Handbook of Tableau Methods

Introduction

Melvin Fitting

mlflc@cunyvm.cuny.edu

Contents

1 General Introduction

1.1 What Is A Tableau?

This chapter is intended to be a prolog setting the stage for the acts that follow—a bit of background, a bit of history, a bit of general commentary. And the thing to begin with is the introduction of the main character. What is a tableau?

It will make the introductions easier if we first deal with a minor nuisance. Suppose we know what a tableau is—what do we call several of

them: "tableaus" or "tableaux?" History and the dictionary are on the side of "tableaux." On the other hand, language evolves and tends to simplify; there is a clear drift toward "tableaus." In this chapter we will use "tableaus," with the non-judgemental understanding that either is acceptable. This brings our trivial aside to a cloeaux.

Now, what is a tableau? In its everyday meaning it is simply a picture or a scene, but of course we have something more technical in mind. A tableau method is a formal proof procedure, existing in many varieties and for several logics, but always with certain characteristics. First, it is a refutation procedure: to show a formula X is valid we begin with some syntactical expression intended to assert it is not. How this is done is a detail, and varies from system to system. Next, the expression asserting the invalidity of X is broken down syntactically, generally splitting things into several cases. This part of a tableau procedure—the *tableau expansion stage*—can be thought of as a generalization of disjunctive normal form expansion. Generally, but not always, it involves moves from formulas to subformulas. Finally there are rules for *closing* cases: impossibility conditions based on syntax. If each case closes, the tableau itself is said to be *closed*. A closed tableau beginning with an expression asserting that X is not valid is a tableau proof of X.

There is a second, more semantical, way of thinking about the tableau method, one that, perhaps unfortunately, has played a lesser role thus far: it is a search procedure for models meeting certain conditions. Each branch of a tableau can be considered to be a partial description of a model. Several fundamental theorems of model theory have proofs that can be extracted from results about the tableau method. Smullyan developed this approach in [84], and it was carried further by Bell and Machover in [5]. In automated theorem-proving, tableaus can be used, and sometimes are used, to generate counter-examples. The connection between the two roles for tableaus—as a proof procedure and as a model search procedure—is simple. If we use tableaus to search for a model in which X is false, and we produce a closed tableau, no such model exists, so X must be valid.

This is a bare outline of the tableau method. To make it concrete we need syntactical machinery for asserting invalidity, and syntactical machinery allowing a case analysis. We also need syntactical machinery for closing cases. All this is logic dependent. We will give examples of several kinds as the chapter progresses, but in order to have something specific before us now, we briefly present a tableau system for classical logic.

1.2 Classical Propositional Tableaus as an Example

In their current incarnation, tableau systems for classical logic are generally based on the presentation of Raymond Smullyan in [84]. We follow this in our sketch of a *signed tableau system* for classical propositional logic. Chapter **??** continues the discussion of propositional logic via tableaus.

(Throughout the rest of this handbook, *unsigned* tableaus are generally used for classical logic, but signs play a significant role when other logics are involved, and classical logic provides the simplest context in which to introduce them.)

First, we need syntactical machinery for asserting the invalidity of a formula, and for doing a case analysis. For this purpose two *signs* are introduced: T and F, where these are simply two new symbols, not part of the language of formulas. *Signed formulas* are expressions of the form $F\,X$ and $T\,X$, where X is a formula. The intuitive meaning of $F\,X$ is that X is *false* (in some model); similarly $T\,X$ intuitively asserts that X is *true*. Then $F\,X$ is the syntactical device for (informally) asserting the invalidity of X: a tableau proof of X begins with $F\,X$.

Next we need machinery—rules—for breaking signed formulas down and doing a case division. To keep things simple for the time being, let us assume that $\neg$ and $\supset$ are the only connectives. This will be extended as needed. The treatment of negation is straightforward: from $T\,\neg X$ we get $F\,X$ and from $F\,\neg X$ we get $T\,X$. These rules can be conveniently presented as follows.

Negation

$$\frac{T\,\neg X}{F\,X} \qquad\qquad \frac{F\,\neg X}{T\,X}$$

The rules for implication are somewhat more complex. From truth tables we know that if $X \supset Y$ is *false*, X must be *true* and Y must be *false*. Likewise, if $X \supset Y$ is *true*, either X is *false* or Y is *true*; this involves a split into two cases. Corresponding syntactic rules are as follows.

Implication

$$\frac{T\,X \supset Y}{F\,X \mid T\,Y} \qquad\qquad \frac{F\,X \supset Y}{\begin{array}{c} T\,X \\ F\,Y \end{array}}$$

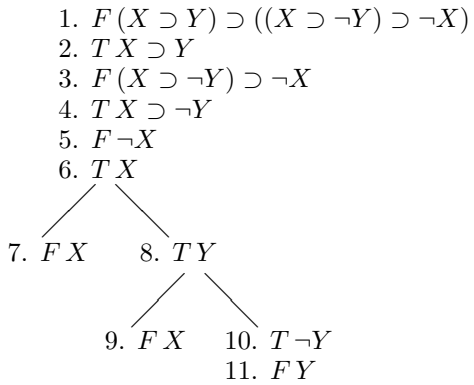
The standard way of displaying tableaus is as downward branching trees with signed formulas as node labels—indeed, the tableau method is often referred to as the *tree method*. Think of a tree as representing the disjunction of its branches, and a branch as representing the conjunction of the signed formulas on it. Since a node may be common to several branches, a formula labeling it, in effect, occurs as a constituent of several conjunctions, while being written only once. This amounts to a kind of structure sharing.

When using a tree display, a tableau expansion is thought of temporally, and one talks about the *stages* of constructing a tableau, meaning the stages of growing a tree. The rules given above are thought of as branch-lengthening rules. Thus, a branch containing $T\,\neg X$ can be lengthened by adding a new node to its end, with $F\,X$ as label. Likewise a branch

containing $F\,X \supset Y$ can be lengthened with two new nodes, labeled $T\,X$ and $F\,Y$ (take the node with $F\,Y$ as the child of the one labeled $T\,X$). A branch containing $T\,X \supset Y$ can be split—its leaf is given a new left and a new right child, with one labeled $F\,X$, the other $T\,Y$. This is how the schematic rules above are applied to trees.

An important point to note: the tableau rules are non-deterministic. They say what can be done, not what must be done. At each stage we choose a signed formula occurrence on a branch and apply a rule to it. Since the order of choice is arbitrary, there can be many tableaus for a single signed formula. Sometimes a prescribed order of rule application is imposed, but this is not generally considered to be basic to a tableau system.

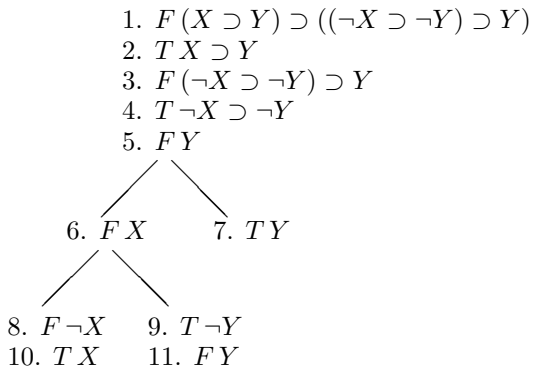
Here is the final stage of a tableau expansion beginning with (that is, *for*) the signed formula $F\,(X \supset Y) \supset ((X \supset \neg Y) \supset \neg X)$.



In this we have added numbers for reference purposes. Items 2 and 3 are from 1 by $F \supset$; 4 and 5 are from 3 by $F \supset$; 6 is from 5 by $F\neg$; 7 and 8 are from 2 by $T \supset$; 9 and 10 are from 4 by $T \supset$; 11 is from 10 by $T\neg$.

Finally, the conditions for closing off a case—declaring a branch closed—are simple. A branch is closed if it contains $T\,A$ and $F\,A$ for some formula A. If each branch is closed, the tableau is closed. A closed tableau for $F\,X$ is a tableau proof of X. The tableau displayed above is closed, so the formula $(X \supset Y) \supset ((X \supset \neg Y) \supset \neg X)$ has a tableau proof.

It may happen that no tableau proof is forthcoming, and we can think of the tableau construction as providing us with counterexamples. Consider the following attempt to prove $(X \supset Y) \supset ((\neg X \supset \neg Y) \supset Y)$.



Items 2 and 3 are from 1 by $F\supset$, as are 4 and 5 from 3. Items 6 and 7 are from 2 by $T\supset$, as are 8 and 9 from 4. Finally 10 is from 8 by $F\neg$, and 11 is from 9 by $T\neg$. The leftmost branch is closed because of 6 and 10. Likewise the rightmost branch is closed because of 5 and 7. But the middle branch is not closed. Notice that every non-atomic signed formula has had a rule applied to it on this branch—there is nothing left to do. (This is a special feature of classical propositional logic: it is sufficient to apply a rule to a formula on a branch only once. This does not apply generally to all logics.) In fact the branch yields a counterexample, as follows. Let v be a propositional valuation that maps X to *false* and Y to *false* in accordance with 6 and 11. Now, we work our way back up the branch. Since $v(Y) =$ *false*, $v(\neg Y) =$ *true*, item 9. Then $v(\neg X \supset \neg Y) =$ *true*, item 4. From this and the fact that $v(Y) =$ *false* we have $v((\neg X \supset \neg Y) \supset Y) =$ *false*, item 3. Also since $v(X) =$ *false*, $v(X \supset Y) =$ *true*, item 2. Finally, $v((X \supset Y) \supset ((\neg X \supset \neg Y) \supset Y)) =$ *false*, item 1. Notice, the valuation v gave to each formula on the unclosed branch the truth value the branch 'said' it had. But then, v is a counterexample to $(X \supset Y) \supset ((\neg X \supset \neg Y) \supset Y)$, so the formula is not valid.

From a different point of view, we can think of a classical tableau simply as a set of sets of signed formulas: a tableau is the set of its branches, and a branch is the set of signed formulas that occur on it. Semantically, we think of the outer set as the disjunction of its members, and these members, the inner sets, as conjunctions of the signed formulas they contain. Considered this way, a tableau is a generalization of disjunctive normal form (a generalization because formulas more complex than literals can occur). Now, the tableau construction process can be thought of as a variation on the process for converting a formula into disjunctive normal form.

In set terms, instead of lengthening branches, we expand sets. For example, suppose $C = \{Z_1, Z_2, \ldots, Z_n\}$ is a set of signed formulas, and one of the members is $T\,\neg X$. Then the set $\{Z_1, Z_2, \ldots, Z_n, F\,X\}$ is said to *follow*

from C. Similarly if the set contains $F\,\neg X$. If C contains $F\,X \supset Y$ then the set $\{Z_1, Z_2, \ldots, Z_n, T\,X, F\,Y\}$ follows from C. Finally, if C contains $T\,X \supset Y$ then the pair of sets $\{Z_1, Z_2, \ldots, Z_n, F\,X\}$, $\{Z_1, Z_2, \ldots, Z_n, T\,Y\}$ follows from C. Next, if D_1 and D_2 are tableaus (each represented as a set of sets of signed formulas) D_2 follows from D_1 if it is like D_1 but with one of its members C replaced with the set or sets that follow from it. Taken this way, a *tableau expansion* for a signed formula S is a sequence of tableaus, beginning with $\{\{S\}\}$, each tableau after the first following from its predecessor.

1.3 Abstract Data Types vs Implementations

Computer Science has made us familiar with the distinction between an abstract data type and an implementation of it. The notion of a *list*, with appropriate operations on it, is an abstract data type. It can be concretely implemented using an array, or using a linked structure, or in other ways as well. Which is better, which is worse? It depends on the intended application. But this abstract/concrete distinction is actually an old one. Before there were electronic computers, there were human computors, generally using mechanical devices like slates, paper, and slide rules. Before there were algorithms designed for computers, there were algorithms designed for humans, using the devices at hand. What we now call a good implementation of an abstract data type was once called *good notation*. Think, for instance, of the distinction between Roman numerals and Arabic notation. They both implement the same data type—non-negative integers—but one is more efficient for algorithmic purposes than the other.

Tableaus were described abstractly above: sets of sets of signed formulas. One of the things that helped make them popular was a good concrete implementation, a good notation: the tree display of a tableau. It is space-saving, since there is structure sharing, that is, formulas common to several branches are written only once. It is time-saving, since formulas are not copied over and over as the tree grows. Instead the state of a tree at any given moment represents a member of a tableau expansion sequence. Trees present a display that people find relatively easy to grasp, at least if it is not too big. And most importantly, there are by-hand algorithms that are wonderfully suited for use with tableaus as trees. In creating a tree tableau one has the same sense of calculation that one has when adding or multiplying using place-value notation.

Still, from an abstract point of view the tree display of signed formulas is not a tableau but an implementation of a tableau. It is not the only one possible; Manna and Waldinger use quite a different notation for their version of tableaus [53, 54]. The connection method can be thought of as based on tableaus, with trees replaced by more general graphs; see [13] for details. Indeed, we will see as this Chapter develops, that the tree

display itself underwent considerable evolution before reaching its current form. Also, for some logics, the straightforward tree version may not be best possible. For instance, with certain logics formulas get removed as well as added to branches, and with others they can come and go several times. For these, something more elaborate than just a growing tree is appropriate. Finally, what is good for hand calculation may not be at all useful for machine implementation. This point will be taken up in Chapter **??** and elsewhere. But perhaps the basic point of this digression is a simple one: trees implement tableaus. They do so quite well for many purposes—so well that they are often thought of as *being* tableaus. This is too restrictive, especially today when computers are being used to explore a wide variety of logics. Be willing to experiment.

1.4 What Good Is a Tableau System?

There are many kinds of proof procedures for many kinds of logics. What advantages does a tableau system have? Let us begin with what might be called the 'practical' ones.

