GENTZEN-TYPE SYSTEMS, RESOLUTION AND TABLEAUX

Arnon Avron

Computer Science Department
Raymond and Beverly Sackler
Faculty of Exact Sciences
Tel Aviv University
Tel Aviv, Israel

E-mail address:aa@math.tau.ac.il

GENTZEN-TYPE SYSTEMS, RESOLUTION AND TABLEAUX

I. Introduction

In advanced books and courses on logic (e.g. [Sm], [BM]) Gentzen-type systems or their dual, tableaux, are described as techniques for showing validity of formulae which are more practical than the usual Hilbert-type formalisms. People who have learnt these methods often wonder why the Automated Reasoning community seems to ignore them and prefers instead the resolution method. Some of the classical books on AD (such as [CL], [Lo]) do not mention these methods at all. Others (such as [Ro]) do, but the connections and reasons for preference remain unclear after reading them (at least to the present author, and obviously also the authors of [OS], in which a theorem-prover, based exclusively on tableaux, is described). The confusion becomes greater when the reader is introduced to Kowalski's form of a clause ([Ko], [Bu]), which is nothing but a Gentzen's sequent of atomic formulae, and when he realizes that resolution is just a form of a Cut, and so that while the *elimination* of cuts is the principal tool in proof-theory, its *use* is the main technique in AD!

It is one of the purposes of this paper to explain the deep connections between Gentzen-type systems, tableaux and resolution. We show that both resolution and tableaux are based on attempts to exploit the power of cut-elimination theorems in Gentzen-type calculi and explain in which cases each should be preferred[1]. We provide purely syntactical proofs to all our claims, including the major results about resolution (which are usually proved by semantical considerations elsewhere). This is important for the goal of applying similar techniques to other logics, which do not have the simple semantics that classical logic has, but do have decent Gentzen-type formulations. (It is indeed easy to get from our work precise criteria for when the existence of good resolution-like proof techniques might be expected.) Finally, we suggest how the two methods (resolution and tableaux) can be

[1] I am sure that much (if not all) of the relevant material is already known. I was unable, however, to find it written down in a systematic way (something that would have saved me time and might save the time of others in the future). Still, [Ga] contains important material on Gentzen-type systems and resolution (but from a different point of view), and crucial hints can be found in [Gi1, sect. 13.4]. That section is mainly devoted to the case of Horn clauses. Nevertheless, many of the ideas and results below (though they have been found independently) are either implicit in it or proved in [Gi2, sect. 2.7].

combined to get more efficiency. Thus we show how tableaux can be used efficiently for directly converting a formula (or a sequent) into clause form, without converting it first to prenex normal form. Another major possibility that we show is that a preparatory work with tableaux might help splitting the work to be done by resolution on one set of clauses into separate works on several, smaller ones. This would certainly make the search for refutation in each shorter. It might also open the door for working in parallel and for exploiting considerations of symmetry (see example below).

II. Gentzen-type Systems

Let L be a formal language in which the notion of a well-formed-formula (wff) is defined. A Gentzen-type calculus in L is first of all an axiomatic system which manipulates higher-level constructs called *sequents*, rather than the formulae themselves. There are several variants of what exactly constitutes a sequent. Here it is convenient to take it to be a construct of the form $\Gamma \Rightarrow \Delta$, where Γ, Δ are finite *sets* of formulae of L and $\Rightarrow$ is a new symbol, not occurring in L. [2] The following two basic features characterize the Gentzen type formalisms which are based on this notion:

1) $A \Rightarrow A$ should be provable for every formula A[3].

2) The following "*cut*" rule should be valid:

$$\frac{\Gamma_1 \Rightarrow \Delta_1 \cup \{A\} \qquad \{A\} \cup \Gamma_2 \Rightarrow \Delta_2}{\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2}$$

Most Gentzen-type formalisms also satisfy the following demand:

Monotonicity: The system is closed under the *Weakening* rule:

$$(W) \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}$$

Another important property of Gentzen-type systems is given in the following:

Definition 1: A rule of a monotonic Gentzen-type system is called *pure*[4] if whenever $\Gamma \Rightarrow \Delta$ can be inferred by it from $\Gamma_i \Rightarrow \Delta_i$ $(i = 1, \ldots, n)$ then $\Gamma, \Gamma' \Rightarrow \Delta, \Delta'$ can also be

[2] In other variants Γ, Δ may be either multisets or sequences of formulae. Having only one-sided sequents is another possibility.

[3] Officially we should have written $\{A\} \Rightarrow \{A\}$. We shall, however, follow tradition and omit the curly brackets from both sides of $\Rightarrow$. Also, we shall usually write Γ, Δ for $\Gamma \cup \Delta$, etc.

[4] This is a variant of a notion which was first introduced in [Av].

inferred by it from $\Gamma_i, \Gamma' \Rightarrow \Delta_i, \Delta'$ ($i = 1, \ldots, n$; Γ', Δ' – arbitrary sets of formulae). A Gentzen-type system is called *pure* if all its rules are pure.

A Gentzen-type system, G, directly defines a Consequence Relation (C.R.) $\vdash_G$ between *sequents*. Usually, however, it is mainly used as a tool for investigating C.R.s between the *formulae* of L. There are two standard ways of using G for defining such C.R.s:

Definition 2: Let G be a Gentzen-type calculus

(1) $A_1, \ldots, A_n \vdash^t_G B$ iff $\vdash_G A_1, \ldots, A_m \Rightarrow B$

(2) $A_1, \ldots, A_n \vdash^v_G B$ iff $(\Rightarrow A_1), , \ldots, (\Rightarrow A_n) \vdash_G (\Rightarrow B)$

Note that $\vdash^t_G B$ iff $\vdash^v_G B$. The difference is when there are assumptions. Other basic connections between the two C.R.s are given in Proposition 1.

Proposition 1. *If $\Gamma \vdash^t_G B$ then $\Gamma \vdash^v_G B$. If G is monotonic and pure*[5] *then the converse is also true.*

Proof: For the first part, use cuts. For the converse, suppose that $\Gamma = \{A_1, \ldots, A_n\}$ and that $(\Rightarrow A_1), (\Rightarrow A_2), \ldots, (\Rightarrow A_n) \vdash_G \Rightarrow B$. Since G is pure this entails that $(\Gamma \Rightarrow A_1), (\Gamma \Rightarrow A_2), \ldots, (\Gamma \Rightarrow A_n) \vdash_G \Gamma \Rightarrow B$. But $\vdash_G A_i \Rightarrow A_i$ and G is monotonic. Hence $\vdash_G \Gamma \Rightarrow A_i$ ($i = 1, \ldots, n$), and so $\vdash_G \Gamma \Rightarrow B$.

Our next proposition is crucial for understanding the method of resolution:

Proposition 2. *Suppose G is monotonic and pure. Then $\vdash_G A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m$ iff $(\Rightarrow A_1), \ldots, (\Rightarrow A_n), (B_1 \Rightarrow), \ldots, (B_m \Rightarrow) \vdash_G \Rightarrow$.*

Proof: Similar.

Notes: 1) For the vailidity of Proposition 2 we need purity on *both* sides. The proposition indeed fails for the intuitionistic calculus, in which $\not\vdash \neg\neg p \Rightarrow p$ but $(p \Rightarrow), (\Rightarrow \neg\neg p) \vdash \Rightarrow$.

2) One important corollary of Proposition 2 is that $\vdash_G \Rightarrow A$ iff $(A \Rightarrow) \vdash_G \Rightarrow$. In other words A is valid if the assumption that it is false is contradictory. In case L has internal negation $\neg$ and internal falsehood $\bot$[6] this can be formally expressed as: $\vdash^t_G A$ iff $\neg A \vdash^v_G \bot$.

[5] It is enough to assume monotonicity and purity on the l.h.s. Hence, the proposition is true, e,g., also for the intuitionistic Gentzen calculus.

