle

Q: $\boxed{\textit{What do we get?}}$
we get exactly the same set of signed formulas. And if we move every T-formula in that set to the hypotheses and every F-formula to the conclusion we get back the original sequent.

Now let us take a closer look at the proof rules in the handout.

Q: $\boxed{\textit{What happens if we apply the same technique to the sequent rules?}}$
If we put H and G together into one set S we get exactly the same rules as in the block tableau calculus. And if we do it the other way around and split the set S into formulas marked with T and F then each block tableau rule becomes exactly the Gentzen rule described in the table.

Thus Gentzen Systems, although totally different in spirit, are actually *isomorphic* to block tableau. Technically the difference between the two is just a simple syntax transformation.

(1) We have the same starting point: $\vdash X$ corresponds to FX.

(2) We have exactly the same rules

(3) We have exactly the same nodes in our proof trees. Every sequent corresponds to a finite set of signed formulas. Conversely every set S of signed formulae corresponds to the sequent $H \vdash G$, where $H = \{X \mid T\mathrm{X} \in S\}$ and $G = \{X \mid F\mathrm{X} \in S\}$

(4) We have exactly the same condition for closing a proof branch: the `axiom` rule corresponds to the *-rule for closing block tableaux.

As an immediate consequence of this result we get

The multi-succedent sequent calculus is consistent and complete

Let me conclude by a few remarks about the semantical differences between tableaux and Gentzen systems. Although these two methods are formally almost identical, they have different intentions.

	Tableau	Sequent
Goal	FX: search for counterexample	$\vdash X$: try to prove X
α-step	counterexample must involve both α_1 and α_2 (AND branch)	proof must show either α_1 or α_2 (OR-branch)
β-step	counterexample uses either β_1 or β_2 (OR-branch)	proof must show both β_1 and β_2 (AND branch)
closed branch	counterexample impossible	partial proof sucessful
open leaf	consistent valuation possible (counterexample found)	proof fails at this leaf
	implicit proof of validity – no counterexample	explicit justification of validity
	negative / indirect approach	positive / constructive proof

But the technical similarity shows us that the indirect proof can be converted into a direct one. Actually, there are even methods which start way in the very compact matrix proofs and end up in the refinement logic that we will discuss next Tuesday.