The classical propositional system presented in section 1.2 can be used to 'calculate' in a way that Hilbert systems, say, can not. Each signed formula that is added comes directly from other signed formulas that are present. There is a choice of which signed formula to work with next, but there are always a finite number of choices—a bounded non-determinism, if you will. By contrast, since modus ponens is generally a Hilbert system rule, to prove Y we must find an X for which both X and $X \supset Y$ are provable. There are infinitely many possibilities for X—an unbounded non-determinism. Also note that the X we need may be considerably more complicated than the Y we are after, unlike with classical tableaus, where the formulas added are always simpler than the formulas they come from. Further, it can be shown that if the rules are applied in a classical tableau argument in a *fair* way, all proof attempts will succeed if any of them do. Thus any choices we make affect efficiency, not success. This means tableaus are well-suited for the discovery of proofs, either by people or by machines.

Once quantifiers are added, as in section 2.6, complications to this simple picture arise. There are infinitely many ways of applying some of the quantifier rules to a signed formula. If we systematically apply all rules in a fair way, it is still possible to show that a proof will be found, if a proof exists. But now, if a proof does not exist, the tableau expansion process will never terminate. Thus we get a semi-decision procedure—but after all, this is best possible. There is a problem for automation of tableaus that stems from the quantifier rules: systematically trying closed term after closed term to instantiate a universal quantifier is a terribly inefficient method. Fortunately there is a way around it, using so-called *free variable tableaus* and unification. This will be discussed in later chapters.

Suitability for proof discovery is something that applies equally well to resolution, and to several other techniques that have been worked out over the years. A peculiar advantage that tableaus have is that it seems to be easier to develop tableau systems for new logics than it is to develop other automatable proof procedures. This may be because tableaus tend to relate closely to the semantic ideas underlying a logic—or maybe the reasons lie elsewhere. What we are describing is an empirical observation, not a mathematical truth. Of course, once a tableau proof procedure has been created for a logic, it may be possible to use it to develop an automatable proof procedure of a different sort. Something like this is at the heart of Maslov's method [59]. But even so, tableaus provide a good starting point.

On a pedagogical level, tableaus can be used to provide quite appealing proofs of metatheoretical results about a logic. Take the issue of proving completeness as an example. One common way of showing a proof procedure is complete is to make use of maximal consistent sets. Such an approach is quite general, and can be applied to tableau proof procedures as readily as to Hilbert systems—see the proof of the Model Existence Theorem in [29]. But there is another approach to proving completeness that is much more intuitive. Start constructing a tableau expansion for $F\,X$. Apply rules fairly: systematically apply each applicable rule. If no closed tableau is produced, it can be shown that the resulting tableau contains enough information to construct a countermodel to X. We saw a propositional example of this kind in section 1.2. This is a nice feature indeed.

Other basic theorems about logic can also be given equally perspicuous tableau proofs. Smullyan, in [84], gives proofs of the compactness theorem, various interpolation theorems, and the Model Existence Theorem, all using tableaus in an essential way. Bell and Machover carried this even further [5]. Of course, we are speaking of classical logic, but similar arguments often carry over to other logics that have tableau proof procedures.

Finally, tableaus are well-suited for computer implementation. Their history in this respect is somewhat curious. We will have more to say on this topic as we discuss the history of tableaus, which we do throughout the rest of this chapter. (For another presentation of tableau history, see [2, 3].)

2 Classical History

Tableau history essentially begins with Gentzen. For classical logic, ignoring issues of machine implementation, it culminates with Smullyan. Here we discuss this portion of the development of our subject. In order to keep clutter down, we confine things to classical propositional logic (and occasionally intuitionistic propositional logic). This is sufficient to illustrate differences between systems and to follow their evolution. We will gener-

ally re-prove the same formula that we did in section 1.2, to allow easy comparison of the various systems.

2.1 Gentzen

In his short career Gentzen made several fundamental contributions to logic, see [89]. The one that concerns us here is his 1935 introduction of the *sequent calculus* in [32]. Before this, Hilbert-style, or axiomatic, proof procedures were the norm. In a sense, a Hilbert system characterizes a logic as a whole—it is difficult to separate out the role of individual connectives since several of them may appear in each axiom. What Gentzen contributed was a formulation of both classical and intuitionistic logics with a clear separation between structural rules (essentially characterizing deduction in the abstract) and specific rules for each connective and quantifier. Further, each connective and quantifier has exactly two kinds of rules; roughly, for its introduction and for its elimination. We say roughly because this terminology is more appropriate for the natural deduction systems that Gentzen also introduced in [32], but the essential idea is basically the same.

We'll sketch Gentzen's system, and make some comments on it. This should be familiar ground to most logicians. But we also note that, as with most things, the true beginnings of our subject are fuzzy. Gentzen's ideas grew out of earlier work of Paul Hertz [38], in 1929. Even the famous *cut* rule is a special case of Hertz's *syllogism* rule. See [89] for further discussion of this.

2.1.1 The Classical Sequent Calculus

First a new construct is introduced, the *sequent*. A sequent is an expression of the form:

$$X_1, \ldots, X_n \to Y_1, \ldots, Y_k$$

where $X_1, \ldots, X_n, Y_1, \ldots, Y_k$ are formulas. The arrow, $\to$, is a new symbol. It is understood that either (or both) of n and k may be 0. Informally, think of the sequent above as asserting: the disjunction of Y_1, $\ldots$, Y_k follows from the conjunction of X_1, $\ldots$, X_n.

The system has axioms, and rules of derivation. Axioms, or *initial sequents* as Gentzen called them, are sequents of the form $A \to A$, where A is a formula. Next, Gentzen has seven *structural rules*. We give six of them here; the seventh, *cut* will be discussed in a section of its own. In these and later rules, Γ, Δ, Θ, and Λ are sequences of formulas, possibly empty.

Thinning

$$\frac{\Gamma \to \Theta}{X, \Gamma \to \Theta} \qquad \frac{\Gamma \to \Theta}{\Gamma \to \Theta, X}$$

Contraction $\dfrac{X, X, \Gamma \to \Theta}{X, \Gamma \to \Theta}$ $\qquad$ $\dfrac{\Gamma \to \Theta, X, X}{\Gamma \to \Theta, X}$

Interchange $\dfrac{\Delta, Y, X, \Gamma \to \Theta}{\Delta, X, Y, \Gamma \to \Theta}$ $\qquad$ $\dfrac{\Gamma \to \Theta, Y, X, \Lambda}{\Gamma \to \Theta, X, Y, \Lambda}$

Next we give Gentzen's rules for the connectives $\neg$, $\wedge$, and $\supset$. The rules for $\vee$ are dual to those for $\wedge$ and are omitted.

Negation $\dfrac{X, \Gamma \to \Theta}{\Gamma \to \Theta, \neg X}$ $\qquad$ $\dfrac{\Gamma \to \Theta, X}{\neg X, \Gamma \to \Theta}$

Conjunction $\dfrac{\Gamma \to \Theta, X \quad \Gamma \to \Theta, Y}{\Gamma \to \Theta, X \wedge Y}$ $\qquad$ $\dfrac{X, \Gamma \to \Theta}{X \wedge Y, \Gamma \to \Theta}$ $\qquad$ $\dfrac{Y, \Gamma \to \Theta}{X \wedge Y, \Gamma \to \Theta}$

Implication $\dfrac{X, \Gamma \to \Theta, Y}{\Gamma \to \Theta, X \supset Y}$ $\qquad$ $\dfrac{\Gamma \to \Theta, X \quad Y, \Delta \to \Lambda}{X \supset Y, \Gamma, \Delta \to \Theta, \Lambda}$

Proofs are displayed in tree form, root at bottom. Each leaf must be labeled with an axiom; each non-leaf must be labeled with a sequent that follows from the labels of its children by one of the rules of derivation. A proof of the sequent $\to X$ is considered to be a proof of the formula X. Here is an example, a proof of $(X \supset Y) \supset ((X \supset \neg Y) \supset \neg X)$, with explanations added.

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{X \to X \quad \dfrac{Y \to Y}{\neg Y, Y \to}\,Negation}{X \supset \neg Y, X, Y \to}\,Implication
}{X \supset \neg Y, Y, X \to}\,Interchange
}{Y, X \supset \neg Y, X \to}\,Interchange
}{X \to X \quad Y, X, X \supset \neg Y \to}\,Interchange
}{X \supset Y, X, X, X \supset \neg Y \to}\,Implication
}{X, X \supset Y, X, X \supset \neg Y \to}\,Interchange
}{X, X, X \supset Y, X \supset \neg Y \to}\,Interchange
}{X, X \supset Y, X \supset \neg Y \to}\,Contraction
}{X, X \supset \neg Y, X \supset Y \to}\,Interchange
}{X \supset \neg Y, X \supset Y \to \neg X}\,Negation
}{X \supset Y \to (X \supset \neg Y) \supset \neg X}\,Implication
}{\to (X \supset Y) \supset ((X \supset \neg Y) \supset \neg X)}\,Implication
$$

Proofs in the sequent calculus are displayed beginning with axioms, ending with the sequent to be proved. Although it was probably of minor importance to Gentzen, others soon realized that by turning the rules

upside-down, a proof discovery system resulted. Given any sequent, there are a limited number of sequents from which it could be derived. Try deriving them—this reduces the problem to a simpler one, since premises of rules involve subformulas of their conclusions. Thus the discovery of a Gentzen-style proof is a much more mechanical thing than it is with Hilbert systems. It is not hard to extract a formal algorithm for decidability of both classical and intuitionistic propositional logics.

When using the rules backward, it is useful to think 'negatively' instead of 'positively.' That is, suppose we want to show a sequent, say $X_1, X_2 \to Y_1, Y_2, A \wedge B$ is provable. Well, suppose it is not. By one of the Conjunction rules, either $X_1, X_2 \to Y_1, Y_2, A$ or $X_1, X_2 \to Y_1, Y_2, B$ is not provable. Continue working backward in this way, until a contradiction (an axiom is not provable) is reached.

This backward way of thinking makes it easy to see in what way the sequent calculus relates to later tableau systems. Recall, we think of a sequent as informally saying the disjunction of the right side follows from the conjunction of the left. Then, if we did not have $X_1, X_2 \to Y_1, Y_2, A \wedge B$, everything on the left 'holds' in some model, and nothing on the right does. That is, denying the sequent informally amounts to assuming the satisfiability of $\{T\,X_1, T\,X_2, F\,Y_1, F\,Y_2, F\,A \wedge B\}$. From this, using a Conjunction rule backwards, we have the satisfiability of one of $\{T\,X_1, T\,X_2, F\,Y_1, F\,Y_2, F\,A\}$ or $\{T\,X_1, T\,X_2, F\,Y_1, F\,Y_2, F\,B\}$. At this point we can represent things using a set of sets of formulas,

$$\Big\{\{T\,X_1, T\,X_2, F\,Y_1, F\,Y_2, F\,A\}, \{T\,X_1, T\,X_2, F\,Y_1, F\,Y_2, F\,B\}\Big\}$$

where the outer set is thought of disjunctively and the inner sets conjunctively. This leads back to our set version of tableaus, in section 1.1.

2.1.2 Cut and the Structural Rules

We gave six structural rules above. The combined effect of Contraction and Interchange is that we can think of the *sequences* of formulas on either side of a sequent arrow as *sets* of formulas. This, combined with Thinning, allows us to use an apparently more general axiom schema: $\Gamma \to \Delta$, where Γ and Δ have a formula in common. These days, it is not uncommon to find sequent calculi formulated with sets instead of sequences, or with Gentzen's axiom scheme modified, or some combination of these. Kleene, in [47], gives three different versions, G1, G2, and G3, differing primarily on structural details.

The rules for conjunction and for implication are not like each other. There are three conjunction rules but only two implication rules. This can be remedied, if desired. It can be shown that an equivalent system results if the two rules for introducing a conjunction on the left of an arrow are

replaced by the following single rule:

$$\frac{X, Y, \Gamma \to \Theta}{X \wedge Y, \Gamma \to \Theta}$$

Proof of the equivalence of the two formulations uses the structural rules in an essential way.

Girard realized that the structural rules are not minor, but central. They are essential for proving the equivalence of the two versions of the conjunction rules, as we just saw. By dropping Thinning and Contraction (and making other changes as well) Girard devised *Linear Logic* [33, 93]. Other so-called substructural logics, such as *Relevance Logic* [22], arise in similar ways. Note that, without the structural rules, there are two different ways of introducing conjunction (and disjunction). Since these are no longer equivalent, substructural logics, in fact, have two notions of conjunction and disjunction.

We have left the Cut rule for last, since its role is both important and unique. The Cut rule is a kind of transitivity condition; in the following, the formula X is cut away.

Cut

$$\frac{\Gamma \to \Theta, X \quad X, \Delta \to \Lambda}{\Gamma, \Delta \to \Theta, \Lambda}$$

All the other Gentzen rules have a special, remarkable property: the subformula property. Each formula appearing above the line of a rule is a subformula of some formula appearing below the line. It is this on which decidability results in the propositional case rest. It is this that makes the construction of proofs seem mechanical. The Cut rule violates the subformula property: X appears above the line, and disappears below. If Cut is allowed, the system is in many ways less appealing.

Why, then, have a Cut rule at all? In showing Gentzen's formulation is at least as strong as a Hilbert axiom system, we must do two things: we must show each Hilbert axiom is Gentzen provable, and we must show each Hilbert rule of inference preserves Gentzen provability. All this is straightforward, except for modus ponens. However, if the Cut rule is available, it is easy to show that modus ponens preserves Gentzen provability. Consequently, it is enough to show Gentzen's systems with and without the Cut rule are equivalent—a result usually referred to as "Cut eliminability."

One can show Cut is eliminable by showing that Gentzen systems with and without cut are both sound and complete. Gentzen did not proceed this way, essentially because completeness proofs for first-order logic are non-constructive, and constructivity was a key part of Gentzen's motivation. Instead, Gentzen gave what today we would describe as an algorithm for removing Cuts from a proof, together with a termination argument. Cut elimination has become the centerpoint of proof theory.

Allowing Cuts in an automated proof system is, in a sense, allowing the use of Lemmas. Cut elimination says they are not necessary. On the other hand, an analysis of Gentzen's proof of Cut elimination shows that, when removing the use of Lemmas, proof length can grow exponentially. Clearly this is an important issue.