[6] See [Av] for definitions.

III. Classical Propositional Calculus

We turn now to investigate the special case of GCPL – the Gentzen-type system for Classical Propositional Logic (CPL) – in which most of the connections between tableaux and resolution are already reflected. The version we use here is the following:

The System GCPL

Axioms: $A \Rightarrow A$

Structural Rules: Cut, Weakening (W)

Logical Rules:

$$(\neg \Rightarrow) \quad \frac{\Gamma \Rightarrow \Delta, A}{\neg A, \Gamma \Rightarrow \Delta} \qquad \frac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A} \quad (\Rightarrow \neg)$$

$$(\rightarrow\Rightarrow) \quad \frac{\Gamma \Rightarrow \Delta, A \quad B, \Gamma \Rightarrow \Delta}{A \rightarrow B, \Gamma \Rightarrow \Delta} \qquad \frac{\Gamma, A \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \rightarrow B} \quad (\Rightarrow\rightarrow)$$

$$(\wedge \Rightarrow) \quad \frac{\Gamma, A, B \Rightarrow \Delta}{\Gamma, A \wedge B \Rightarrow \Delta} \qquad \frac{\Gamma \Rightarrow \Delta, A \qquad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \wedge B} \quad (\Rightarrow \wedge)$$

$$(\vee \Rightarrow) \quad \frac{\Gamma, A \Rightarrow \Delta \qquad \Gamma, B \Rightarrow \Delta}{\Gamma, A \vee B \Rightarrow \Delta} \qquad \frac{\Gamma \Rightarrow \Delta, A, B}{\Gamma \Rightarrow \Delta, A \vee B} \quad (\Rightarrow \vee)$$

It is not difficult to show soundness and completeness of the above system w.r.t. the standard two-valued semantics, but we shall not need these results here. What *is* important for us is the obvious fact that GCPL is both monotonic and pure. Hence Propositions 1 and 2 apply to it. In particular: $\vdash^{v}_{\mathrm{GCPL}} = \vdash^{t}_{\mathrm{GCPL}}$.

Notation: Until the end of this section we shall use $\vdash$ for both $\vdash_{\mathrm{GCPL}}$ (the C.R. between sequents) and $\vdash^{t}_{\mathrm{GCPL}}$.

Perhaps the most characteristic property of the logical rules of GCPL is that all of them are invertible. In other words, the premises of each such rule can be deduced from its conclusion. By this we mean not only that the premises should necessarily be provable whenever the conclusion is, but that they *follow* from it. For example: $(\Gamma \Rightarrow \Delta, A \rightarrow B) \vdash (A, \Gamma \Rightarrow \Delta, B)$, since the r.h.s. can be deduced from the l.h.s. using the provable sequent $A, A \rightarrow B \Rightarrow B$ and a cut. Similar considerations apply for the other rules.

The invertibility of the rules has an immediate important corollary. Before stating it we need a definition.

Definition 3: A *Clause* is a sequent which consists solely of atomic formulae[7].

[7] This is known as "Kowalski's form" of a clause.

Proposition 3. *Every sequent is equivalent to a set of clauses (by "equivalence" we mean that every clause in the set is deducible from the original sequent while the sequent itself is deducible from the set as a whole).*

Proof: By induction on the complexity of the sequent.

Note: A set of clauses which is equivalent to a given sequent actually represents a conjunctive normal form (CNF) of it. Indeed, a clause $p_1, \ldots, p_n \Rightarrow q_1, \ldots, q_m$ is equivalent to $\Rightarrow \neg p_1 \vee \ldots \vee \neg p_n \vee q_1 \vee \ldots \vee q_m$, while a set of sequents of the form $\{(\Rightarrow A_1), \ldots, (\Rightarrow A_\ell)\}$ is equivalent to the sequent $\Rightarrow A_1 \wedge \ldots \wedge A_\ell$. It follows, in particular, that every sentence is equivalent to a sentence in CNF.

The proof of Proposition 3 provides a constructive method for reducing a sequent $\Gamma \Rightarrow \Delta$ to an equivalent set of clauses. Thus, for example, if $A \rightarrow B \in \Delta$ we replace $\Gamma \Rightarrow \Delta$ by $\{A\} \cup \Gamma \Rightarrow \{B\} \cup \big(\Delta \perp \{A \rightarrow B\}\big)$ and continue the reduction. If, on the other hand, Γ contains $A \rightarrow B$ we replace it by *two* sequents: $\{B\} \cup \big(\Gamma \perp \{A \rightarrow B\}\big) \Rightarrow \Delta$ and $\Gamma \perp \{A \rightarrow B\} \Rightarrow \Delta \cup \{A\}$, and reduce each.

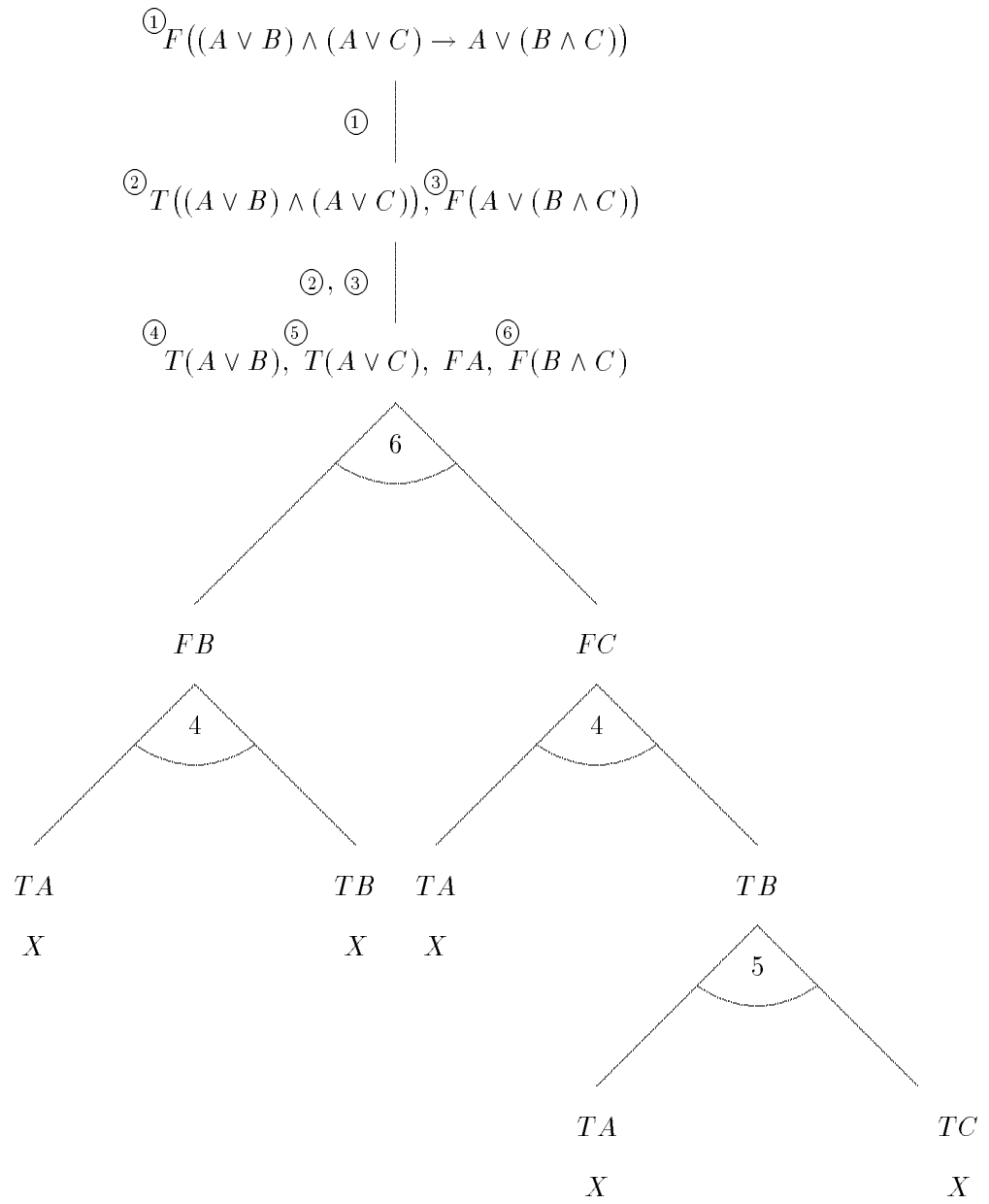
The process we have just described is the essence of the method of tableaux for determining whether $A_1, \ldots, A_n \vdash B$ in classical propositional logic. What is actually done there is to check whether $A_1, \ldots, A_n \Rightarrow B$ is a theorem of GCPL. For this we reduce it first to clause form, and then check each of the resulting clauses for provability. The idea is to take advantage of the fact that it is very easy to determine provability of clauses: a clause $\Gamma \Rightarrow \Delta$ is provable iff $\Gamma \cap \Delta \neq \emptyset$. This claim is obvious by semantical considerations, but it is also an immediate corollary of the famous cut-elimination theorem of Gentzen [Ge], according to which every provable sequent has a proof without cuts.[8]

