Topic: Gentzen Systems, Multiple conclusioned Smullyan p.101–108

Assignments Exercises: Smullyan p.24 1–7 as matrix – Reading: Handouts

NEXT CLASS : 5130 Upson Hall

6 Gentzen Systems

6.1 The compactness theorem for deducibility

There is one, quite useful form of the compactness theorem which has to do with the notion of deducing a fact from certain assumptions. We say that X *is deducible from* S if for some $X_i \in S$ the formula $X_1 \wedge .. X_n \supset X$ is a tautology. Now the compactness theorem relates formal deducibility to the notion of *logical consequence*.

Theorem 6.1 (Compactness / Deduction Theorem)
If X is true in all valuations that satisfy S then X is deducible from S.

Since $S \cup \{\sim X\}$ is unsatisfiable there must be a finite subset $S_0 = \{X_1, ..., X_n, \sim X\}$ of $S \cup \{\sim X\}$, which is also unsatisfiable (compactness, not assuming S to be consistent). Thus $X_1 \wedge .. X_n \supset X$ must be a tautology.

O.k – so far for the theoretical foundations of propositional logic and tableau calculi. Today we want to discuss alternative calculi which are a bit better for computerized proofs – either by being more compact to allow more automated proof search, or by emphasizing the concept of logical deduction to support interaction.

6.2 Detour: Matrix Systems as compact description of tableaux

Use an example proof of $(((p \wedge q) \supset r) \supset ((\sim p \supset q) \supset r))^F$ for illustration (a failing proof). Write part of the tableaux proof on the blackboard

When you write down a tableaux proof there is too much copying of text involved. It would be more convenient, and also better for automatic proof search, to have a more compact presentation of tableaux proofs.

So what is really essential? Following Smullyans unified notation on page 21 we analyze only the *formula* instead of decomposing it by applying proof rules. For the sake of legibility we put T and F into a superscript, but otherwise the table is exactly identical to the tables on page 21.

type α successor	$(A \wedge B)^T$ A^T, B^T	$(A \vee B)^F$ A^F, B^F	$(A \supset B)^F$ A^T, B^F	$(\sim A)^T$ A^F	$(\sim A)^F$ A^T
type β successor	$(A \wedge B)^F$ A^F, B^F	$(A \vee B)^T$ A^T, B^T	$(A \supset B)^T$ A^F, B^T		

Next, we notice that for each formula X in a tableau, there is exactly ONE of the tableau rules that can be applied, namely the one that is described by the formulas type and polarity. It generates an immediate subformulae of X but it is not necessary to write this formula again. Instead, we simply add marks for ALL possible rule applications in a tableaux for X, by adding types and superscripts (polarities) to the syntax tree of X, starting with X^F and ending at the atoms.

Actually you can do that in the formula by moving the polarities and types inside and marking the connective, instead of the formula – that is easier to write.

- $((p \wedge q) \supset r) \supset_\alpha^F ((\sim p \supset q) \supset r)$
- $((p \wedge q) \supset_\beta^T r) \supset_\alpha^F ((\sim p \supset q) \supset_\alpha^F r)$
- $((p \wedge_\beta^F q) \supset_\beta^T r^T) \supset_\alpha^F ((\sim p \supset_\beta^T q) \supset_\alpha^F r^F)$
- $((p^F \wedge_\beta^F q^F) \supset_\beta^T r^T) \supset_\alpha^F ((\sim_\alpha^F p \supset_\beta^T q^T) \supset_\alpha^F r^F)$
- $((p^F \wedge_\beta^F q^F) \supset_\beta^T r^T) \supset_\alpha^F ((\sim_\alpha^F p^T \supset_\beta^T q^T) \supset_\alpha^F r^F)$
- In a tableaux proof we are only interested in closing branches / paths by finding atoms in that branch that marked both with T and F. So let us eliminate the polatities from the connectives – we don't need them anymore.

  $((p^F \wedge_\beta q^F) \supset_\beta r^T) \supset_\alpha ((\sim_\alpha p^F \supset_\beta q^T) \supset_\alpha r^F)$

  And besides, this F and T is too much writing, so we mark polarity F with an overbar and just drop the T.