2.1.3 Intuitionistic Logic

In [32] Gentzen showed something of the versatility inherent in the sequent calculus by giving an intuitively plausible system for intuitionistic logic, as well as one for classical logic. (Chapter **??** contains a full treatment of theorem-proving in intuitionistic systems.) Intuitionistic logic is meant to be constructive—in particular, a proof of $X \vee Y$ should be either a proof of X or a proof of Y. This is different than in classical logic where one can have a proof of $X \vee \neg X$ without having either a proof or a disproof of X. Now, recall that the right-hand side of a sequent is interpreted as a disjunction. Then, intuitionistically, we should be able to say *which* member of the disjunction is a consequence of the left-hand side. This, of course, is all quite informal, but it led to Gentzen's dramatic modification of the sequent calculus rules: allow at most one formula to appear on the right of an arrow. Gentzen showed this gave a system that was equivalent to an axiomatic formulation of intuitionistic logic, by making use of his Cut elimination theorem. Nothing else was possible, since there was no known semantics for intuitionistic logic at that time.

2.1.4 Gentzen's Immediate Heirs

Gentzen's introduction of the sequent calculus was enormously influential, and similar formulations were soon introduced (after the void of World War II) for other kinds of logics. We briefly sketch some of the early developments.

Beginning in 1957, Ohnishi and Matsumoto gave calculi for several modal logics [62, 63, 64, 61, 55]. We describe their system for $S4$ as a representative example. (See Chapter **??** for an extended discussion of the role of tableaus in modal theorem-proving.) We take $\Box$ as primitive, and for a sequence Γ of formulas, we write $\Box\Gamma$ for the result of prefixing each formula in Γ with $\Box$. Now we add to Gentzen's rules the following.

S4
$$\frac{X, \Gamma \to \Theta}{\Box X, \Gamma \to \Theta} \qquad \frac{\Gamma \to X}{\Box\Gamma \to \Box X}$$

Note that the second of the rules for introducing $\Box$ allows only a single formula X on the right, analogous to Gentzen's rules for intuitionistic logic. This should come as no surprise, since there are close connections between $S4$ and intuitionistic logic.

Since semantical methods were not much used in modal logic at the time, the equivalence of these systems with Hilbert style ones was via a

translation procedure, making essential use of Cut elimination. The sequent formulations were, in turn, used to obtain decision procedures for the logics.

At about the same time, Kanger also gave sequent style formulations for some modal logics [46]. His system for $S5$ is of special interest because it introduced a new piece of machinery: propositional formulas were indexed with positive integers. These integers can be thought of as corresponding to the possible worlds of Kripke models, though this is not how Kanger thought of them. If X is indexed with n, it is written as X^n. The Gentzen rules are modified so that in a conjunction rule, for instance, a conjunction receives the same index as its conjuncts (which must be the same). Then the following two rules are added.

S5

$$\frac{X^m, \Box X^n, \Gamma \to \Theta}{\Box X^n, \Gamma \to \Theta}$$

Where $m \neq n$.

$$\frac{\Gamma \to \Theta, X^n}{\Gamma \to \Theta, \Box X^m}$$

Where no formula with index n occurs within the scope of $\Box$ in Γ or Θ.

This is an early forerunner of the now widespread practice of adding extra machinery to sequent and tableau systems. We will see more examples later on. The Ohnishi and Matsumoto systems, and the Kanger systems, can be found in some detail in [23].

In 1967 Rousseau [75] treated many-valued logics using Gentzen methods. (Chapter **??** discusses current tableau theorem-provers for many-valued logics.) The basic ideas are relatively simple. A classical sequent, $X_1, \ldots, X_n \to Y_1, \ldots, Y_k$ is considered satisfiable if, under some valuation, either some X_i is *false* or some Y_i is *true*. Consider the left-hand side as the falses's and the right-hand side as the true's. Then a sequent is satisfiable if one of the false's is *false* or one of the true's is *true*. Rousseau extended this to an m-valued logic—say the truth values are $0, 1, \ldots, m-1$. Now a sequent is an expression:

$$\Gamma_0 \mid \Gamma_1 \mid \ldots \mid \Gamma_{m-1}$$

where each Γ_i is a sequence of formulas. The sequent is considered satisfiable if some member of Γ_i has truth value i. This reading of a sequent suggests appropriate rules. For instance, Gentzen's axiom schema, $A \to A$, turns into the schema $A \mid A \mid \ldots \mid A$. Rousseau gave a method of producing connective rules, and showed soundness and completeness. For many-valued logics, the more interesting issue is that of quantification, which we do not touch on here, though it was discussed by Rousseau.

2.2 Beth

Gentzen's motivation was proof-theoretic. He was more-or-less explicitly analyzing proofs, and his work has become the foundation of modern proof theory. There is no attempt at a completeness or soundness argument in his paper—only constructive proofs of equivalence with other formalisms. Beth, on the other hand, was motivated by semantic concerns [9, 10]. In 1955 he introduced the terminology, 'semantic tableau,' and thought of one as a systematic attempt to find a counter-example. To quote from [9]:

> "If such a counter-example is found, then we have a negative answer to our problem. And if it turns out that no suitable counter-example can be found, then we have an affirmative answer. In this case, however, we must be sure that no suitable counter-example whatsoever is available; therefore, we ought not to look for a counter-example in a haphazard manner, but we must rather try to construct one in a systematic way. Now there is indeed a systematic method for constructing a counter-example, if available; it consists in drawing up a *semantic tableau*."

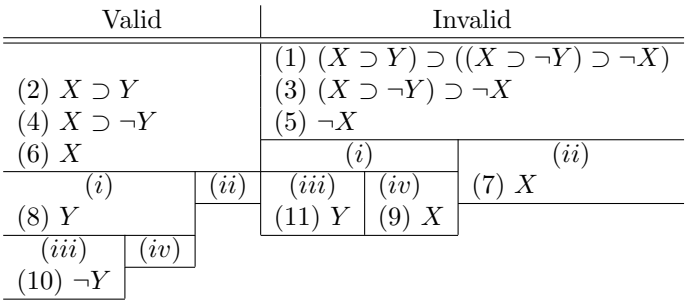
Beth arranged his counter-example search in the form of a table with two columns, one labeled 'Valid,' the other, 'Invalid.' Perhaps 'True' and 'False' (in some model) would be more accurate. To determine whether Y is a consequence of $X_1, \ldots, X_n$, begin by placing $X_1, \ldots, X_n$ in the *Valid* column, and Y in the *Invalid* one. This corresponds to beginning with the conjecture that Y is *not* a consequence of $X_1, \ldots, X_n$, rather like using the sequent calculus backward. Next, systematically break down the formulas in each column. For example, if $A \wedge B$ appears in the *Valid* column, add both A and B to it. Similarly if $A \vee B$ appears in the *Invalid* column, add both A and B to it. If $\neg A$ occurs in a column, add A to the other one. Things get a little awkward with disjunctive cases however. If $A \vee B$ occurs in the *Valid* column we should be able to add one of A or B—the problem is which. Beth's solution was to split the *Valid* column in two, thus displaying both possibilities. Of course a corresponding split has to be made in the *Invalid* column. Since further splitting might occur, it is necessary to label the various columns, to keep the *Valid* and the *Invalid* columns that belong together properly associated. In practice this can be hard to follow if there are many cases, but the principle is certainly clear.

If a semantic tableau is constructed as outlined above, there are basically only two possible outcomes. It may happen that a formula appears in both the *Valid* and the *Invalid* columns, which indicates an impossibility. If the tableau system really does embody a thorough, systematic analysis, such an impossibility tells us there are no models in which $X_1, \ldots, X_n$ are true but Y is not; that is, there are no counter-examples, and so Y must be a consequence of $X_1, \ldots, X_n$. The other possibility is that no such

contradiction ever appears. In this case, Beth observed, the tableau itself supplies all the necessary information to produce a counter-example, and so Y is not a consequence of $X_1, \ldots, X_n$.

The description above is correct in the propositional case, and Beth's method supplies a decision procedure. In the first-order case things are more complex since if there are no counterexamples, a tableau construction may never terminate. If this happens we still generate the information to construct a model, but only in the limit. A rigorous treatment of this point requires some care, and we gloss over it.

Here is an example of a Beth semantic tableau—a proof once again of the familiar tautology $(X \supset Y) \supset ((X \supset \neg Y) \supset \neg X)$.



In the tableau above, we begin with 1 in the *Invalid* column, which gives 2 in the *Valid*, and 3 in the *Invalid* ones. From 3 we conclude 4 is *Valid* and 5 is *Invalid*. Likewise 5 produces 6. Now things become more complicated. If $X \supset Y$ is true in a model, either X is not true there, or Y is. Then formula 2 causes a split into two cases, labeled i and ii, one placing X in the *Invalid* column, formula 7, the other placing Y in the *Valid* column, formula 8. Likewise formula 4 causes another split, creating subcases iii and iv, with formula 9 on the *Invalid* side and formula 10 on the *Valid* one. Formula 10 in turn yields formula 11. Now we see we have arrived at a contradictory tableau. Case ii is impossible because of formulas 6 and 7. Case iii is impossible because of 8 and 11, and case iv is impossible because of 6 and 9; this means case i is impossible. Since each subcase is impossible, or *closed* as Beth called it, the tableau itself is closed; there are no counter-examples; $(X \supset \neg Y) \supset \neg X$ *is* a consequence of $X \supset Y$.

2.2.1 Tableaus and Consequences

Beth recognized that tableaus make it possible to give proofs of results about classical logic that are intuitively satisfying. In his book [12] Beth used tableaus to show a *subformula principle* saying that if a formula has a proof, it has one in which only subformulas of it occur. He derived a version

of Herbrand's Theorem, and Gentzen's Extended Hauptsatz. He explicitly discussed the relationship between tableaus and the sequent calculus. He even gave a tableau-based proof of Gentzen's Cut Elimination Theorem (a proof with a fundamental flaw, as it happens). And he considered the relationship between tableaus and natural deduction.

Among Beth's fundamental consequences is his famous Definability Theorem of 1953 [8, 7], relating implicit and explicit definability in first-order logic. Statements of it can be found in many places, in particular in [19, 84, 29]; details are beside the point here. It is usual to derive Beth's Theorem from the Craig Interpolation Theorem and Beth takes this route in his book (this was not his original proof however). In turn, Beth uses tableaus to prove the Craig Lemma (not Craig's original proof either). The reason this is mentioned here is to illustrate Beth's almost physical sense of the tableau mechanism. The following is a quote from the beginning of his proof of Craig's Lemma.

> "Let us suppose that the semantic tableau for a certain sequent (f) is closed. We consider the tableau as a system of communicating vessels. The left and right columns are considered as tubes which are connected at the bottom of the apparatus. The formulas U create a downward pressure in the left tubes and likewise the formulas V create a downward pressure in the right tubes; these various pressures result in a state of equilibrium.
> "This picture suggests the construction, for each of the formulas U and V, of a formula U^0 or V^0 which sums up the total contribution of U or V to the balance of pressures."

The proof continues somewhat more technically. These days tableaus are often used to prove versions of Craig's Lemma, but never in quite as picturesque a fashion.

2.2.2 Intuitionistic Logic Again

Gentzen's approach to intuitionistic logic could not be based on semantics, since none was available at that time. An algebraic/topological semantics was developed soon after [90, 69, 70], but this was not particularly satisfactory as an explication of intuitionistic ideas to a classical mathematician. In 1956 Beth provided a much more intuitively appealing semantics [10, 12], known today as *Beth models*. (These have been largely superseded by an alternative semantics due to Kripke.) What concerns us here is that, at the same time, Beth introduced a tableau system for intuitionistic logic. He presented this in the form of a sequent calculus and, unlike in Gentzen's system, several formulas could appear on the right of an arrow, at least sometimes. He explicitly noted it could be used as written, as a sequent calculus, or upsidedown, as a tableau system.

Beth's intuitionistic tableau calculus introduced a new element: there were two kinds of branching, conjunctive and disjunctive. When a branch splits disjunctively, closure of one of the new branches is enough to close the original one. With conjunctive branching, on the other hand, both branches must close for the original branch to be closed. Conjunctive branching is the only kind that occurs in classical tableaus. Intuitionistic propositional logic has a higher degree of computational complexity than classical propositional logic, and this can be traced to the possibility of disjunctive branching.

Beth gave a constructive proof of the equivalence of his system with that of Gentzen. More interestingly, he proved the soundness and completeness of his intuitionistic tableau system with respect to his semantics. Since he was concerned with intuitionism as a philosophy of mathematics, he explicitly considered which points of his completeness proof would be problematic for an intuitionist. He found this centered on the *Tree Theorem*, essentially König's Lemma, though he does not call it that.

We do not give Beth's intuitionistic tableau system here. It is easier to present the basic ideas using signed formulas, and we do so later.

2.3 Hintikka

It has been noted that scientific advances come when the times are ready, and often occur to several people simultaneously. The work of Beth and Hintikka is such an event, with Hintikka's primary paper, [41], appearing in 1955, the same year as Beth's. (See also [40].) Like Beth, and unlike Gentzen, Hintikka was motivated by semantic concerns: the idea behind a proof of X is that it is a systematic attempt to construct a model in which $\neg X$ is true; if the attempt fails, X has been established as valid. Or, as Hintikka puts it:

> "... we interpreted all proofs of logical truth in a seemingly negative way, *viz., as proofs of impossibility of counter-examples*."

As expected, a proof attempt proceeds by breaking formulas down into constituent parts.

> "... the typical situation is one in which we are confronted by a complex formula (or sentence) the truth or falsity of which we are trying to establish by inquiring into its components. Here the rules of truth operate from the complex to the simple: they serve to tell us what, under the supposition that a given complex formula or sentence is true, can be said about the truth-values of its components."