When it comes to practice the method of tableaux proceeds as follows: Instead of working with a tree of sequents, we work with a tree of finite sets of *signed* formulae of the form TA or FA. (A – a usual formula.) This tree is expanded systematically. At any stage of the expansion each of its branches represents a sequent. A non-marked occurrence of TA on a branch means that A is on the l.h.s. of the corresponding sequent, while such an occurrence of FA means that it is on the r.h.s. A signed formula is marked when a reduction step is applied to it. Once a branch contains both TA and FA for some A it is discarded (and so it is not expanded any more). If all branches are discarded in this way

[8] A purely syntactic proof of this theorem can be found in [Ge], [Gi2] or [Ga].

then the original sequent is provable.[9]

Example 1: Prove $(A \vee B) \wedge (A \vee C) \rightarrow A \vee (B \wedge C)$:



The resolution method for determining the provability of a sentence A is dual to the method of tableaux. Instead of checking whether $\vdash \Rightarrow A$, it uses Proposition 2 and checks whether $(A \Rightarrow) \vdash \Rightarrow$. For this we reduce $A \Rightarrow$ (rather than $\Rightarrow A$) to an equivalent set S of clauses[10]. It remains then to check that S is inconsistent. This relies on the fact that

[9] For someone who hears about tableaux for the first time this description might be insufficient, and we refer him to [Smu] for further details and explanations.

[10] This is equivalent to finding a clause form for $\neg A$.

inconsistency too is easier to prove for a set of clauses than for an arbitrary set or sequents. The crucial point this time is that for a set of clauses cuts (or resolutions[11]) suffice for showing inconsistency. This is a corollary of the following straightforward (though less known) generalization of the cut-elimination theorem:

The strong cut-elimination theorem[12]. *If $S \vdash \Gamma \Rightarrow \Delta$ then there is a proof of $\Gamma \Rightarrow \Delta$ from S in which every cut is made on a formula which occurs in some sequent of S (in particular, the case $S = \emptyset$ is just the standard cut-elimination theorem of Gentzen).*

First Proof: Just like Gentzen's proof of the special case in [Ge], only no reduction can be made if one of the premises of the cuts is an assumption (from S).

Second Proof: By induction on the number of clauses in S. The case $n = 0$ is just Gentzen's theorem. For the induction step we use the fact that if $S = S' \cup \{\Gamma' \Rightarrow \Delta'\}$ and $S \vdash \Gamma \Rightarrow \Delta$ then $S' \vdash \Gamma \Rightarrow \Delta, B$ and $S' \vdash A, \Gamma \Rightarrow \Delta$ for every $A \in \Delta'$ and $B \in \Gamma'$ (this follows from the purity of the rules and provability of $A, \Gamma' \Rightarrow \Delta'$ and $\Gamma' \rightarrow \Delta', B$ for every A and B as above). By applying the induction hypothesis to S', we get proofs as required for the various $\Gamma \Rightarrow \Delta, B$ and $A, \Gamma \Rightarrow \Delta$. $\Gamma \Rightarrow \Delta$ can then be inferred from the sequents and $\Gamma' \Rightarrow \Delta'$ by cuts.

Note: The second proof suggests a further strengthening: Let "hyper-resolution" be the following rule:

$$\frac{(A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m), (\Gamma_1 \Rightarrow \Delta_1, A_1), \ldots, (\Gamma_n \Rightarrow \Delta_n, A_n), (\Gamma'_1, B_1 \Rightarrow \Delta'_1), \ldots, (\Gamma'_m, B_m \Rightarrow \Delta'_m)}{\Gamma_1, \Gamma_2, \ldots, \Gamma_n, \Gamma'_1, \ldots, \Gamma'_m \Rightarrow \Delta_1, \ldots, \Delta_n, \Delta'_1, \ldots, \Delta'_m}$$

Call $A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m$ the "nucleus" of the rule. Then $S \vdash \Gamma \Rightarrow \Delta$ iff there is a proof of $\Gamma \Rightarrow \Delta$ from S which uses only logical rules, weakenings and hyper-resolutions with elements of S as nucleii.

We next use the last theorem to prove the completeness of the resolution principle.

Definition: A normal clause is a clause $\Gamma \Rightarrow \Delta$ in which $\Gamma \cap \Delta = \emptyset$.

Proposition 4. *Let S be a set of clauses, $\Gamma \Rightarrow \Delta$ a normal clause and suppose $S \vdash \Gamma \Rightarrow \Delta$. Then there are $\Gamma' \subseteq \Gamma$, $\Delta' \subseteq \Delta$ and $S' \subseteq S$ such that every element in S' is normal, none*

[11] On the propositional level there is no difference.

[12] To the best of my knowledge, the first to observe this generalization, as well as its relevance to resolution, was Girard. See [Gi1, sect. 13.4], [Gi2, sect. 2.7].

of them is subsumed by another, and there is a proof of $\Gamma' \Rightarrow \Delta'$ *from* S' *in which the only rules used are cuts on formulae that occur in* S'.

Proof: By the previous theorem there is a proof P of $\Gamma \Rightarrow \Delta$ from S in which every cut used is on a formula in S, i.e., atomic formula. Since $\Gamma \Rightarrow \Delta$ also contains only atomic formulae no nonatomic formula can occur in P. Hence the only rules used in P are cuts on formulae in S and weakenings. With this knowledge, the proposition can easily be proved by induction on the length of P.