  $((\bar{p} \wedge_\beta \bar{q}) \supset_\beta r) \supset_\alpha ((\sim_\alpha p \supset_\beta q) \supset_\alpha \bar{r})$

Next, we observe, that closing a path in a tableau proof only depends on the atoms, so it suffices to consider paths that only consist of atoms.
Well, two atomic formulae, let's say p and q, can only be on the same path, if the last rule that decomposed a formula in which both p and q occurred was an α-rule. Otherwise p and q would be on *different* paths, since β-rules always split a tableau into branches. So if we look at the formula, that means that the last connective that links p and q must be of type α. We call p and q *α-related* in this case.
Q: *So what is a path now?? (WAIT)*
A path is nothing but a set of α-related atoms of the formula. And a tableau proof shows that all possible sets of α-related atoms are closed. There is no need to write down all the subformulae. We only have to explore α-related sets.
Now we do the final trick. We write down the formula in a two-dimensional fashion, placing α-related atoms side by side and *β-related* ones on top of each other. If there is a possible ambiguity, we add brackets.
— *draw diagram for X^F* —
Now in this diagram, which we call *matrix*, a path is a horizontal line from left to right and it is closed, if it contains a connection between two *complementary* atoms, that is with the same variable but different polarities (one with bar, the other without).
The formula is not a tautology, if you can find a path that is not closed and the atoms on that path describe a counterexample. The nice thing is – you just read it from left to right and see what is going on.

Theorem 6.2 *A propositional formula X is a tautology iff every path through a matrix representation of X contains a complementary connection.*

For those who are interested: once you have this matrix-representation of a formula it is quite easy to program a method that explores all paths and checks whether they are closed or not. The neat thing is: you don't have to copy anything – you just keep in mind what you checked and what you didn't check yet. And if you want to do this efficiently, you try to close paths as quickly as possible by following the connections
– *illustrate with X^F*

Let us use another example p24: exa 8 $(p \vee (q \wedge r)) \supset; ((p \vee q) \wedge (p \vee r))$ (also on handout)
Gradually add polarities and types

- $(p \vee (q \wedge r)) \supset^F_\alpha ((p \vee q) \wedge (p \vee r))$
- $(p \vee^T_\beta (q \wedge r)) \supset^F_\alpha ((p \vee q) \wedge^F_\beta (p \vee r))$
- $(p^T \vee^T_\beta (q \wedge^T_\alpha r)) \supset^F_\alpha ((p \vee^F_\alpha q) \wedge^F_\beta (p \vee^F_\alpha r))$
- $(p^T \vee^T_\beta (q^T \wedge^T_\alpha r^T)) \supset^F_\alpha ((p^F \vee^F_\alpha q^F) \wedge^F_\beta (p^F \vee^F_\alpha r^F))$
- $(p \vee_\beta (q \wedge_\alpha r)) \supset_\alpha ((\bar{p} \vee_\alpha \bar{q}) \wedge_\beta (\bar{p} \vee_\alpha \bar{r}))$

Construct matrix and connections

$$\begin{bmatrix} [\,q \quad r\,] & [\,\bar{p} \quad \bar{q}\,] \\ p & [\,\bar{p} \quad \bar{r}\,] \end{bmatrix}$$

There are 4 possible paths – each contains exactly one complementary connection. In contrast to that – how long does it take just to write down the tableau.
Exercise: Do Smullyan p.24, 1–7 with matrices.

6.3 Quiz: 5 min

Which of the following choices is true?
If an infinite set S of formulae is unsatisfiable then

1. *All finite subsets of S are unsatisfiable*
2. *At least one finite subset of S is unsatisfiable but not necessarily all of them*
3. *Neither of the above*

6.4 Block Tableaux

The matrix description is very good for a global analysis of the validity of a formula because it is much more compact than a regular tableau. However, it does not really reflect on the idea of a *proof* , because we're checking something entirely different. The calculi, that we want to develop now, go into the other direction. They 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.
For this reason, they emphasize the concept of logical consequence much stronger, because that is what a human user of an interactive proof system can grasp much easier when he or she tries to reason about the validy of a formula. So, instead using proof rules that force us to look at a full tableau tree at once, we must design the calculus in a way that supports *local* reasoning. Essentially this means that we add all the formulae to a proof node that are above it in the tree but haven't been decomposed yet and can still be analyzed. We could also use simply all formulae above, but the once that were decomposed already, can't really give us any new information anymore
ILLUSTRATE: tableau proof for $(((p \supset q) \supset p) \supset p)$ and blocked tableau
So what we essentially do is to use a tableau calculus that operates on SETS of formulae. Smullyan calls this a block tableaux. The modifications for this calculus are simple.