2.3.1 Model Sets

The essentially new element in Hintikka's treatment was the *model set*. It makes possible a considerable simplification in the proof of completeness

for tableaus by abstracting properties of satisfiability of formula sets out of details of the tableau construction process. And it suggests the possibility of extensions to modal logics, which Hintikka himself later developed. As usual, we illustrate the ideas via classical propositional logic. First, though, we mention a pecularity of Hintikka's treatment: he assumed all negations occur at the atomic level. Any non-atomic occurrence of a negation symbol was taken to be eliminable, via the usual negation normal form rules. Likewise, implication was not taken as primitive.

Now, suppose we have a classical propositional model, $\mathcal{M}$. Associate with it the set μ of propositional formulas that are true in it. We can say things like: $X \wedge Y \in \mu$ if and only if $X \in \mu$ and $Y \in \mu$. This if-and-only-if assertion can be divided into two implications; Hintikka's insight was to see that one of these implications determines the other. To be more precise, consider the following two sets of conditions.

(C.0)(a) If $A \in \mu$, then not $\neg A \in \mu$, where A is atomic.
(C.1)(a) If $X \wedge Y \in \mu$, then $X \in \mu$ and $Y \in \mu$.
(C.2)(a) If $X \vee Y \in \mu$, then $X \in \mu$ or $Y \in \mu$.
(C.0)(b) If not $\neg A \in \mu$, then $A \in \mu$, where A is atomic.
(C.1)(b) If $X \in \mu$ and $Y \in \mu$, then $X \wedge Y \in \mu$.
(C.2)(b) If $X \in \mu$ or $Y \in \mu$, then $X \vee Y \in \mu$.

If μ is a set meeting all the conditions above, both (a) and (b), there is clearly a model $\mathcal{M}$ whose true formulas are exactly those of μ. Now, Hintikka proved that if μ is known to meet only the (a) conditions, this is still enough—any such set can be extended to one meeting the (b) conditions as well, and hence corresponds to some model. This led Hintikka to call sets meeting the (a) conditions *model sets*. Today they are sometimes called *downward saturated sets* or even *Hintikka sets*. Using this terminology, we have the following.

Hintikka's Lemma Every downward saturated set is satisfiable.

In section 1.2 we saw that an unclosed tableau branch can be used to generate a model provided that on it all possible rules have been applied. In a natural sense, the set of signed formulas on such a branch is a version of Hintikka's notion of model set, and our creation of a model amounts to a special case of Hintikka's proof of his Lemma. Of course as we stated it, Hintikka's Lemma is for propositional logic. He actually proved a version for first-order logic; it extends to admit equality, and to various non-classical logics as well.

2.3.2 The Hintikka Approach

Suppose we wish to establish the validity of some formula, say that of our old friend $(X \supset Y) \supset ((X \supset \neg Y) \supset \neg X)$. Hintikka's idea is simple: show

$\neg[(X \supset Y) \supset ((X \supset \neg Y) \supset \neg X)]$ is not satisfiable, and do this by showing it belongs to no model (downward saturated) set. Do this by supposing otherwise, and deriving a contradiction. Recall, Hintikka assumed formulas are in negation normal form, so if we begin with a downward saturated set μ with $\neg[(X \supset Y) \supset ((X \supset \neg Y) \supset \neg X)] \in \mu$, we are really assuming the following.

1. $(\neg X \vee Y) \wedge ((\neg X \vee \neg Y) \wedge X) \in \mu$

From 1, by (C.1)(a) we have:

2. $\neg X \vee Y \in \mu$
3. $(\neg X \vee \neg Y) \wedge X \in \mu$

Then from 3 we get:

4. $\neg X \vee \neg Y \in \mu$
5. $X \in \mu$

Now, by 2, either $\neg X \in \mu$ or $Y \in \mu$. If the first of these held, we would have $X \notin \mu$, by (C.0)(a), contradicting 5. Consequently we have

6. $Y \in \mu$

By 4, either $\neg X \in \mu$ or $\neg Y \in \mu$. But, using (C.0)(a), the first of these possibilities contradicts 5, and the second contradicts 6. Thus we have arrived at a contradiction—no such μ can exist.

2.4 Lis

Beth and Hintikka each had all the pertinent parts of tableaus as we know them, but their systems were not 'user-friendly.' Beth proposed a graphical representation for tableaus, see section 2.2, but his two-column tables, with two-column subtables (and subsubtables, and so on) are not handy in practice. Hintikka, in effect, used a tree structure but with sets of formulas at nodes, requiring much recopying. Notational simplification was the essential next step in the development of tableaus and, just as with the preceding stage, it was taken independently by two people: Zbigniew Lis and Raymond Smullyan. Lis published his paper [52] in 1960, but in Polish (with Russian and English summaries), in *Studia Logica*. At that time there was a great gulf fixed between the East and the West in Europe, and Lis's ideas did not become generally known. They were subsequently rediscovered and extended by Smullyan, culminating in his 1968 book [84]. The work of Lis himself only came to general attention in the last few years.

Lis, following Beth, divided formulas into two categories, Beth's 'valid' and 'invalid.' But Lis did so not by separating them into columns, but keeping them together and distinguishing them by 'signs.' Lis used arithmetical notation, + or −, for Beth's two categories. (He also used a formal

numeration system to record which formulas followed from which—we ignore this aspect of his system.) He then stated the following rules (we only give the propositional ones).

(i) If $\pm\neg X$, then $\mp X$.

(ii) Conjunctive Rules

a) If $+(X \wedge Y)$, then $+X$, $+Y$.

b) If $-(X \vee Y)$, then $-X$, $-Y$.

c) If $-(X \supset Y)$, then $+X$, $-Y$.

(iii) Disjunctive Rules

a) If $-(X \wedge Y)$ then $-X \mid -Y$

b) If $+(X \vee Y)$ then $+X \mid +Y$

c) If $+(X \supset Y)$ then $-X \mid +Y$

These rules are intended to be used in the same way the 'T' and 'F' signed rules were in section 1.2, though his display of trees was rather like Beth's tables. Lis also gave rules for quantifiers, for equality, and even for definite descriptions.

In addition to the system of semantic tableaus using signs, Lis also presented what he called a *natural deduction* system—what we would call an *unsigned* tableau system. For this, drop all occurrences of the + sign, and replace occurrences of the − sign with occurrences of negation, ¬. (This makes half of rule (i) redundant.)

2.5 Smullyan

It is through Smullyan's 1968 book *First-Order Logic* [84] that tableaus became widely known. They also appeared in the 1967 textbook [45], which was directed at beginning logic students. Smullyan's book was preceded by [81, 82, 83] in which the still unknown contributions of Lis were rediscovered, deepened, and extended. Smullyan called his version 'analytic tableau,' meaning by this that the subformula principle is a central feature. Smullyan used tableaus as the basis of a general treatment of classical logic, including an analysis of the variety of completeness proofs. Drawing together ideas from several sources, and adding new ones of his own, quite an elegant treatment resulted.

2.5.1 Unifying Notation

Like Lis, Smullyan introduced both signed and unsigned tableau systems. Where Lis used + and − as signs, Smullyan used T and F, but the essential idea is the same. But Smullyan, instead of treating these as parallel, similar systems, abstracted their common features. He noted that

signed formulas act either *conjunctively* or *disjunctively* (in the propositional case—quantification adds two more categories). He grouped the conjunctive cases together as *type A* formulas, and the disjunctive ones as *type B*, using α for a generic type A formula and β as generic type B. For each of these, two *components* were defined: α_1 and α_2 for type A; β_1 and β_2 for type B. Smullyan's tables for both the signed and the unsigned versions are as follows.

α	α_1	α_2
$T\,(X \wedge Y)$	$T\,X$	$T\,Y$
$F\,(X \vee Y)$	$F\,X$	$F\,Y$
$F\,(X \supset Y)$	$T\,X$	$F\,Y$
$T\,\neg X$	$F\,X$	$F\,X$
$F\,\neg X$	$T\,X$	$T\,X$
$(X \wedge Y)$	X	Y
$\neg(X \vee Y)$	$\neg X$	$\neg Y$
$\neg(X \supset Y)$	X	$\neg Y$
$\neg\neg X$	X	X

β	β_1	β_2
$F\,(X \wedge Y)$	$F\,X$	$F\,Y$
$T\,(X \vee Y)$	$T\,X$	$T\,Y$
$T\,(X \supset Y)$	$F\,X$	$T\,Y$
$\neg(X \wedge Y)$	$\neg X$	$\neg Y$
$(X \vee Y)$	X	Y
$(X \supset Y)$	$\neg X$	Y

The idea is, in any interpretation an α is true if and only if both α_1 and α_2 are true; and a β is true if and only if at least one of β_1 or β_2 is true. (A signed formula $T\,X$ is true in an interpretation if X has the value *true*; likewise $F\,X$ is true if X has the value *false*.)

Today negation is often left out of the conjunctive/disjunctive classification, not because it is mathematically inappropriate, but because it introduces redundancy if one is attempting to automate semantic tableaus. This was not a concern of Smullyan's. In [29] these tables are extended to include all other binary connectives except for equivalence and exclusive-or (the dual to equivalence), which follow a different pattern. But if one is willing to weaken the subformula principle somewhat, even these can be included, using the following definitions (in which we use $\not\equiv$ for exclusive-or).

α	α_1	α_2
$T\,(X \equiv Y)$	$T\,(X \supset Y)$	$T\,(Y \supset X)$
$F\,(X \not\equiv Y)$	$T\,(X \supset Y)$	$T\,(Y \supset X)$
$(X \equiv Y)$	$(X \supset Y)$	$(Y \supset X)$
$\neg(X \not\equiv Y)$	$(X \supset Y)$	$(Y \supset X)$

β	β_1	β_2
$F\,(X \equiv Y)$	$F\,(X \supset Y)$	$F\,(Y \supset X)$
$T\,(X \not\equiv Y)$	$F\,(X \supset Y)$	$F\,(Y \supset X)$
$\neg(X \equiv Y)$	$\neg(X \supset Y)$	$\neg(Y \supset X)$
$(X \not\equiv Y)$	$\neg(X \supset Y)$	$\neg(Y \supset X)$

The effects of uniform notation are quite lovely: all the classical propositional tableau rules for extending branches reduce to the following pair.

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

Smullyan took this abstract approach considerably further, eventually doing away with formulas altogether. In [85] the essence of the tableau approach to classical logic was distilled, and in [86] this was extended, to intuitionistic and modal logic.

2.5.2 The Role of Signs

From the beginning of the subject the connection between tableaus and the sequent calculus was clear. Loosely, a tableau proof is a sequent proof backwards. In the sequent calculus we show a sequent is valid; in a tableau system we show a formula (or a set of formulas) is unsatisfiable. So, in order to make the sequent/tableau relationship clear, we need a suitable translation between sequents and (finite) sets of formulas. In one direction things are simple: map the sequent $X_1, \ldots, X_n \rightarrow Y_1, \ldots, Y_k$ to the set $\{X_1, \ldots, X_n, \neg Y_n, \ldots, \neg Y_k\}$. Then the sequent is valid if and only if the corresponding set is unsatisfiable. But a problem arises in going the other way, from a set to a sequent. Given the set $\{X, \neg Y\}$, say, it could have come from the sequent $X \rightarrow Y$ or from $X, \neg Y \rightarrow$, and for more complex sets the number of possibilities can be much greater. Thus we have a many-one mapping. While one can work with this, it makes things unnecessarily complicated. Signed formulas deal with this problem nicely.

Following Smullyan [84], if $S = \{T\,X_1, \ldots, T\,X_n, F\,Y_1, \ldots, F\,Y_k\}$ is a set of signed formulas, let $|S|$ be the sequent $X_1, \ldots, X_n \rightarrow Y_1, \ldots, Y_k$. If we think of the strings of formulas on the left and the right of the sequent arrow as *sets* rather than as *sequences*, thus ignoring the structural rules, this defines a one-one translation between sequents and sets of signed formulas. What is more, using uniform notation, the sequent calculus rules can be presented in the following abstract form (again, omitting the structural rules).

Axioms $\qquad |S, T\,X, F\,X|$

Inference Rules $\dfrac{|S, \alpha_1, \alpha_2|}{|S, \alpha|}$ $\qquad$ $\dfrac{|S, \beta_1| \quad |S, \beta_2|}{|S, \beta|}$

This more abstract approach makes it possible, for instance, to give a uniform proof of cut elimination, one that applies to both tableaus and the sequent calculus, rather than proving it for one and deriving it for the other as a consequence. From this point of view, tableau and sequent proofs *are* the same thing, which is what everyone suspected all along.

2.5.3 Cut and Analytic Cut

Just as with the sequent calculus, one can introduce a Cut rule for tableaus, and show it can be eliminated from proofs. The sequent calculus formulation of Cut was given in section 2.1.2; for tableaus it has a much simpler appearance. Here are signed and unsigned versions.

Cut $\dfrac{}{T\,X \mid F\,X}$ $\qquad$ $\dfrac{}{X \mid \neg X}$

That is, at any time during a signed tableau construction, a branch may be split, with $T\,X$ added to one fork, and $F\,X$ added to the other, for *any* formula X, and similarly for the unsigned version. Clearly this violates the subformula principle. But Smullyan also considered what he called *Analytic Cut*, which is simply the Cut rule as given above, but with the restriction that X must be a subformula of some formula already appearing on the branch. Like unrestricted Cut, Analytic Cut also shortens proofs, and it clearly does not violate the subformula principle. It lends itself well to automation, and has played some role in this area.

2.6 The Complications Quantifiers Add

In our survey of various proof systems above, we ignored quantifiers. The reason is simple: all the systems treat quantifiers more-or-less the same way, so differences between systems can be illustrated sufficiently well at the propositional level. Now it is time to say a little about them. (A full treatment of first-order tableaus can be found in Chapter **??**.)

Quantifier rules for classical logic are deceptively simple. If $(\forall x)\varphi(x)$ is true in some model, then $\varphi(c)$ is also true for any closed term c. On the other hand, if $(\forall x)\varphi(x)$ is false in a model, there is some member of the domain of the model for which $\varphi(x)$ does not hold. Then, if d is a new constant symbol, we can interpret it to designate some member of the domain for which $\varphi(x)$ fails, and so $\varphi(d)$ will be false in the model. Note that the model in which $\varphi(d)$ is false is not the original model, since we had to re-interpret d, but since d was chosen to be a constant symbol that was new to the proof, this has no effect on how formulas already appearing are interpreted.