Corollary 1 (Completeness of resolution in the propositional case). *If* S *is an inconsistent set of clauses then there is a subset of normal clauses of* S *from which* $\Rightarrow$ *can be derived using only cuts.*

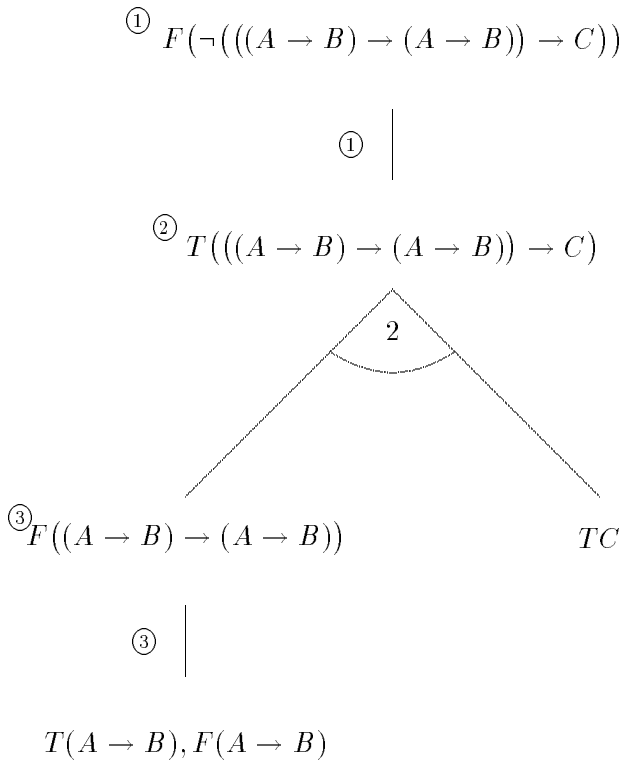
How do we prove the general claim $A_1, \ldots, A_n \vdash B$ using the resolution methodΓ In many textbooks this problem is reduced to showing that the sentence $A_1 \wedge \ldots \wedge A_n \rightarrow B$ is provable. Hence they start by reducing $A_1 \wedge \ldots A_n \rightarrow B \Rightarrow$ to a set of clauses. Using the tableaux rules this sequent is reduced, first of all, to the sequents $(\Rightarrow A_1), \ldots, (\Rightarrow A_n)$ and $(B \Rightarrow)$ (in some textbooks it is indeed recommended to find the clause form of $\neg(A_1 \wedge \ldots \wedge A_n \rightarrow B)$ by taking the union of the clause forms of $A_1, \ldots, A_n$ and $\neg B$!). It is worth noting, however, that if we translate "$A_1, \ldots, A_n \vdash B$" directly to "$\vdash A_1, \ldots, A_n \Rightarrow B$" and apply Proposition 2 we get the same reduction directly, without the detour via $A_1 \wedge \ldots \wedge A_n \rightarrow B$.

Let us summarize what we have seen about the relations in CPL between the methods of resolution and of tableaux. First, the validity of both is a consequence of the (strong) cut-elimination theorem. Second, for showing the validity of a sentence A both start by reducing an appropriate simple sequent to clause form ($\Rightarrow A$ in tableaux, $A \Rightarrow$ in resolution). Third, in both, at the final stage, we use special properties that clauses have in the context of GCPL. In the case of tableaux this is a simple criterion for provability of one clause, while in resolution – a criterion for the inconsistency of a *set* of clauses. Now, *on average*, reducing sequents of the form $A \Rightarrow$ and $\Rightarrow A$ to clause forms are equally difficult tasks. In most cases, on the other hand, the work needed on the resulting set of clauses is smaller in the tableaux case, since the work with resolution requires search. Moreover, in case A is *not* valid, the work with tableaux will stop as soon as we find a normal clause among the clauses that correspond to A (this might happen even before they have all been found!). In the case of resolution we would be forced to try all possibilities before concluding that

A is not valid. Another advantage of tableaux is that usually we need not, in fact, reach a complete clause form while applying it. We conclude therefore that in the context of CPL the tableaux method is superior to the resolution method.

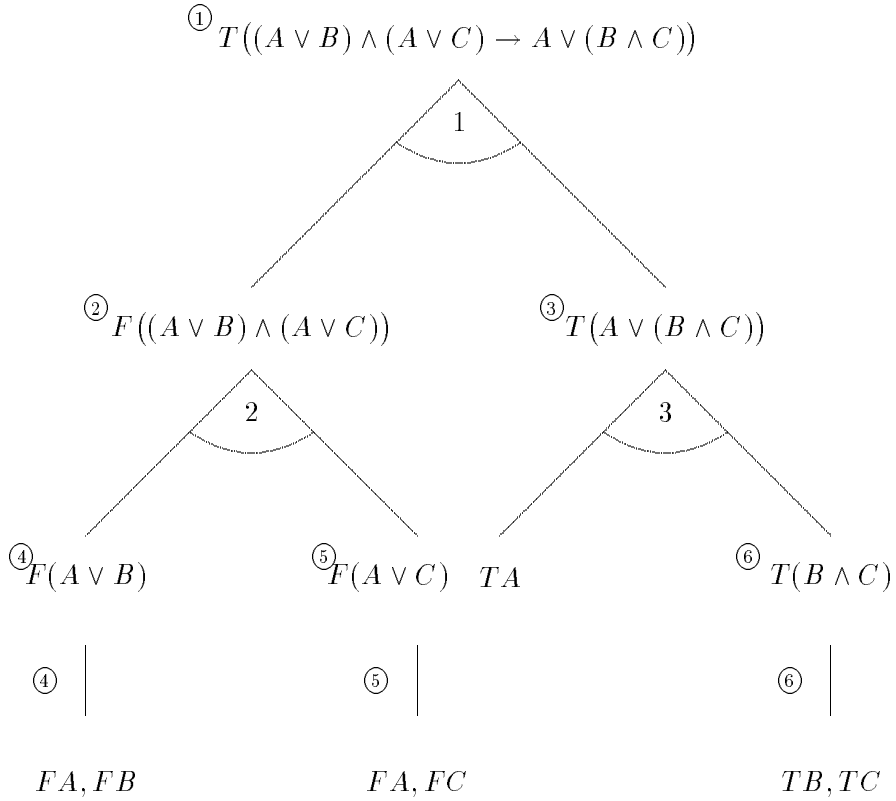
Another important point to note here is the following: As we have seen, both methods start by finding a set of clauses which is equivalent to a given sentence or sequent. For both methods we should obviously use the most efficient way of achieving this goal. Now in [CL] and [Lo] the suggested algorithm requires, first of all, the elimination of the implication connective and then to apply a certain list of equivalences (including, e.g., those of De-Morgan). This is rather unfortunate. As the example of $A_1 \wedge \ldots \wedge A_n \rightarrow B$ above indicates, the tableaux method for doing this is better! In fact tableaux are, above all, *the best way known to logicians to find a clause form (or a CNF) to a given sentence or sequent*. Now, the discarding of a branch as soon as both TA and FA occur on it for some A is one way in which this method frequently saves efforts and produces economical clauses forms. If *all* branches happen to be discarded then we conclude that the starting formula (or sequent) is valid, but this is just a special case!

Example 2: Reduce $\neg\big(\big((A \rightarrow B) \rightarrow (A \rightarrow B)\big) \rightarrow C\big)$ to clause form:



The left branch has been discarded before applying further reductions on $A \to B$. On the other, only TC remains unreduced. The clause form is therefore the singleton $\{(C \Rightarrow)\}$.

Example 3: Use resolution to prove $(A \vee B) \wedge (A \vee C) \to A \vee (B \wedge C)$



We got the following set of clauses : $\{(\Rightarrow A, B), (\Rightarrow A, C), (A \Rightarrow), (B, C \Rightarrow)\}$. It is easy to derive $\Rightarrow$ from it using 4 resolutions.

Despite what we have said above on the superiority of the method of tableaux, there are clearly cases in which reducing $A \Rightarrow$ to clause form is faster then reducing $\Rightarrow A$. This might well compensate for the extra work that is usually needed at the end. A clear case of this sort is when A is given in a (not too simple) disjunctive normal form. Reducing $A \Rightarrow$ is easy then, while reducing $\Rightarrow A$ might be tedious. The reason is that reducing $\Rightarrow A$ might involve a lot of $(\Rightarrow \wedge)$-reductions, which leads to a great deal of branching and repetitions. In general too many reductions which involve branching might slow down the reduction process considerably.