1. The root of a tableau tree is now a finite set of formulae
2. There are α and β rules

$$\begin{array}{c} \boxed{S, \alpha} \\ | \\ \boxed{S, \alpha_1, \alpha_2} \end{array} \qquad \begin{array}{c} \boxed{S, \beta} \\ \diagup \quad \diagdown \\ \boxed{S, \beta_1} \quad \boxed{S, \beta_2} \end{array}$$

   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.

Actually Smullyan has two α-rules, one that add α_1 and the other add α_2 only. But this is not really necessary because adding the other α-successor does not affect whether a branch can be closed or not.
Now if we spell these rules out for each logical connective, and write the rules in a less graphical fashion, but simply use one line for each generated branch, you 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.

left	right
α S, T(A∧B) S, T(A), T(B)	S, F(A∧B) β S, F(A) S, F(B)
β S, T(A∨B) S, T(A) S, T(B)	S, F(A∨B) α S, F(A) , F(B)
β S, T(A⊃B) S, F(A) S, T(B)	S, F(A⊃B) α S, T(A), F(B)
α S, T(∼A) S, T(A)	S, F(∼A) S, F(A) α
* S, T(A), F(A)	

Example: Block Tableau for F$((p \wedge q) \supset (q \vee p))$ *— show also the analytic tableau*
It is very straightforward to see how to convert a conventional tableau for a single formula into a block tableau for that formula, as the above example showed. You just keep all the formulae in the set, that have been created before and not been decomposed yet. Otherwise it is a *one-to-one simulation*, because we use the same rules at the same nodes (except that our α-rule creates one successor where the analytic rule creates two). If the convential tableau is closed, so is the block tableau. If it is open, so is the block tableau. So we know that block tableaux are complete because:

Analytic standard tableaux can be simulated by block tableau

Similarly we can show correctness or consistency by simulating each block tableau by a regular analytic tableau: again we use the same rules at the same nodes, and notice that α-rules create two successors instead of one. Thus we get.

The block tableau calculus is consistent and complete

The same simulation also works for analytic tableaux that operate on sets of formulae (see Smullyan p.33, last Tuesday).

6.5 Gentzen Systems

Now block tableaux are only a step towards a more intuitive proof method, namely *Gentzen Sequents*. While the tableau method tries to find a *counterexample* for the validity of a formula X or shows that there can't be one, Gentzen sequents aim at *proving* X. We write therefore consider the formula X as *proof goal* and denote this by $\vdash X$.
– *in the process write down the rules of the handout* —
To prove a formulae, we use again proof rules, which can easily be motivated by intuition. To prove $\vdash A \wedge B$, for instance we have to prove both A and B, that is we will get $\vdash A$ and $\vdash B$ as *subgoals* of our proof that still have to be solved. To prove $\vdash A \Rightarrow B$ we assume that A is true and have to prove B. We write this as $A \vdash B$ because the turnstyle symbolizes that B can be deduced/proven from B.
So we see that some proof rules can create *assumptions* and thus we also need rules that tell us how to deal with these assumptions. For instance, if we have the formula $A \wedge B$ as assumption, we may split it into two parts and use A and B as separate assumptions. If we have $A \vee B$ as assumption for proving a goal G, we may use either A or B to prove G. We get two subgoals in this case.
The rule for dealing with the assumption $A \Rightarrow B$ when proving a goal G is more tricky. We can decompose this assumption only, if we have already proven A. In this case we know that we have B and can use this as assumption for proving G. So we get two subgoals: in one we have to prove A in the other we prove G under the assumption B. Well, in the first case we do not want to throw away the goal G, but rather be careful and keep it as *alternative proof goal* – we will see next week that we don't actually need this, but it is much more difficult to show that our calculus is complete if we throw away the goal G.
Therefore we do not only allow several assumptions but also multiple goals, or *conclusions*. So sequents are in the end defined as objects of the form $\Delta \vdash \Gamma$ where both Δ and Γ are *sets of formulae*. (Technically a sequent is an ordered pair $<\Delta,\Gamma>$ of sets of formulae). We read such a sequent as "under the assumption that all the formulae in Δ are true we can show that one of the formulae in Γ is true" and Gentzen's proof rules show how to do that.