Now we give quantifier tableau rules, using Smullyan's uniform notation. In stating things we use the informal convention that, if $\varphi(x)$ is a formula and c is a constant symbol, $\varphi(c)$ is like $\varphi(x)$ but with occurrences of c substituted for all free occurrences of the variable x. Quantified formulas are classified into type C, universal, and type D, existential, with γ as generic type C, and δ as generic type D.

γ	$\gamma(c)$
$T\,(\forall x)\varphi(x)$	$T\,\varphi(c)$
$F\,(\exists x)\varphi(x)$	$F\,\varphi(c)$
$(\forall x)\varphi(x)$	$\varphi(c)$
$\neg(\exists x)\varphi(x)$	$\neg\varphi(c)$

δ	$\delta(c)$
$T\,(\exists x)\varphi(x)$	$T\,\varphi(c)$
$F\,(\forall x)\varphi(x)$	$F\,\varphi(c)$
$(\exists x)\varphi(x)$	$\varphi(c)$
$\neg(\forall x)\varphi(x)$	$\neg\varphi(c)$

Now, the quantifier rules are these.

$$\frac{\gamma}{\gamma(c)}$$

Where c is any constant symbol whatever.

$$\frac{\delta}{\delta(c)}$$

Where c is a constant symbol that is new to the branch.

Here is an example of a classical first-order proof, of $(\forall x)(\forall y)R(x,y) \supset (\forall z)R(z,z)$. In it, 2 and 3 are from 1 by $F\supset$; 4 is from 3 by $F\forall$; 5 is from 2 by $T\forall$; and 6 is from 5 by $T\forall$. Notice that when the $F\forall$ rule was applied, the constant symbol c had not yet been used.

1. $F\,(\forall x)(\forall y)R(x,y) \supset (\forall z)R(z,z)$
2. $T\,(\forall x)(\forall y)R(x,y)$
3. $F\,(\forall z)R(z,z)$
4. $F\,R(c,c)$
5. $T\,(\forall y)R(c,y)$
6. $T\,R(c,c)$

For a sequent calculus formulation things are reversed from the tableau version. Existential quantifiers, instead of introducing new constant symbols, remove constant symbols. Using the notation of section 2.5.2, here are sequent rules for classical quantifiers.

$$\frac{|S, \gamma(c)|}{|S, \gamma|}$$

Where c is any constant symbol.

$$\frac{|S, \delta(c)|}{|S, \delta|}$$

Where c is a constant symbol that does not occur in $\{S, \delta\}$.

As we said, the quantifier rules are deceptively simple. Since there are (we assume) infinitely many constant symbols available, if the γ-rule can be applied at all, it can be applied in infinitely many different ways. This means a tableau can never be completed in a finite number of steps. Essentially, this is the source of the undecidability of classical first-order logic.

3 Modern History

After reaching a stable form in the classical case, the next stage in the development of tableaus was the extension to various non-classical logics. In this section we sketch a few such systems, say how they came about, and present the intuitions behind them. The particular systems chosen illustrate the variety of extra machinery that has been developed for and added to tableau systems: reinterpreting signs, generalizing signs, modifying closure rules, allowing trees to change in ways other than simple growth, adjoining 'side' information, and using pairs of coupled trees.

3.1 Intuitionistic Logic

Sequent calculi for intuitionistic logic were around from the beginning—Gentzen and Beth both developed them—so it is not surprising that a tableau version would be forthcoming (see Chapter **??**). The first explicitly presented as such seems to be in the 1969 book of Fitting [24]. In this the signed tableau system of Smullyan was adapted, with the signs given a new informal interpretation. In the resulting tableau system, proof trees were allowed to shrink as well as grow.

For both Lis and Smullyan, signs primarily were a device to keep track of left and right sides of sequents, without explicitly using sequent notation. Signs also had an intuitive interpretation that was satisfying: $T\,X$ and $F\,X$ can be thought of as asserting that X is true or false in a model. But now, think of $T\,X$ as informally meaning that X is *intuitionistically* true, that is, X has been given a proof that an intuitionist would accept. Likewise think of $F\,X$ as asserting the opposite: X has not been given an intuitionistically acceptable proof. (This is quite different from assuming X is intuitionistically refutable, by the way.) Some tableau rules are immediately suggested. For instance, intuitionists read disjunction constructively: to prove $X \vee Y$ one should either prove X or prove Y (see [39]). Then if

we have $T\,X \vee Y$ in a tableau, informally $X \vee Y$ has been intuitionistically proved, hence this is the case for one of X or Y, so the tableau branch splits to $T\,X$ and $T\,Y$. Likewise if we have $F\,X \vee Y$, we do not have an intuitionistically acceptable proof of $X \vee Y$, so we can have neither a proof of X nor of Y, and so we can add both $F\,X$ and $F\,Y$ to the branch. That is, intuitionistic rules for disjunction look like classical ones! The same is the case for conjunction. But things begin to get interesting with implication. We quote Heyting [39].

> "The *implication* $\mathfrak{p} \to \mathfrak{q}$ can be asserted, if and only if we possess a construction $\mathfrak{r}$, which, joined to any construction proving $\mathfrak{p}$ (supposing that the latter be effected), would automatically effect a construction proving $\mathfrak{q}$. In other words, a proof of $\mathfrak{p}$, together with $\mathfrak{r}$, would form a proof of $\mathfrak{q}$."

If $T\,X \supset Y$ occurs in a tableau, informally we have a proof of $X \supset Y$, and so we have a way of converting proofs of X into proofs of Y. Then, in our present state of knowledge, either we are not able to prove X, or we are, in which case we can provide a proof of Y as well. That is, the tableau branch splits to $F\,X$ and $T\,Y$, just as it does classically.

Now suppose $F\,X \supset Y$ occurs in a tableau. Then intuitively, we do not have a mechanism for converting proofs of X into proofs of Y. This does not say anything at all about whether we are able to prove X. What it says is that someday, not necessarily now, we may discover a proof of X without being able to convert it to a proof of Y. That is, *someday* we could have both $T\,X$ and $F\,Y$. We are talking about a possible future state of our mathematical lives. Now, as we move into the future, what do we carry with us? If we *have not* proved some formula Z, this is not necessarily a permanent state of things—tomorrow we may discover a proof. But if we *have* proved Z, tomorrow this will still be so—a proof remains a proof. Thus, when passing from a state to a possible future state, signed formulas of the form $T\,Z$ should remain with us; signed formulas of the form $F\,Z$ need not. This suggests the following tableau rule: if a branch contains $F\,X \supset Y$, add both $T\,X$ and $F\,Y$, but first delete all signed formulas on the branch that have an F sign. (Negation has a similar analysis.)

We must be a little careful with this notion of formula deletion, though. The tree representation for tableaus that we have been using marks the presence of a node by using a formula as a label. If we simply delete formulas, information about node existence and tableau structure could be lost. What we do instead, when using this representation of tableaus, is leave deleted formulas in place, but check them off, placing a $\surd$ in front of them. (Of course, when using the set of sets representation for tableaus from section 1.2, things are simpler: just replace one set by another, since there is no structure sharing.) There is still one more problem, though. It may happen that a formula should be deleted on one branch, but not

on another, and using the tree representation for tableaus, its presence on both branches might be embodied in a single formula occurrence. In this case, check it off where it occurs, and add a fresh, unchecked occurrence to the end of the branch on which it should not be deleted.

To state the intuitionistic rules formally we use notation from [24], if S is a set of signed formulas, let S_T be the set of T-signed members of S. We write $S, F\,Z$ to indicate a tableau branch containing the signed formula $F\,Z$, with S being the set of remaining formulas on the branch. Now, the propositional intuitionistic rules are these.

Conjunction

$$\frac{S, T\,X \wedge Y}{S, T\,X, T\,Y} \qquad \frac{S, F\,X \wedge Y}{S, F\,X \mid S, F\,Y}$$

Disjunction

$$\frac{S, T\,X \vee Y}{S, T\,X \mid S, T\,Y} \qquad \frac{S, F\,X \vee Y}{S, F\,X, F\,Y}$$

Implication

$$\frac{S, T\,X \supset Y}{S, F\,X \mid S, T\,Y} \qquad \frac{S, F\,X \supset Y}{S_T, T\,X, F\,Y}$$

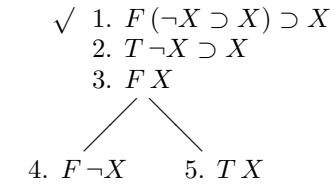
Negation

$$\frac{S, T\,\neg X}{S, F\,X} \qquad \frac{S, F\,\neg X}{S_T, T\,X}$$

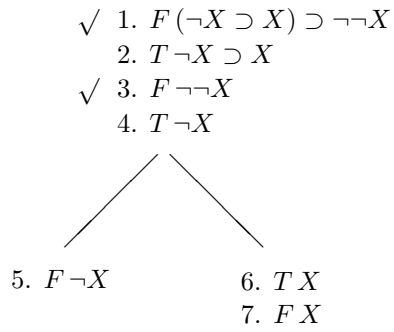
Then unlike with classical tableaus, as far as usable formulas are concerned, intuitionistic tableaus can shrink as well as grow. If a branch contains, say, both $F\,X \supset Y$ and $F\,A \supset B$, using an implication rule on one will destroy the other. It is possible to make a bad choice at this point and miss an available proof. For completeness sake, both possibilities must be explored. This is the analog of Beth's *disjunctive* branching, see section 2.2.2. As a matter of fact, the tableau rules above correspond to Beth's rules in the same way that the classical tableau rules of Lis and Smullyan correspond to Beth's classical rules.

A tableau branch is closed if it contains $T\,X$ and $F\,X$, for some formula X, *where neither is a deleted signed formula*. Of course the intuitive idea is somewhat different than in the classical case now: the contradiction is that an intuitionist has both verified and failed to verify X. Nonetheless, a contradiction is still a contradiction.

We conclude with two examples, a non-theorem and a theorem. The non-theorem is $(\neg X \supset X) \supset X$. A tableau proof attempt begins with $F\,(\neg X \supset X) \supset X$. An application of the $F \supset$ rule causes this very F-signed formula to be deleted, and produces 2 and 3 below. Use of $T \supset$ on 2 gives 4 and 5. The right branch is closed, but closure of the left branch is impossible since an application of $F\neg$ to 4 causes deletion of 3.



The formula $(\neg X \supset X) \supset \neg\neg X$, on the other hand, is a theorem. A proof of it begins with $F\,(\neg X \supset X) \supset \neg\neg X$, then continues as follows.



An application of the $F \supset$ rule deletes 1 and adds 2 and 3. Then $F\neg$ applied to 3 deletes it and adds 4. The $T \supset$ rule applied to 2 adds 5 and 6, and finally, the $T\neg$ rule applied to 4 adds 7. The tableau is closed because of 4 and 5, and 6 and 7, none of which are checked off.

At the cost of a small increase in the number of signs, Miglioli, Moscato, and Ornaghi have created a tableau system for intuitionistic logic that is more efficient than the one presented above [56, 57, 58]. In addition to the signs T and F, one more sign, F_c, is introduced. In terms of Kripke models, we can think of $T\,X$ as true at a possible world if X is true there, in the sense customary with intuitionistic semantics. Likewise we can think of $F\,X$ as true at a possible world if X is *not* true there. But now, think of $F_c\,X$ as true at a world if $\neg X$ is true there. This requires additional tableau rules, but reduces duplications inherent in the tableau system without the additional sign. Without going into details, this should suggest some of the flexibility made possible by the use of signed formulas, a topic to be continued in the next section.

3.2 Many-Valued Logic

The signs of a signed tableau system can be reinterpreted, as in intuitionistic logic, and they can be extended, as happened in many-valued logic (see Chapter **??**). Finitely-valued Łukasiewicz logics were given a tableau treatment by Suchon in 1974 [87]. Surma considered a more general situation in 1977, [88], and this was further developed by Carnielli in 1987,

[17, 18]. (The paper [17] contains an error in the quantifier rules which is corrected in [18].) In these papers, the two signs of Lis and Smullyan were extended to a larger number, with one sign for each truth value of the logic. Essentially, this is the tableau version of the many-valued sequent calculus of [75], discussed in section 2.1.4. To show a formula X is a theorem, one must construct a closed tableau for $V\,X$, where V is a sign corresponding to the truth value v, for each non-designated value v.

While this is a natural idea, in practice it means several tableaus may need to be constructed for a single validity proof. Also, the rules themselves tend to be complicated. Suppose, for instance, we consider Kleene's strong three-valued logic, a well known logic. We can take as the three truth values $\{\mathit{false}, \bot, \mathit{true}\}$, where $\bot$ is intended to represent 'unknown.' Disjunction is easily characterized: order the truth values by: $\mathit{false} < \bot < \mathit{true}$; then disjunction-of is simply maximum-of. Suppose we introduce F, U, and T as signs corresponding to the three truth values. Then one of the Carnielli rules for disjunction is the following.

$$\frac{U\,X \vee Y}{\begin{array}{c|c|c} F\,X & U\,X & U\,X \\ U\,Y & U\,Y & F\,Y \end{array}}$$

More recently, Hähnle showed that many-valued logics could often be treated more efficiently by tableaus if *sets* of truth values were used as signs, rather than single truth values, [34, 35, 36]. Think of $S\,X$, where S is a set, as asserting that the truth value of X (in some model) is a member of S. Then to show X is valid one needs a single closed tableau, beginning with $\overline{D}\,X$, where $\overline{D}$ is the set of non-designated truth values. The following is a typical example of a Hähnle style rule, again for the strong Kleene logic.

$$\frac{\{F,U\}\,X \vee Y}{\{F,U\}\,X \mid \{F,U\}\,Y}$$

As this example suggests, once sets are used as signs, a generalized uniform notation becomes possible. In many cases, this works quite well, [35]. We have not discussed quantification, which is a central issue in many-valued logics.