This brings us to another important point in our discussion. In many cases the best way to proceed is by a *combination* of the two methods. For this we have, first of all, to see them both in the more general context of proving arbitrary *sequents*, not only sentences (or even sequents of the form $\Gamma \Rightarrow A$). To show then that a sequent $A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m$

is provable, the tableaux method directly tries to find a proof of it, while the resolution method again tries, by Proposition 2, to show that the set $\{(\Rightarrow A_1), \ldots, (\Rightarrow A_n), (B_1 \Rightarrow), \ldots, (B_m \Rightarrow)\}$ is inconsistent. One possible way to combine them might proceed as follows: take the sequent to be proved and apply as many non-branching tableaux reductions to it as you can. Apply then the resolution method to the sequent you have got. (Note that since sides are switched when we start doing this, the reduction of *all* formulae will at least *start* with non-branching reductions!)

Example 4: Prove $(A \vee B) \wedge (A \vee C) \rightarrow A \vee (B \wedge C)$

Stage 1:

$$F\big((A \vee B) \wedge (A \vee C) \rightarrow A \vee (B \wedge C)\big)$$

$$\Big|$$

$$T\big((A \vee B) \wedge (A \vee C)\big), F\big(A \vee (B \wedge C)\big)$$

$$\Big|$$

$$T(A \vee B), T(A \vee C), FA, F(B \wedge C)$$

Stage 2: Interchange F and T. Reduce each formula to clause form:

$$\begin{array}{cccc} F(A \vee B) & F(A \vee C) & TA & T(B \wedge C) \\ \Big| & \Big| & & \Big| \\ FA, FB & FA, FC & & TB, TC \end{array}$$

We got the following set of clauses: $\{(\Rightarrow A, B), (\Rightarrow A, C), (A \Rightarrow), (B, C \Rightarrow)\}$ to which we apply cuts (compare Example 3).

A more radical approach is to use some *branching* tableaux reductions as well and so get *two or more* sequents that *together* are equivalent to the original one. The resolution method can then be applied to each of them separately.

Example 5: Let us return to the previous example, and suppose that at the end of Stage 1 we apply one more reduction to $F(B \wedge C)$. We get then two branches, one representing $A \vee B, A \vee C \Rightarrow A, B$ the other $A \vee B, A \vee C \Rightarrow A, C$. At Stage 2 we get, accordingly, two sets of clauses $\{(\Rightarrow A, B), (\Rightarrow A, C), (A \Rightarrow), (B \Rightarrow)\}$ and $\{(\Rightarrow A, B), (\Rightarrow A, C), (A \Rightarrow), (C \Rightarrow)\}$. ¿From each of them one sequent can immediately be deleted as useless by standard criteria, and two cuts suffice to get $\Rightarrow$ from the others.

Note: The two sets of clauses are obviously symmetrical with respect to B and C. A human logician would immediately realize, therefore, that it suffices to prove the inconsistency of only one of them (in fact, he would already realize this symmetry at the end of Stage 1 and

thereby save even more work). It is a subject of current research in what cases and how an automated system will also be able to take advantage of such considerations of symmetry.

The possibility of splitting the work done by resolution into several smaller jobs is rather interesting. We should admit, however, that we are just at the beginning of exploring the questions how best to choose the point of switching and what branching reductions to apply before that.

IV. Classical Predicate Calculus

We start with GCL – the Gentzen-type system for classical logic that we are going to use. It is obtained from GCPL by the addition of the following 4 rules:

$$(\forall \Rightarrow) \quad \frac{\Gamma, A(t/x) \Rightarrow \Delta}{\Gamma, \forall x A \Rightarrow \Delta} \qquad {}^{(*)}\frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta \quad \forall y A(y/x)} \quad (\Rightarrow \forall)$$

$${}^{(*)}(\exists \Rightarrow) \quad \frac{\Gamma, A \Rightarrow \Delta}{\Gamma, \exists y A(y/x) \Rightarrow \Delta} \qquad \frac{\Gamma \Rightarrow \Delta, A(t/x)}{\Gamma \Rightarrow \Delta, \exists x A} \quad (\Rightarrow \exists)$$

$(*)$ In $(\Rightarrow \forall)$ and $(\exists \Rightarrow)$ the eigenvariable x should not occur free in the conclusion of the rule.

The side conditions on the applications of $(\Rightarrow \forall)$ and $(\exists \Rightarrow)$ mean that these rules are *impure*. In general, therefore, the second part of Proposition 1 fails for GCL. This is the source of one of the two main differences between the propositional case and the present one: the two associated C.R.s $\vdash^t_{\mathrm{GCL}}$ and $\vdash^v_{\mathrm{GCL}}$ *are not identical*[13]. Thus $\varphi \vdash^v \forall x \varphi$ (the "generalization rule") while $\varphi \nvdash^t \forall x \varphi$. Similarly $\varphi \vdash^v \varphi(t/x)$ but $\varphi \nvdash^t \varphi(t/x)$. Below we shall see that the method of tableaux is based on $\vdash^t$, while resolution on $\vdash^v$. It follows that the two methods might provide different answers to the question whether a formula B follows from the set $\{A_1, \ldots, A_n\}$. Still, there is an important case in which this would not happen; when all the A_i's are *sentences*. (This includes, of course, the case $n = 0$.) This claim easily follows from the fact that with respect to sentences the rules of GCL *are* pure. Hence if $\Gamma \Rightarrow \Delta$ follows from $\Gamma_1 \Rightarrow D_i$ $(i = 1, \ldots, n)$ and $\Gamma' \cup \Delta'$ contains only sentences then $\Gamma, \Gamma' \Rightarrow \Delta, \Delta'$ follows from $\Gamma'_i, \Gamma' \Rightarrow \Delta_i, \Delta'$ $(i = 1, \ldots, n)$. Using this our claim follows as in the proof of Proposition 1.

The fact that GCL is not pure implies also that Proposition 2 (on which the method of resolution is based) is not true as it stands. Again, it *is* true, and with the same proof, when it is applied to a sequent of sentences.

[13] We shall henceforth omit the subscript GCL from $\vdash_{\mathrm{GCL}}$, $\vdash^t_{\mathrm{GCL}}$ and $\vdash^v_{\mathrm{GCL}}$.

Example 6: $\not\vdash p(x) \Rightarrow p(y)$ but $\big(p(y) \Rightarrow\big), \big(\Rightarrow p(x)\big) \vdash \Rightarrow$

Notes: Semantically, $\vdash^t$ corresponds to the "truth" C.R. of [Av]: $A_1, \ldots, A_n \vdash^t B$ if given any model M,B is true for any assignment in M which makes all the A_i's true. $\vdash^v$, on the other hand, corresponds to the "validity" C.R.: $A_1, \ldots, A_n \vdash^v B$ iff B is *valid* in every model in which $A_1, \ldots, A_n$ are all valid. See [Av] for further discussion.

The second crucial difference between the propositional case and the present one has to do with the two other new rules: $(\forall \Rightarrow)$ and $(\Rightarrow \exists)$. Unlike the rest of the rules, these rules are *not invertible*. Moreover, their conclusion has an infinite set of potential premises. This fact has far reaching consequences for the method of tableaux. As before this method is nothing but a systematic search for a cut-free proof in GCL of the sequent under consideration. *Unlike* the previous case, however, when it comes to reducing $T(\forall x\, A)$ to some $T\big(A(t/x)\big)$ (or $F(\exists x A)$ to $F\big(A(t/x)\big)$) we cannot be sure that we have chosen the right term t, or that only one reduction suffices. The standard solution in the past had been to keep signed formulae of these forms after a reduction has been applied to them, and to regularly return to them, trying all the possibilities in some order. Gentzen's classical cut-elimination theorem guarantees that if we start with a provable sequent then this systematic search for a proof will eventually terminate with a success (this theorem again states that a provable sequent has a cut-free proof).