	left	right	
`andL`	$\Delta, A \wedge B \vdash \Gamma$ $\Delta, A, B \vdash \Gamma$	$\Delta \vdash \Gamma, A \wedge B$ $\Delta \vdash \Gamma, A$ $\Delta \vdash \Gamma, B$	`andR`
`orL`	$\Delta, A \vee B \vdash \Gamma$ $\Delta, A \vdash \Gamma$ $\Delta, B \vdash \Gamma$	$\Delta \vdash \Gamma, A \vee B$ $\Delta \vdash \Gamma, A, B$	`orR`
`impL`	$\Delta, A \Rightarrow B \vdash \Gamma$ $\Delta \vdash \Gamma, A$ $\Delta, B \vdash \Gamma$	$\Delta \vdash \Gamma, A \Rightarrow B$ $\Delta, A \vdash \Gamma, B$	`impR`
`notL`	$\Delta, \neg A \vdash \Gamma$ $\Delta \vdash \Gamma, A$	$\Delta \vdash \Gamma, \neg A$ $\Delta, A \vdash \Gamma$	`notR`
`axiom`	$\Delta, A \vdash \Gamma, A$		

You have all the rules in the handout. Let me go over those that I haven't explained yet. (orL, notL, notR) . . .
The `axiom` rule finally shows how to complete a proof. If a goal formula occurs in the assumptions then we are done, we assumed it to be true, so it is proven to be true.

Now, how do we show that the sequent proof method is consistent and complete. I mean, intuitively it is clear why the rules are correct but that is not enough to convince ourselves that we haven't overlooked a subtle mistake.
So we could go on and do the same thing as we did with tableaux: we extend the notion of boolean valuations to sequents. $\Delta \vdash \Gamma$ is true under v_0 iff there is an $\mathrm{Y} \in \Gamma$ such that val(Y,v_0) = t, whenever val(X,v_0) = t for all $\mathrm{X} \in \Delta$. Similarly we could now define the notions of satisfiability and tautology, and then go on to prove that a sequent $\Delta \vdash \Gamma$ is a tautology if and only if there is a sequent proof for it.

There is a much simpler way, because of a similarity between sequents and blocked tableaux. This similarity is so strong that we can call it a *duality*. Let us explore this for a moment.
We know that in order to prove a formula Y_i the tableau method tries to find a closed tableau for $\mathsf{F}(Y_i)$. We also know that the sequent $\{X_1, ...X_n\} \vdash \{Y_1, ...Y_m\}$ pretty much expresses the same as the formula $(X_1 \wedge ...X_n) \Rightarrow (Y_1 \vee ...Y_m)$. So in order to prove this formula, the tableau method would, fater a few steps generate the finite set of signed formulae $\{\mathsf{T}(X_1), ...\mathsf{T}(X_n), \mathsf{F}(Y_1), ...\mathsf{F}(Y_m)\}$.
Now let us look at the sequent rules again: if we convert every formula X in the hypotheses into $\mathsf{T}(X)$ and every formula Y in the conclusion into $\mathsf{F}(Y)$, put Δ and Γ together into one set and forget about the turnstyle – what do we get?
– yes – we get exactly the same rules as in the blocked tableau calculus –
So Gentzen systems are technically only a different notation for blocked tableaux.

1. We have the same starting point: $\vdash X$ corresponds to $\mathsf{F}(X)$.
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 formulae. Conversely every set S of signed formulae corresponds to the sequent $\Delta \vdash \Gamma$, where $\Delta = \{X | \mathsf{T}(X) \in S\}$ and $\Gamma = \{X | \mathsf{F}(X) \in S\}$
4. We have exactly the same condition for closing a proof branch: the `axiom` rule corresponds to the *-rule for closing blocked tableaux

So every sequent proof can be directly simulated by a blocked tableau and vice versa. And as a 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 thses two methods are formally almost identical, they have different intentions.

Tableau	Sequent	
Goal	$\mathsf{F}(X)$: search for counterexample	$\vdash X$: try to prove X
α-step	counterexample must involve both α_1 and α_2	proof must show either α_1 or α_2
β-step	counterexample users either β_1 or β_2	proof must show both β_1 and β_2
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.

Next tuesday we want to go one step further and talk about a Gentzen system that doesn't use multiple succedents. This *refinement logic* has many properties that are interesting for a computer scientists, because it catches the notion of a construction in a very nice and natural way. It is also the logic that we use in our NuPRL system and I will show you how we can use this system to support and automate the development of formal proofs. Therefore the next class will be in **5130 Upson Hall**. We have a nice computer projector installed in that room so I can give you a few demos.