3.3 Modal Logic

What is now called *relational semantics* for modal logic was fully developed by the mid-sixties, drawing on work of Kanger [46], Hintikka [42, 43], and Kripke [48, 49, 50, 51]. This led to a renewed interest in modal logic itself, and to the development of tableau systems for various such logics (covered in Chapter **??**). See [30] for a general overview of the subject. Kripke himself gave Beth-style tableaus for the modal logics he treated. In fact, his completeness proofs for axiom systems proceeded by showing

equivalence to Beth tableau systems (using cut elimination) then proving completeness for these by a systematic tableau style construction. This was complex and hard to follow, and soon Henkin-style completeness arguments became standard. A tree-style system for S4 appeared in [24], similar to the intuitionistic system of that book. Systems for several modal logics, based on somewhat different principles, appeared in [25], and for temporal logics in [73]. But the most extensive development was in Fitting's 1983 book [26] which, among other things, gave tableau systems for dozens of normal and non-normal modal systems. We sketch a few to give an idea of the style of treatment, and the intuition behind the tableaus.

3.3.1 Destructive Tableau Systems

Fitting extended Smullyan's uniform notation to the modal case. Here is a signed-formula version. The idea is: a ν formula is true at a possible world if and only if the corresponding ν_0 is true at every accessible world; a π formula is true at a possible world if the corresponding π_0 is true at some accessible world.

ν	ν_0
$T\,\Box X$	$T\,X$
$F\,\Diamond X$	$F\,X$

π	π_0
$F\,\Diamond X$	$F\,X$
$T\,\Box X$	$T\,X$

Next, if S is a set of signed formulas, a set $S^{\#}$ is defined. The idea is, if the members of S are true at a possible world, and we move to a 'generic' accessible world, the members of $S^{\#}$ should be true there. The definition of $S^{\#}$ differs from modal logic to modal logic. We give the version for K, the smallest normal modal logic.

$$S^{\#} = \{\nu_0 \mid \nu \in S\}$$

The rules for K are exactly as in the classical Smullyan system, together with the following 'destructive' rule.

$$\frac{S, \pi}{S^{\#}, \pi_0}$$

Unlike the other rules (but exactly like the intuitionistic rules in section 3.1), this one modifies a whole branch. If S, π is the set of signed formulas on a branch, the whole branch can be replaced with $S^{\#}, \pi_0$. Since this removes formulas, and modifies others, it is an information-loosing rule, hence the description 'destructive.' We continue to use the device of checking off deleted formulas in trees.

Here is a simple example of a proof in this system, of $\Box(X \supset Y) \supset (\Box X \supset \Box Y)$. It begins as follows.

$\surd$	1.	$F\,\Box(X \supset Y) \supset (\Box X \supset \Box Y)$
$\surd$	2.	$T\,\Box(X \supset Y)$
$\surd$	3.	$F\,\Box X \supset \Box Y$
$\surd$	4.	$T\,\Box X$
$\surd$	5.	$F\,\Box Y$
	6.	$T\,X \supset Y$
	7.	$T\,X$
	8.	$F\,Y$

Here 2 and 3 are from 1, and 4 and 5 are from 3 by $F\supset$. Now take 5 as π, and 1 through 4 as S, and apply the modal rule. Formula 1 is simply deleted; 2 is deleted but 6 is added; 3 is deleted; 4 is deleted but 7 is added (at this point, S has been replaced by $S^{\#}$); and finally 5 is deleted but 8 is added (this is π_0). Now an application of $T\supset$ to 6 produces a closed tableau.

The underlying intuition is direct. All rules, except the modal one, are seen as exploring truth at a single world. The modal rule corresponds to a move from a world to an alternative one. A soundness argument can easily be based on this. Completeness can be proved using either a systematic tableau construction or a maximal consistent set approach. As is the case with both classical and intuitionistic tableaus, interpolation theorems and related results can be derived from the tableau formulation.

Several other normal modal logics can be treated by modifying the definition of $S^{\#}$, or by adding rules, or both. For instance, the logic $K4$ (adding transitivity to the model conditions) just requires a change in a definition, to the following.

$$S^{\#} = \{\nu, \nu_0 \mid \nu \in S\}$$

Generally speaking, modal logics that have tableau systems of this kind can not have a semantics whose models involve *symmetry* of the accessibility relation. Interestingly enough, though, such logics can often be given tableau systems in this style if a cut rule is allowed, and in fact a *semi-analytic* version is enough. Semi-analyticity extends and weakens the notion of analytic cut, but is still not as broad as the unrestricted version. See Fitting, [26], for more details.

Various regular but non-normal logics can be dealt with by restricting rule applicability (see [26, 30] for a definition of regularity). For instance, if we use the system for K above, but restrict the modal rule to those cases in which $S^{\#}$ is non-empty, we get a tableau system for the smallest regular logic C. It is even possible to treat such quasi-regular logics as $S2$ and $S3$ by similar techniques.

3.3.2 Making Accessibility Explicit

In the various tableau systems described in the previous section, possible worlds were implicit, not explicit. Other approaches have brought possible worlds visibly into the picture. Hughes and Cresswell [44], for instance, have a system of *diagrams* which are tableau-like, and involve boxes representing possible worlds, with arrows representing accessibility. In 1972 Fitting [25] gave tableau systems using *prefixes* in which the idea was to designate possible worlds in such a way that syntactical rules determined accessibility. In a straightforward way, prefixes correspond to the Hughes and Cresswell boxes. The notion of prefixes is at its simplest for $S5$, where we can take as prefixes just natural numbers. The resulting system can be seen as a direct descendant of the sequent system of Kanger, discussed in section 2.1.4.

A *prefixed signed formula* for $S5$ is just $n\,Z$, where n is a non-negative integer, and Z is a signed formula. The α- and β- rules from section 2.5.1 are modified in a direct way: conclude $n\,\alpha_1$ and $n\,\alpha_2$ from $n\,\alpha$, and similarly for β. The following modal rules are used (similarity to quantifier rules is intentional). This time there is no notion of formula deletion.

$$\frac{n\,\nu}{k\,\nu_0}$$

Where k is any non-negative integer.

$$\frac{n\,\pi}{k\,\pi_0}$$

Where k is any non-negative integer that is new to the branch.

Prefixes should be thought of as names for possible worlds. The system for $S5$ is particularly simple because the logic is characterized by models in which every world is accessible from every other. For other modal logics *sequences* of integers are used, where the intuition is: *extension-of* corresponds to *accessible-from*. This builds on an abstract of Fitch concerning modal natural deduction systems. See [25, 26] for details.

There are many ways explicit reference to possible worlds can be incorporated into tableaus. The most obvious is to simply record accessibility information directly, in a side table. Other techniques have also been used, motivated by automation concerns. By such methods very general theorem proving mechanisms can be created. There is a drawback however. One of the nice features of tableaus is the extra information they can provide about the logic, most notably, proofs of interpolation theorems. Such proofs are not available once explicit possible worlds appear. On the other hand, decision procedures can often be easier to describe and program, using the additional machinery.

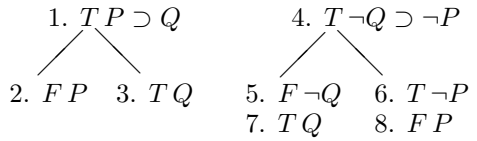
3.4 Relevance Logic

Fault has been found with tautologies like $(P \wedge \neg P) \supset Q$, since the antecedent and the consequent are not related to each other. A sense of dissatisfaction with such things led to the creation of the family of *relevance logics*; see [22] for a survey of the subject. While the semantics for relevance logic is generally quite complicated, that for so-called *first-degree entailment* is rather simple—this is the fragment in which one considers only formulas of the form $A \supset B$ where A and B do not contain implications. The reason this is of interest here is that the method of 'coupled trees' developed for it shows yet another way of working with tableaus.

Smullyan presented a system of 'linear reasoning' for first-order classical logic in [84]. The system was tableau based, and was motivated by Craig's original proof of his Interpolation Theorem. Something very much like this, but for propositional classical logic only, appeared in a pedagogically nice form in Jeffrey [45], under the name of *coupled trees*. It is Jeffrey's version with which we begin.

Suppose we want to give a classical propositional logic proof of $X \supset Y$, but instead of constructing a closed tableau for $F\,X \supset Y$, we do the following. Completely construct *two* tableaus, $\mathcal{T}_1$ for $T\,X$, and $\mathcal{T}_2$ for $T\,Y$. The open branches of $\mathcal{T}_1$ represent all the ways in which X could be true, and similarly for $\mathcal{T}_2$. Let us say a branch θ_1 *covers* a branch θ_2 if every signed atomic formula on θ_2 also occurs on θ_1. Suppose every open branch of $\mathcal{T}_1$ covers some open branch of $\mathcal{T}_2$—intuitively, each of the ways X could be true must also be a way in which Y is true. Then we have argued for the validity of $X \supset Y$. In effect, we are thinking of the proof of $X \supset Y$ as beginning with X, breaking it down by constructing $\mathcal{T}_1$ for $T\,X$, making the transition from $\mathcal{T}_1$ to $\mathcal{T}_2$ using the covering condition, then building up to Y using the tableau $\mathcal{T}_2$ for $T\,Y$ backwards. Such a proof technique is complete for implications.

To illustrate the technique, here is a coupled tableau proof of $(P \supset Q) \supset (\neg Q \supset \neg P)$.



We do not describe the construction of the two tableaus, which is elementary. But note that the left branch of tableau one covers the right branch of tableau two, via formulas 2 and 8, and the right branch of tableau one covers the left branch of tableau two, via formulas 3 and 7. Thus we have a correctly constructed coupled tableau argument.

There are some problems with this intuitively simple idea, however. The formula $P \supset (Q \vee \neg Q)$ is classically valid, though not provable by the technique described above. To get around this Jeffrey added a device that amounts to allowing applications of the cut rule in the tableau for the antecedant of an implication. Also, closed branches are ignored and, while this may seem natural at first encounter, even closed branches contain information. It is the restriction of the covering condition to only open branches that allows a coupled tree argument for $(P \wedge \neg P) \supset Q$, which is the standard example of a tautology not acceptable to relevance logicians.

Dunn's proposal in [21] is to simplify Jeffrey's system to the extreme. To show $X \supset Y$, completely construct tableaus $\mathcal{T}_1$ for $T\,X$ (not using the cut rule) and $\mathcal{T}_2$ for $T\,Y$, and see if *every* branch of $\mathcal{T}_1$, closed or not, covers a branch of $\mathcal{T}_2$. Dunn showed this gave a sound and complete proof procedure for first-degree entailment.

What sense can be made semantically of using tableau branches even if closed? Suppose, instead of working in the ideal setting of classical logic, we work in something more like the real world. We may have information that tells us a proposition P is *true*, or *false*. But equally well, we may have no information about P at all, neither *true* nor *false*, or we may have contradictory information, both *true* and *false*. In effect we are using a four-valued logic whose truth values are all subsets of $\{false, true\}$. This is a logic that was urged as natural for computer science in Belnap [6]. Dunn showed that a simple semantics for first-degree entailment could be given using this four-valued logic: $X \supset Y$ is a valid first-degree entailment if and only if, under every valuation v in the four-valued logic, if $v(X)$ is at least true (has *true* as a member) then $v(Y)$ is also at least true.

The relationship between the four-valued logic and tableaus is simple: if a branch θ of a tableau has had all applicable tableau rules applied to it we can think of θ as determining a four-valued valuation as follows. Map P to $\{true\}$ if $T\,P$ is on θ but $F\,P$ is not; map P to $\{false\}$ if $F\,P$ is on θ but $T\,P$ is not; map P to $\emptyset$ if neither $T\,P$ nor $F\,P$ is on θ; and map P to $\{false, true\}$ if both $T\,P$ and $F\,P$ are on θ. Indeed, a similar 'ambiguation' can be developed starting with many-valued logics other than the classical, two-valued one. The technique is fairly general.

There are other approaches to relevance logic that make use of tableaus, each with additional features of interest. Hähnle [36] gives a formulation of first-degree entailment using many-valued tableaus with sets of truth values as signs; see 3.2. Also, Schröder [77] gives a tableau system in which additional bookkeeping machinery is introduced to check that each occurrence of a propositional variable was actually used to close a branch. Relevance logic itself is part of the more general subject of *substructural logic*, which is covered in Chapter **??**.

4 Post-Modern History

In our subject, as in every other, post-modernism begins before modernism ends, in fact, before it starts. By the *post-modern period* for tableaus we mean the period of their automation (involving issues discussed in Chapter **??**). Indeed, Beth had machine theorem-proving very much in mind [11], though this did not have a lasting influence. It is curious that resolution and tableaus in their current form appeared within a few years of each other [52, 74, 84]. Robinson, who invented resolution, was primarily interested in automation, Smullyan and Lis were not interested in automation at all. Perhaps this accounts for much of the emphasis on resolution in the automated theorem-proving community. But another determinant was more technical—nobody seems to have connected tableau methods and unification for a long time, without which only toy examples are possible. Still, all along there has been a subcurrent of interest in the uses of tableaus for automated theorem-proving—today it has become a major stream. We will sketch the swelling of this interest—it is complex, with basic ideas occurring independently several times. We will not bring our history up to today because present developments are many and are continuing to appear at a rapid rate. We lay the historical foundations for today's activity. We also confine the discussion to 'pure' tableau issues—we do not consider the increasingly fruitful relationships between tableaus and other theorem-proving mechanisms like connection graphs or resolution.

4.1 The Beginnings

Even though resolution has historically dominated automated deduction, among the first implemented theorem provers are some based on tableau ideas. In 1957–58, Dag Prawitz, Håkan Prawitz, and Neri Voghera developed a tableau-based system that was implemented on a Facit EDB [68]. At approximately the same time, the summer of 1958, Hao Wang proposed a family of theorem provers based on the sequent calculus, which he then implemented on an IBM 704 [95]. The first of Wang's programs, for classical propositional logic, proved all the approximately 220 propositional theorems of Russell and Whitehead's *Principia Mathematica* in 3 minutes! This was quite a remarkable achievement for 1958.