The weakest point of the tableaux method just described is that it provides no clue to the order in which we should try to substitute the terms in the reductions of the problematic cases. This is a crucial point, since each useless choice might cause a chain of useless branchings. If there are several of them the search for a proof would never be finished in a feasible time. The "classical" tableaux method is not practical, therefore, for anything which is more than trivial.[14]

A possible idea for overcoming the problem caused by the noninvertible rules is the following: Suppose we reach a sequent $A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m$ of *sentences* to which no nonproblematic reduction can be applied. The A_i's are then of the form $\forall x\ A_i'$, the B_i of the form $\exists x\ B_i'$. As in the previous section, we can try at this point to show that the set $\big\{(\Rightarrow A_1), \ldots, (\Rightarrow A_n), (B_1 \Rightarrow), \ldots, (B_m \Rightarrow)\big\}$ is inconsistent. The advantage is, first of all,

[14] A much more practical version of tableaux is described in [Fi]. There tableaux with free variables are employed and their use (like in the case of resolution) is based on the ideas of substitutions for free variables and of unification. The connection method of Bibel [Bi] is strongly related to this form of tableaux. The exact relations between these methods and the ideas presented here will be described elsewhere.

that to each element of this set we can apply at least one nonproblematic reduction. In other words: the above set is inconsistent iff the set $\{(\Rightarrow A'_1), \ldots, (\Rightarrow A'_n), (B'_1 \Rightarrow), \ldots, (B'_m \Rightarrow)\}$ is inconsistent.

The classical resolution method, to which we now turn, directly treats only the case when each of the $(\Rightarrow A_i)$'s and the $(B_j \Rightarrow)$'s can be reduced to clause form using only the nonproblematic reductions. Such, e.g., is the case when each A_i is of the form $\forall x_1 \ldots \forall x_{\ell_i} A'_i$ and every B_j of the form $\exists x_1 \ldots \exists x_{\ell_j} B'_j$, where A'_i and B'_j are quantifiers-free. This indeed is the case which is usually discussed in the textbooks[15].

Example 7 ([Lo], pp. 32-37): Let

$$A = \neg\big(\exists x \neg p(x) \wedge \big[\exists x p(x) \vee \exists x \big(p(x) \wedge q(x)\big)\big] \wedge \neg\exists x p(x)\big)$$

Starting with the nonproblematic tableaux reductions we get the tree in Figure 1 below.

Notes: 1) The left-hand branch contains both $F\big(\exists x p(x)\big)$ and $T\big(\exists x p(x)\big)$, and so it is closed. Only the right-hand branch should therefore be dealt with.

2) In the tree above a and b are new *constants*. This is unavoidable, since the validity of the switching from showing provability to showing inconsistency is justified only if all the formulae involved are sentences. Now the corresponding Gentzen-type rules refer to *variables*. Still, it is easy to show that if x does not occur free in $\Gamma \cup \Delta$, then, e.g., $\vdash \Gamma \Rightarrow \Delta, A$ iff $\vdash \Gamma \Rightarrow \Delta, A(a/x)$ provided a is a new constant which does not occur in $\Gamma \Rightarrow \Delta, A$. In the usual tableaux method we use, indeed, new *constants* while doing $(\Rightarrow \forall)$ or $(\exists \Rightarrow)$ reductions. The resulting sequent is *not* equivalent to the original one, but it is provable iff the original one is. (The practical difference between new constants and variables in *the context of resolution* will become clear below.)

Example 7, continued: The right-hand branch of the tree which we got represents the sequent $p(b), q(b) \Rightarrow p(a), \exists x p(x)$. This is a sequent to which doing the switching is useful. What we get is $\big\{\big(\Rightarrow p(b)\big), \big(\Rightarrow q(b)\big), \big(p(a) \Rightarrow\big), \big(\exists x p(x) \Rightarrow\big)\big\}$. A $(\exists \Rightarrow)$ reduction on the last sequent produces the clause $p(x) \Rightarrow$, which subsumes $p(a) \Rightarrow$. The final set of clauses we get is, accordingly: $\big\{\big(\Rightarrow p(b)\big), \big(\Rightarrow q(b)\big), \big(p(x) \Rightarrow\big)\big\}$. This set is inconsistent, as can be shown by one step of resolution.

[15] Usually, in fact, only the case $m = 0$ is considered, and the A_i's are assumed to be disjunctions of literals. These extra assumptions are not needed here.

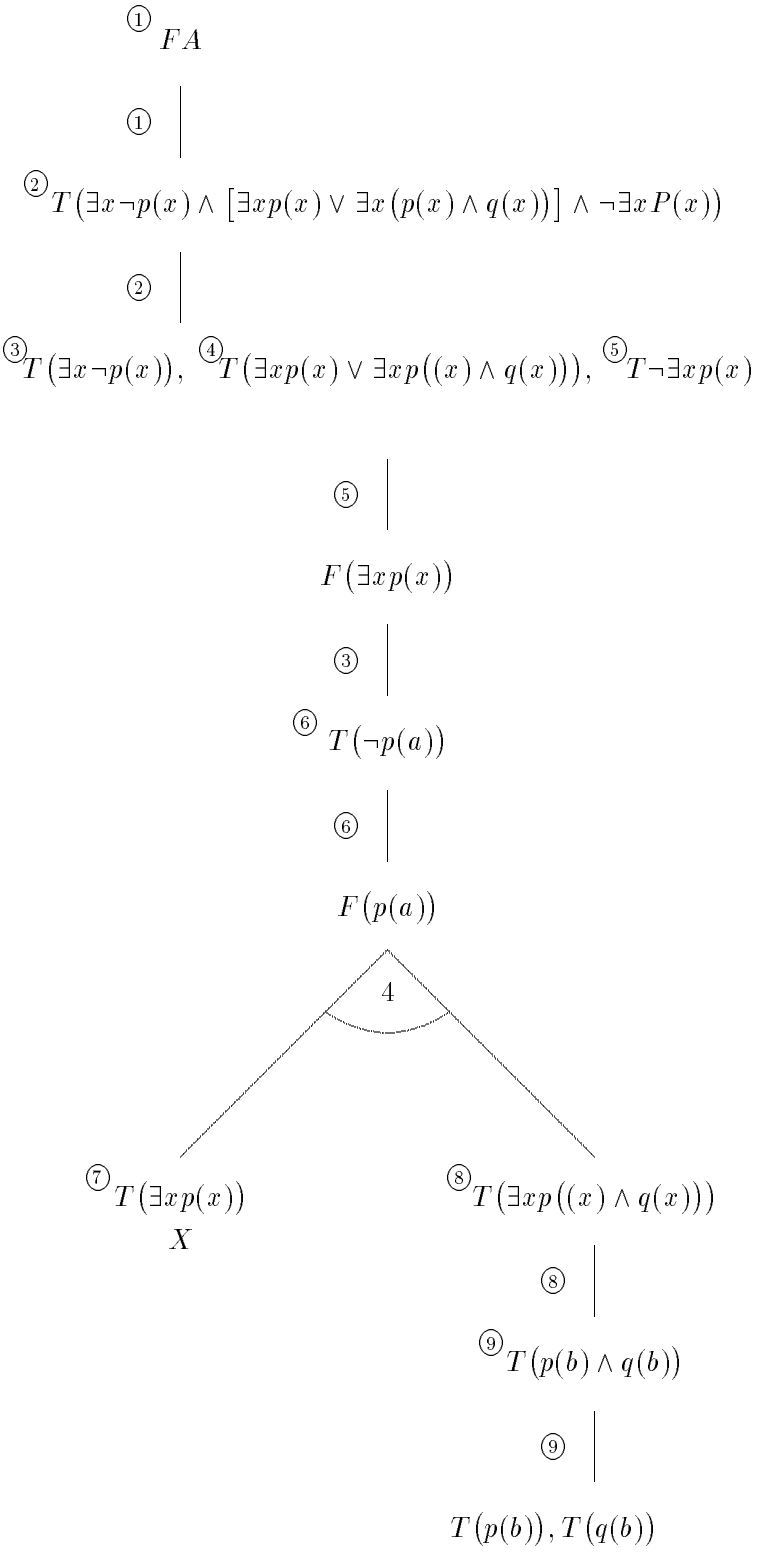


Figure 1

Notes: 1) In this stage we *cannot* substitute a constant for x while reducing $\exists xp(x)\Rightarrow$. The reason is that we are not trying to show provability of that sequent, but to use it as an

assumption in deriving the empty sequent!