The γ, or universal quantifier, rule is a major source of difficulties for first-order tableau implementations. It allows us to pass from γ to $\gamma(t)$ for *any* t, but how do we know which t will be a useful choice? Eventually, unification solved this problem, as we will see in the next section, but unification was not available in the 1950's. Prawitz and his colleagues worked with a formalization having constant symbols but no function symbols, which simplified the structure of terms (representation of strings, terms, and formulas was non-trivial on these early machines.) Then the γ rule was implemented to simply instantiate to $\gamma(c)$ for every constant symbol

c that had been introduced by δ rule applications. In general, the set of such constant symbols grows without bound, which is the source of the undecidability of first-order logic. But also, as the authors note, this method creates many useless instantiations, and introduces an exponential growth factor into tableau construction, thus limiting the theorems provable by their implementation to rather simple examples. Wang discussed essentially the same idea, but did not actually carry out an implementation.

Instead of a full first-order system, Wang implemented a calculus for a decidable fragment—the AE formulas (with equality). A formula X is an AE formula if X is in prenex form, and all universal quantifiers precede any existential quantifiers. Since the rules for putting a formula into prenex form are not hard to mechanize, we can extend the definition to include those formulas that convert to AE form. The AE class is a decidable logic, and a complete tableau procedure for it is remarkably simple. Suppose, for instance, that we have a typical AE formula, $(\forall x_1)(\forall x_2)(\exists y_1)(\exists y_2)\varphi(x_1, x_2, y_1, y_2)$, and we attempt a tableau proof. We have two δ-rule applications to begin with, each introducing a new constant symbol, say c_1 and c_2. Thus the tableau begins as follows.

1. $F\,(\forall x_1)(\forall x_2)(\exists y_1)(\exists y_2)\varphi(x_1, x_2, y_1, y_2)$
2. $F\,(\forall x_2)(\exists y_1)(\exists y_2)\varphi(c_1, x_2, y_1, y_2)$
3. $F\,(\exists y_1)(\exists y_2)\varphi(c_1, c_2, y_1, y_2)$

Except for the γ case, classical tableau rules need only be applied to formulas once. Making use of this fact, we now have only γ-rules to apply. Suppose we apply them in all possible ways, *using only the constant symbols* c_1 *and* c_2 (four applications in all). After this, we only use propositional rules. It is rather easy to show that if this does not produce a proof, no proof is possible. More generally, without compromising completeness, γ-rule applications in proofs of AE formulas can be limited to constant symbols that were previously introduced by δ-rule applications, all of which must come first. It follows that a boundable number of γ-rule applications is always enough. (If there are no initial universal quantifiers, add a dummy one, then apply the procedure just outlined.)

Wang implemented the AE system just described. Of the 158 first-order propositions with equality in *Principia Mathematica* his program proved 139 of them. Subsequent modifications made possible proofs of all the theorems of *9 to *13 of *Principia* in about four minutes. Although Wang only worked with a decidable portion of first-order logic,

> "A rather surprising discovery, which tends to indicate our general ignorance of the extensive range of decidable subdomains, is the absence of any theorem of the predicate calculus in *Principia* which does not fall within the simple decidable subdomain of the AE predicate calculus."

It is not clear what this says about the work of Russell and Whitehead, but it is a curious discovery.

One interesting experiment that Wang undertook was to have the computer generate propositional formulas at random, test them for theoremhood, and print out those that passed an *ad hoc* test for being 'nontrivial.' In a way, the experiment was a failure, because 14,000 propositions were formed and tested in one hour, and 1000 were retained as nontrivial. The mass of data was simply too great. It is interesting just how hard it is to say what is interesting, and why.

This work seems to have had few direct successors. Possibly the introduction of the Davis-Putnam method, and then resolution, drew research attention elsewhere. Popplestone, in 1967 [66], implemented a Beth tableau style theorem prover and specifically noted its relationship with Wang's version. The universal quantifier rule was still seen as a central problem, and heuristics were introduced to deal with it. In 1978 Mogilevskii and Ostroukhov [60] implemented (in ALGOL) a Smullyan-style theorem prover, but only for propositional classical logic, though they mention variations for $S4$ and for intuitionistic logic.

4.2 Dummy Variables and Unification

It is universally recognized that the γ or universal quantifier rule is the most problematic for a first-order tableau implementation. The rule allows passage from γ to $\gamma(t)$ for *any* term t. Without guidance on what term or terms to choose, automation is essentially hopeless. Systematically trying everything would, of course, yield a complete theorem-prover, but one that is hopelessly inefficient. Wang avoided the problem by confining his theorem-prover to a subsystem of full first-order logic. In the meantime Robinson, anticipated by Herbrand, introduced *unification* into automated theorem-proving [74] and this, or its weaker cousin *matching*, serves very nicely as an appropriate tool for dealing with the universal quantifier problem. The idea, simply expressed, is to modify the γ-rule so that it reads: from γ pass to $\gamma(x)$, where x is a new free variable (a dummy, to use terminology from [67]). Then we use unification to discover what is a good choice for x—a good choice being something that will aid in tableau closure. For instance, if a branch contains $T\,P(t)$ and $F\,P(u)$, a substitution that unifies t and u will close the branch. What is wanted is a substitution that will simultaneously close all branches.

Unification was introduced independently into tableau theorem proving by several people, beginning with the 1974 paper of Cohen, Trilling, and Wegner [20]. While their paper was primarily devoted to presenting the virtues of ALGOL-68, it in fact gave a first-order theorem-prover based on Beth tableaus. It was written using a systematically-try-everything approach to the γ-rule, but then the introduction of Skolem functions and unification were specifically considered. This paper was followed in 1980

by Bowen [15, 14] and Broda [16] (neither of which seems to be aware of [20]), both motivated by logic programming issues. Bowen used a sequent calculus, though he noted relationships with work of Beth and Smullyan; Broda used semantic tableaus directly. Wrightson in 1984 [97], Reeves in 1985 [71], and Fitting in 1986 [27] also explicitly brought unification into the picture, while Oppacher and Suen in their HARP theorem-prover of 1988 [65] use matching, only moving to unification when "necessitated by the presence of complex terms."

The technique of using dummy variables to deal with universal quantifiers was described in too simple a way above. If we restrict things so that the rule, passing from γ to $\gamma(x)$, can be applied to a given formula only once, an incomplete theorem-prover results. No general upper limit on the number of applications can be set (or else first-order logic would be decidable). On the other hand, if we place no restrictions on which unifiers we can accept, an unsound system can result. The problem is this. Applications of the δ-rule require introduction of new constants. If we use free variables in γ-rules and delay determination of their ultimate values, we don't know what is new and what is not when δ-rule applications come up. Wrightson [97] and Reeves [71] deal with this difficulty essentially by imposing constraints on the unification process. Neither Bowen nor Broda discuss the issue explicitly, though they may have had a similar device in mind.

Matching with constraints gets quite complicated when function symbols are present. Of course the problem of what to do with the δ-rule can be easily avoided by Skolemizing away quantifier occurrences that would lead to δ-rule applications before the proof actually starts. If this is done, a rather simple sound and complete proof procedure combining tableaus and unification results. Such an approach was discussed by Reeves in [71]. An implementation of tableaus involving initial Skolemization, written in LISP, was presented by Fitting in 1986 [27].

4.3 Run-Time Skolemization

Above we noted that the combination of unification and tableaus leads to problems with the δ-rule, and that Skolemization provides one possible way out. Unfortunately, classical logic (more generally, many-valued logic with a finite number of truth values) is virtually the only first-order logic in which one can Skolemize a formula ahead of time. Tableau systems have been developed for a wide variety of logics, and this problem with the δ-rule could limit their usefulness. Fortunately there is a modification that works for many non-classical logics. It is commonly known as *run-time Skolemization*, a name whose significance will soon become apparent.

Suppose we are using the version of the γ-rule described in the previous section, passing from γ to $\gamma(x)$, where x is a new free variable. This kind of tableau system is sometimes referred to as a *free-variable tableau system*. In

applying the δ-rule, passing from δ to $\delta(t)$, we need to be able to guarantee that the term t will be new to the branch, no matter how free variables are instantiated. A simple way of ensuring this is to take for t the expression $f(x_1, \ldots, x_n)$, where f is a new function symbol and x_1, ..., x_n are all the free variables that occur on the branch. Clearly, we can think of the introduction of this term as part of a Skolemization process that goes on simultaneously with the tableau construction. In effect, Skolem functions come with an implicit 'time stamp' and this makes the technique suitable for many modal and similar logics.

Run-time Skolemization for tableaus seems to have first appeared in 1987 in Schmitt's THOT system [76], though it probably occurred to others around the same time. It is not necessary for classical logic, but its use does eliminate a preprocessing step, so it was included in the Prolog implementation of [29]. For non-classical logics, however, Skolemization ahead of time is generally impossible, so the run-time version used by Fitting in 1988 for modal logics [28] was essential.

Soon after Fitting's 1990 book [29] appeared, Hähnle and Schmitt noted that the version of run-time Skolemization used was unnecessarily inefficient. In [37] they showed it is enough to take as a rule: from δ pass to $\delta(f(x_1, \ldots, x_n))$ where x_1, ..., x_n are all the free variables *that occur in* δ. This work was extended in [4]. Subsequently Shankar [78] used a proof-theoretic analysis to show that a more complicated restriction on free variables—but still simpler than using all those that occur on a branch—suffices for first-order intuitionistic logic. Shankar also observed that his way of analyzing a tableau system will yield similar results for modal and other logics as well.

4.4 Where Now

If a logic has a tableau system at all, it probably will be the basis for the first theorem-prover to be implemented for it, though other kinds of theorem-provers may follow in time. For a given logic, tableaus may or may not turn out to be the best possible, most efficient approach to automated theorem-proving. Nevertheless, tableaus will continue to have a central role because they are relatively easy to develop, and in turn can be used to help create theorem-provers based on other methodologies. Many people have noted connections with resolution; in fact Maslov's method readily converts tableau systems to resolution-style systems [59]. There is a clear relationship with the connection method explored, among other places, in [97, 96]. Wallen's 1990 book on non-classical theorem-proving [94] exploits a relationship between tableaus and the matrix method. Examples continue to appear in the literature.

The development of theorem-provers that are not tableau-based, but are derived from them, is a topic of current research. The field is developing rapidly, so anything more specific we say about it will be out of date by the

time this appears in print. I don't know what comes after post-modernism generally, but for tableau theorem-proving, maybe we have reached it.

5 Conclusions

We have given a general overview of how tableaus began and developed. Their history is much like their appearance—branching and re-branching. Ideas occurred independently more than once; researchers influenced each other directly and indirectly. Details are less important than the general picture of tableau history, beginning with Gentzen, growing to encompass semantical ideas with Beth and Hintikka, becoming an elegant tool with Lis and especially Smullyan, extending to many logics, developing relationships with other proof techniques, and suggesting exciting automation possibilities. Some of these topics will be explored further in subsequent chapters.

With all this, there is much we have not discussed. Relationships with logic programming were not mentioned though connections are many ([31] will serve as a representative example). Equality is a central topic in logic, but we said little about it (see Chapter **??**). The system of Lis [52] includes a complete set of rules for equality, but it was little known at the time. Jeffrey's book [45] presents essentially the same rules. Reeves discusses the topic in [72] and there is a theoretical treatment in Fitting's book [29]. Much has occurred since then—we cannot discuss it adequately here. Higher order logic has not been mentioned, though tableau systems for it exist. Toledo, [92], investigates tableau systems for arithmetic that use the ω-rule, allowing infinite branching. Andrews, [1], develops tableaus for type theory. Smith, [80], presents two tableau systems for monadic higher-order logic. Van Heijenoort did research on tableau systems for higher order logic, but this has not yet been published.

The invention of tableau systems will continue, simply because they are easier to think of than other formulations. There is something inherently natural about them, whether they grow out of proof theory as with Gentzen, or out of semantics as with Beth and Hintikka. The increasing interest in non-classical theorem-proving has brought tableaus to a position of prominence, because they exist for many, many logics. The creation of logics, the development of tableau systems for them, all are very active areas of research. May the history of tableaus need rewriting in another generation.

Acknowledgements

I want to thank Perry Smith and Reiner Hähnle for their suggestions, which considerably improved this chapter.

References

[1] Andrews, P. B. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Academic Press, Orlando, Florida, 1986.

[2] Anellis, I. From semantic tableaux to smullyan trees: the history of the falsifiability tree method. *Modern Logic 1*, 1 (June 1990), 36–69.

[3] Anellis, I. Erratum, from semantic tableaux to smullyan trees: the history of the falsifiability tree method. *Modern Logic 2*, 2 (Dec. 1991), 219.

[4] Beckert, B., Hähnle, R., and Schmitt, P. H. The *even more* liberalized δ-rule in free variable semantic tableaux. In *Proceedings of the third Kurt Gödel Colloquium KGC'93, Brno, Czech Republic* (aug 1993), G. Gottlob, A. Leitsch, and D. Mundici, Eds., Springer LNCS 713, pp. 108–119.

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

[6] Belnap Jr., N. D. A useful four-valued logic. In *Modern Uses of Multiple-Valued Logic*, J. M. Dunn and G. Epstein, Eds. D. Reidel, Dordrecht and Boston, 1977, pp. 8–37.

[7] Beth, E. W. On Padoa's method in the theory of definition. *Indag. Math. 15* (1953), 330–339.

[8] Beth, E. W. Some consequences of the theorem of Löwenheim-Skolem-Gödel-Malcev. *Indag. Math. 15* (1953).

[9] Beth, E. W. Semantic entailment and formal derivability. *Mededelingen der Kon. Ned. Akad. v. Wet. 18*, 13 (1955). new series.

[10] Beth, E. W. Semantic construction of intuitionistic logic. *Mededelingen der Kon. Ned. Akad. v. Wet. 19*, 11 (1956). new series.

[11] Beth, E. W. On machines which prove theorems. *Simon Stevin Wissen-Natur-Kundig Tijdschrift 32* (1958), 49–60. Reprinted in [79] vol. 1, pp 79 – 90.

[12] Beth, E. W. *The Foundations of Mathematics*. North-Holland, Amsterdam, 1959. Revised Edition 1964.

[13] Bibel, W., Kurfess, F., Aspetsberger, K., Hintenaus, P., and Schumann, J. Parallel inference machines. In *Future Parallel Computers*, P. Treleaven and M. Vanneschi, Eds. Springer, Berlin, 1987, pp. 185–226.

[14] Bowen, K. Programming with full first-order logic. In *Machine Intelligence*, Hayes, Michie, and Pao, Eds., vol. 10. 1982, pp. 421–440.

[15] Bowen, K. A. Programming with full first order logic. Tech. Rep. 6-80, Syracuse University, Syracuse, NY, Nov. 1980.

[16] Broda, K. The relation between semantic tableaux and resolution theorem provers. Tech. Rep. DOC 80/20, Imperial College of Science

and Technology, London, Oct. 1980.

[17] Carnielli, W. A. Systematization of finite many-valued logics through the method of tableaux. *Journal of Symbolic Logic 52*, 2 (1987), 473–493.

[18] Carnielli, W. A. On sequents and tableaux for many-valued logics. *Journal of Non-Classical Logic 8*, 1 (1991), 59–76.

[19] Chang, C. C., and Keisler, H. J. *Model Theory*, third ed. North-Holland Publishing Company, 1990.

[20] Cohen, J., Trilling, L., and Wegner, P. A nucleus of a theorem-prover described in ALGOL-68. *International Journal of Computer and Information Sciences 3*, 1 (1974), 1–31.

[21] Dunn, J. M. Intuitive semantics for first-degree entailments and 'coupled trees'. *Philosophical Studies 29* (1976), 149–168.

[22] Dunn, J. M. Relevance logic and entailment. In *Handbook of Philosophical Logic*, D. Gabbay and F. Guenthner, Eds., vol. 3. Kluwer, Dordrecht, 1986, ch. III.3, pp. 117–224.

[23] Feys, R. *Modal Logics*. No. IV in Collection de Logique Mathématique, Sèrie B. E. Nauwelaerts (Louvain), Gauthier-Villars (Paris), 1965. Joseph Dopp, editor.

[24] Fitting, M. C. *Intuitionistic Logic Model Theory and Forcing*. North-Holland Publishing Co., Amsterdam, 1969.

[25] Fitting, M. C. Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic 13* (1972), 237–247.

[26] Fitting, M. C. *Proof Methods for Modal and Intuitionistic Logics*. D. Reidel Publishing Co., Dordrecht, 1983.

[27] Fitting, M. C. A tableau based automated theorem prover for classical logic. Tech. rep., Herbert H. Lehman College, Bronx, NY 10468, 1986.

[28] Fitting, M. C. First-order modal tableaux. *Journal of Automated Reasoning 4* (1988), 191–213.

[29] Fitting, M. C. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, 1990.

[30] Fitting, M. C. Basic modal logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, D. M. Gabbay, C. J. Hogger, and J. A. Robinson, Eds., vol. 1. Oxford University Press, Oxford, 1993, pp. 368–448.

[31] Fitting, M. C. Tableaux for logic programming. *Journal of Automated Reasoning 13* (1994), 175–188.

[32] Gentzen, G. Untersuchungen über das logische Schliessen. *Mathematische Zeitschrift 39* (1935), 176–210, 405–431. English translation, "Investigations into logical deduction," in [89].

[33] Girard, J.-Y. Linear logic. *Theoretical Computer Science 45* (1986), 159–192.

[34] Hähnle, R. Towards an efficient tableau proof procedure for multiple-valued logics. In *Proceedings Workshop on Computer Science Logic, Heidelberg* (1990), vol. 533 of *LNCS*, Springer-Verlag, pp. 248–260.

[35] Hähnle, R. Uniform notation of tableaux rules for multiple-valued logics. In *Proceedings International Symposium on Multiple-Valued Logic, Victoria* (1991), IEEE Press, pp. 238–245.

[36] Hähnle, R. *Automated Theorem Proving in Multiple-Valued Logics*, vol. 10 of *International Series of Monographs on Computer Science*. Oxford University Press, 1993.

[37] Hähnle, R., and Schmitt, P. H. The liberalized δ-rule in free variable semantic tableaux. *Journal of Automated Reasoning, to appear* (1993).

[38] Hertz, P. Über Axiomensysteme für beliebige Satzsysteme. *Mathematische Annalen 101* (1929), 457–514.

[39] Heyting, A. *Intuitionism, an Introduction*. North-Holland, Amsterdam, 1956. Revised Edition 1966.

[40] Hintikka, J. A new approach to sentential logics. *Soc. Scient. Fennica, Comm. Phys.-Math. 17*, 2 (1953).

[41] Hintikka, J. Form and content in quantification theory. *Acta Philosophica Fennica – Two Papers on Symbolic Logic 8* (1955), 8–55.

[42] Hintikka, J. Modality and quantification. *Theoria 27* (1961), 110–128.

[43] Hintikka, J. *Knowledge and Belief*. Cornell University Press, 1962.

[44] Hughes, G. E., and Cresswell, M. J. *An Introduction to Modal Logic*. Methuen and Co., London, 1968.

[45] Jeffrey, R. C. *Formal Logic: Its Scope and Limits*. McGraw-Hill, New York, 1967.

[46] Kanger, S. G. Provability in logic (Acta Universitatis Stockholmiensis, Stockholm Studies in Philosophy, 1). Almqvist and Wiksell, Stockholm, 1957.

[47] Kleene, S. C. *Introduction to Metamathematics*. D. Van Nostrand, North-Holland, P. Noordhoff, 1950.

[48] Kripke, S. A completeness theorem in modal logic. *Journal of Symbolic Logic 24* (1959), 1–14.

[49] Kripke, S. Semantical analysis of modal logic I, normal propositional calculi. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik 9* (1963), 67–96.

[50] Kripke, S. Semantical considerations on modal logics. *Acta Philosophica Fennica, Modal and Many-valued Logics* (1963), 83–94.

[51] Kripke, S. Semantical analysis of modal logic II, non-normal modal propositional calculus. In *The Theory of Models*, J. W. Addison, L. Henkin, and A. Tarski, Eds. North-Holland, Amsterdam, 1965, pp. 206–220.

[52] Lis, Z. Wynikanie semantyczne a wynikanie formalne (logical consequence, semantic and formal). *Studia Logica 10* (1960), 39–60. Polish, with Russian and English summaries.

[53] Manna, Z., and Waldinger, R. *The Logical Basis for Computer Programming*. Addison-Wesley, 1990. 2 vols.

[54] Manna, Z., and Waldinger, R. *The Deductive Foundations of Computer Programming*. Addison-Wesley, 1993.

[55] Matsumoto, K. Decision procedure for modal sentential calculus S3. *Osaka Mathematical Journal 12* (1960), 167–175.

[56] Miglioli, P., Moscato, U., and Ornaghi, M. An improved refutation system for intuitionistic predicate logic. Rapporto interno 37/88, Dipartimento di Scienze dell'Informazione, Universitá degli Studi di Milano, 1988.

[57] Miglioli, P., Moscato, U., and Ornaghi, M. How to avoid duplications in refutation systems for intuitionistic logic and Kuroda logic. Rapporto interno 99/93, Dipartimento di Scienze dell'Informazione, Universitá degli Studi di Milano, 1993.

[58] Miglioli, P., Moscato, U., and Ornaghi, M. An improved refutation system for intuitionistic predicate logic. *Journal of Automated Reasoning 13*, 3 (1994), 361–373.

[59] Mints, G. Proof theory in the USSR 1925 – 1969. *Journal of Symbolic Logic 56*, 2 (1991), 385–424.

[60] Mogilevskii, G. L., and Ostroukhov, D. A. A mechanical propositional calculus using Smullyan's analytic tables. *Cybernetics 14* (1978), 526–529. Translation from *Kibernetika*, 4, 43–46 (1978).

[61] Ohnishi, M. Gentzen decision procedures for Lewis's systems S2 and S3. *Osaka Mathematical Journal 13* (1961), 125–137.

[62] Ohnishi, M., and Matsumoto, K. Gentzen method in modal calculi I. *Osaka Mathematical Journal 9* (1957), 113–130.

[63] Ohnishi, M., and Matsumoto, K. Gentzen method in modal calculi II. *Osaka Mathematical Journal 11* (1959), 115–120.

[64] Ohnishi, M., and Matsumoto, K. A system for strict implication. *Annals of the Japan Assoc. for Philosophy of Science 2* (1964), 183–188.

[65] Oppacher, F., and Suen, E. HARP: A tableau-based theorem prover. *Journal of Automated Reasoning 4* (1988), 69–100.

[66] Popplestone, R. J. Beth-tree methods in automatic theorem-proving. In *Machine Intelligence*, N. L. Collins and D. Michie, Eds.,

vol. 1. American Elsevier, New York, 1967, pp. 31–46.

[67] Prawitz, D. An improved proof procedure. *Theoria 26* (1960). Reprinted in [79] vol. 1, pp 162 – 199.

[68] Prawitz, D., Prawitz, H., and Voghera, N. A mechanical proof procedure and its realization in an electronic computer. *Journal of the ACM 7* (1960), 102–128.

[69] Rasiowa, H. Algebraic treatment of the functional calculi of Heyting and Lewis. *Fundamenta Mathematica 38* (1951).

[70] Rasiowa, H. Algebraic models of axiomatic theories. *Fundamenta Mathematica 41* (1954).

[71] Reeves, S. V. *Theorem-proving by Semantic Tableaux*. PhD thesis, University of Birmingham, 1985.

[72] Reeves, S. V. Adding equality to semantic tableaux. *Journal of Automated Reasoning 3* (1987), 225–246.

[73] Rescher, N., and Urquhart, A. *Temporal Logic*. Springer-Verlag, 1971.

[74] Robinson, J. A. A machine-oriented logic based on the resolution principle. *Journal of the ACM 12* (1965), 23–41.

[75] Rousseau, G. Sequents in many valued logic I. *Fundamenta Mathematica 60* (1967), 23–33.

[76] Schmitt, P. H. The THOT theorem prover. Tech. Rep. TR–87.09.007, IBM Heidelbert Scientific Center, 1987.

[77] Schröder, J. Körner's criterion of relevance and analytic tableaux. *Journal of Philosophical Logic 21*, 2 (1992), 183–192.

[78] Shankar, N. Proof search in the intuitionistic sequent calculus. In *Automated Deduction — CADE-11*, D. Kapur, Ed., no. 607 in Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 1992, pp. 522–536.

[79] Siekmann, J., and Wrightson, G., Eds. *Automation of Reasoning*. Springer-Verlag, Berlin, 1983. 2 vols.

[80] Smith, P. Higher-Order Logic, Model Theory, Recursion Theory, and Proof Theory. Unpublished Manuscript, 1993.

[81] Smullyan, R. M. A unifying principle in quantification theory. *Proceedings of the National Academy of Sciences 49*, 6 (June 1963), 828–832.

[82] Smullyan, R. M. Analytic natural deduction. *Journal of Symbolic Logic 30* (1965), 123–139.

[83] Smullyan, R. M. Trees and nest structures. *Journal of Symbolic Logic 31* (1966), 303–321.

[84] Smullyan, R. M. *First-Order Logic*. Springer-Verlag, 1968. Revised edition, Dover Press, NY, 1994.

[85] Smullyan, R. M. Abstract quantification theory. In *Intuitionism and Proof Theory, Proceedings of the Summer Conference at Buffalo N. Y. 1968*, A. Kino, J. Myhill, and R. E. Vesley, Eds. North-Holland, Amsterdam, 1970, pp. 79–91.

[86] Smullyan, R. M. A generalization of intuitionistic and modal logics. In *Truth, Syntax and Modality, Proceedings of the Temple University Conference on Alternative Semantics*, H. Leblanc, Ed. North-Holland, Amsterdam, 1973, pp. 274–293.

[87] Suchoń, W. La méthode de Smullyan de construire le calcul n-valent des propositions de Łukasiewicz avec implication et négation. *Reports on Mathematical Logic, Universities of Cracow and Katowice 2* (1974), 37–42.

[88] Surma, S. J. An algorithm for axiomatizing every finite logic. In *Computer Science and Multiple-Valued Logics*, D. C. Rine, Ed. North-Holland, Amsterdam, 1977, pp. 143–149. Revised edition, 1984.

[89] Szabo, M. E., Ed. *The Collected Papers of Gerhard Gentzen*. North-Holland, Amsterdam, 1969.

[90] Tarski, A. Der Aussagenkalkül und die Topologie. *Fundamenta Mathematica 31* (1938), 103–34. Reprinted as 'Sentential calculus and topology' in [91].

[91] Tarski, A. *Logic, Semantics, Metamathematics*. Oxford, 1956. J. H. Woodger translator.

[92] Toledo, S. *Tableau Systems for First Order Number Theory and Certain Higher Order Theories*, vol. 447 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1975.

[93] Troelstra, A. S. *Lectures on Linear Logic*. No. 29 in CSLI Lecture Notes. CSLI, 1992.

[94] Wallen, L. A. *Automated Deduction in Nonclassical Logics*. The MIT Press, 1990.

[95] Wang, H. Toward mechanical mathematics. *IBM Journal for Research and Development 4* (1960), 2–22. Reprinted in *A Survey of Mathematical Logic*, Hao Wang, North-Holland, (1963), pp 224 – 268, and in [79], vol 1, pp 244 – 264.

[96] Wrightson, G. Non-classical theorem proving using links and unification in semantic tableaux. Tech. Rep. CSD-ANZARP-84-003, Victoria University, Wellington, NZ, 1984.

[97] Wrightson, G. Semantic tableaux, unification and links. Tech. Rep. CSD-ANZARP-84-001, Victoria University, Wellington, New Zealand, 1984.