2) The clause form which is obtained in [Lo] for the sentence in Example 7 is $\{(\Rightarrow p(a), p(b)), (\Rightarrow p(a), q(b)), (p(x) \Rightarrow)\}$, which is more complicated. We would have obtained the same set had we started by replacing $\exists x p(x) \vee \exists x (p(x) \wedge q(x))$ by the equivalent Prenex form: $\exists x \exists y (p(x) \vee (p(y) \wedge q(y)))$, or by the method of conversion to tableaux which is described at the end of this paper (see example 8).

Returning now to the resolution method, the question is now: how are we to prove that a given inconsistent set of clauses is indeed inconsistentΓ The answer might seem to be easy: just proceed as in the propositional case. This is wrong, however: despite the fact that all the quantifiers have been eliminated, *the problem has not been reduced to the propositional case*. This is illustrated by the following simple example: the set $\{(\Rightarrow p(x)), (p(a) \Rightarrow)\}$ is inconsistent, as the following deduction shows:

$$\dfrac{(\Rightarrow \forall)\dfrac{\Rightarrow p(x)}{\Rightarrow \forall x p(x)} \quad (\forall \Rightarrow)\dfrac{p(a) \Rightarrow}{\forall x p(x) \Rightarrow}}{(\text{Cut})\dfrac{}{\Rightarrow}}$$

There is, however, no way to derive $\Rightarrow$ from this inconsistent set using only propositionally valid rules!

The deep reason for this state of affairs has already been noted above: the C.R. which is associated with deductions in *GCL from assumptions* is $\vdash^v$, which is stronger than $\vdash^t$ when open formulae are involved. Now the main difference between $\vdash^t$ and $\vdash^v$ is the substitution rule, which is valid for the later but not for the former. In terms of sequents this rule allows inferring from a sequent any substitution-instance of it. This can also be achieved by using the quantifiers' rules and cuts (as in the example above), but if we want to avoid using quantifiers then we have no choice but to add substitution as an official rule. This would allow us to prove the following key result:

The strong cut-elimination theorem for GCL:[16] Let GCLS = GCL + substitution. Let S be a set of sequents and suppose $S \vdash_{\text{GCLS}} \Gamma \Rightarrow \Delta$. Then there is a proof in GCLS of $\Gamma \Rightarrow \Delta$ from S in which the substitution rule is applied only to sequents in S and all cuts are on instances of formulae which occur in sequents of S.

[16] Again, Girard seems to be the first to observe the validity of an equivalent theorem. See [Gi2, Theorem 2.7.1]. The only difference is that Girard does not consider substitution as an independent rule, but restricts himself to the case in which S is closed under it.

Proof (outline): First we use Gentzen's method to show that all cuts which are not on instances of formulae which occur in S are eliminable (the substitution rule is needed when the cut formula begins with a quantifier and we want to apply the induction hypothesis to immediate subformulae of it). Then we use induction on length of such proofs to show that all applications of the substitution rule can be done on sequents in S.

Gentzen's original cut-elimination theorem is again just the case $S = \emptyset$ in the last theorem. What *we* need here is another easy corollary, the proof of which we leave to the reader (see the proofs of Proposition 4 and its Corollary 1):

Corollary 2. *Let S be a set of clauses and s a clause such that $S \vdash s$. Let S' be a subset of S with the following properties:*

(1) *S' contains no tautology.*

(2) *Each element in S is subsumed by some element of S'.*

(3) *No element in S' is subsumed by another element of S'.*

Then there exists a clause s' which subsumed s so that s' can be inferred from S' using only substitutions and cuts.

Corollary 3. *If S is an inconsistent set of clauses then there is a finite set S' of instances of clauses in S from which $\Rightarrow$ can be derived using only cuts.*[17]

This last corollary (which we have proved in a purely syntactical way, and without any detour through ground clauses[18]) is the main syntactical fact on which the resolution method is based. It entails that in order to show the inconsistency of a set S of clauses we should just find the appropriate substitution instances of clauses in S, and then proceed as in the propositional case. Just as in the classical tableaux method the main problem is therefore: how to make the choice of the instances of S. Unlike that method, on the other hand, this time we *do* have a general guiding principle: substitutions should be made only if they open the door for applications of the cut rule. As is well known, this is most efficiently done not by doing all the substitutions at the beginning (as the last corollary may suggest) but by making them either as a part of a combination with cut ("binary resolution") or in order to unify two or more formulae in a clause ("factorization"). In both cases the most general substitution which accomplishes the task is chosen. Details can be found in any

[17] Herbrand theorem easily follows from this corollary!

[18] Although it is straightforward to prove it with the extra requirement that S' contains only ground clauses!

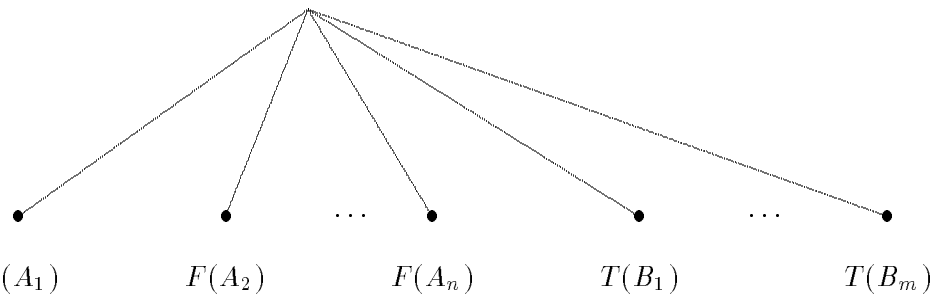
textbook on automated reasoning (the proofs given there at this point are purely syntactical and so we need not repeat them here. Many of them, as well as proofs of completeness of various refinements, can be simplified a lot using the last theorem and its corollaries).

To complete the picture we note that the usual method of proving a general sequent of sentences using resolution is to replace it first by a sequent to which this method is directly applicable and which is provable iff the original one is (although in general they are not equivalent). This is done by replacing first each sentence by an equivalent one in Prenex normal form, and then using Skolem functions to eliminate the unwanted quantifiers (the existential ones on the left, the universal ones on the right). The justification of these steps can also be done purely syntactically, using Gentzen midsequent theorem. Details can be found in [Ga1]. It is worth noting, however, that the passage to Prenex form often unnecessarily complicates matters. A much better procedure is based on the following observation: If S is a set of sequents then $S \cup \{\Gamma' \Rightarrow \Delta' \ , \ \exists\, x A\} \vdash \Gamma \Rightarrow \Delta$ iff $S \cup \big\{\Gamma' \Rightarrow \Delta' \ , \ A\big(f(y_1, \ldots, y_n)/x\big) \vdash \Gamma \Rightarrow \Delta$, where $y_1, \ldots, y_n$ are the free variables in $\Gamma \Rightarrow \Delta, \exists\, x A$ and f is a new function symbol, not occurring anywhere in $S, \Gamma, \Gamma', \Delta, \Delta'$ and A (a similar observation applies in the dual case). Since in resolution we are interested with provability *from* a set of sequents (and no, as in tableaux, in the provability of a sequent), this means that instead of the two problematic tableaux rules one can use the following two rules:

(1) Replace $F(\exists x A)$ by $F\big(A\big(f(y_1, \ldots, y_n)/x\big)\big)$ where f is a new function symbol that does not occur in the tree and $y_1, \ldots, y_n$ are the free variables on the branch in which the signed formula occurs.

(2) Replace $T(\forall x A)$ by $T\big(A\big(f(y_1, \ldots, y_n)/x\big)\big)$ under the same conditions.

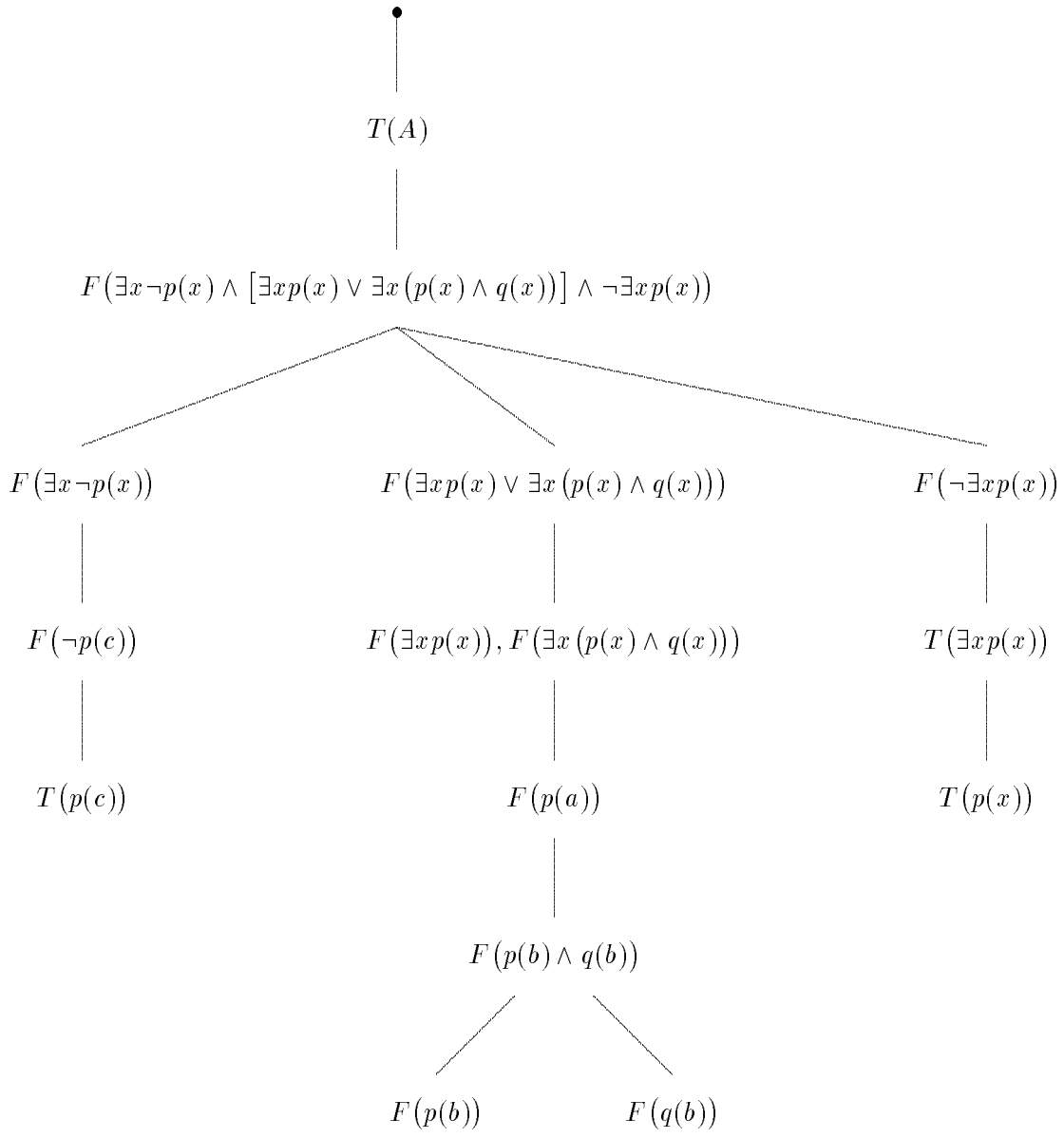
Now given a sequent of *sentences* $A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m$, find a set of clauses which is inconsistent iff this sequent is provable as follows: first construct the following tree.



(which represents the set of sequents: $\{\Rightarrow A_1, \ldots, \Rightarrow B_n, B_1 \Rightarrow, \ldots, B_m \Rightarrow\}$). Next, expand the tree using the new set of tableaux rules (i.e., all the previous invertible rules together

with the new ones). Stop when there are no more rules to apply. Every branch which is not closed contributes then a clause which consists of its atomic formulae. More precisely, if these formulae are $T(p_1), \ldots, T(p_{n'}), F(q_1), \ldots, F(q_{m'})$ then the corresponding clause is $p_1, \ldots, p_{n'} \Rightarrow q_1, \ldots, q_{m'}$.

Example 8. Returning to example 7, in order to prove A (i.e., $\Rightarrow A$) we construct the following tree



We get therefore the following set of clauses: $\big\{\big(p(c) \Rightarrow\big), \big(\Rightarrow p(a), p(b)\big), \big(\Rightarrow p(a), q(b)\big), \big(p(x) \Rightarrow\big)\big\}$. After deleting $p(c) \Rightarrow$ (which is subsumed by $p(x) \Rightarrow$) we get the clause form in [Lo] (compare notes after example 7).

Another important point which we would like to raise is that again the tableaux and the resolution methods can be combined to get better efficiency. Not only can tableaux help in obtaining an economical clause form in a faster way, but it might also split the work to be done by resolution into smaller jobs. In Example 7 we saw the extreme case, in which the work on one of these jobs was already finished in the tableaux stage (had we not closed the left-hand branch, we would have got the set $\{(\Rightarrow p(a)), (p(x) \Rightarrow)\}$ from it)!

References

[Av] Avron A., *Simple Consequence Relations*, **Information and Computation, 92** (1991), pp. 105-139.

[Bi] Bibel W., **Autometrd Theorem Proving**, Vieweg Verlag, Braunschweig, 1982.

[BM] Bell J.L. and Machover M., **A Course in Mathematical Logic**, North-Holland, Amsterdam, 1977.

[Bu] Bundy A., **The Computer Modelling of Mathematical Reasoning**, Academic Press, New York, 1983.

[CL] Chang C. & Lee R.C., **Symbolic Logic and Mechanical Theorem Proving**, Academic Press, New York, 1973.

[Fi] Fitting M.C., **First Order Logic and Automated Theorem Proving**, Springer-Verlag, 1989.

[Ga] Gallier J.H. **Logic and Computer Science – Foundations of Automatic Theory Proving**, Harper & Row, New York, 1986.

[Ge] Gentzen G. *Investigations into Logical Deduction*, in: **The Collected Work of Gerhard Gentzen**, ed. by M.E. Szabo, North Holland, Amsterdam, 1969.

[Gi1] Girard J.Y., Lafont Y. & Taylor P., **Proofs and Types**, Cambridge University Press, Cambridge, 1989.

[Gi2] Girard J.Y. **Proof Theory and Logical Complexity**, Bibliopolis, 1987.

[Ko] Kowalski R. **Logic for Problem Solving**, Elsevier, North Holland, New York, 1979.

[Lo] Loveland D.W. **Automated Theorem Proving: A Logical Basis**, Elsevier North-Holland, New York, 1989.

[OS] Oppacher F. & Suen E. *Controlling Deduction with Proof Condensation and Heuristics*, in: **Proceedings of CADE 8** (1986), pp. 364-393, Springer-Verlag.

[Ro] Robinson J.A. **Logic: Form and Function**, Elsevier North-Holland, New York, 1979.

[Sm] Smullyan R.M. **First-order Logic**, Springer-Verlag, Berlin, 1986.