INVESTIGATIONS INTO THE COMPLEXITY OF SOME PROPOSITIONAL CALCULI

by

Marcello D'Agostino

Technical Monograph PRG-88
ISBN 0-902928-67-8

November 1990

Oxford University Computing Laboratory
Programming Research Group
11 Keble Road
Oxford OX1 3QD
England

Oxford University Computing Laboratory
Programming Research Group
11 Keble Road
Oxford OX1 3QD
England

Electronic mail: marcello@prg.oxford.ac.uk (JANET)

Abstract

Cut-free Gentzen systems and their semantic-oriented variant, the tableau method, constitute an established paradigm in proof-theory and are of considerable interest for automated deduction and its applications. In this latter context, their main advantage over resolution-based methods is that they do not require reduction in clausal form; on the other hand, resolution is generally recognized to be considerably more efficient within the domain of formulae in clausal form.

In this monograph we analyse and develop a recently-proposed alternative to the tableau method, the system **KE** [Mon88a, Mon88b]. We show that **KE**, though being 'close' to the tableau method and sharing all its desirable features, is essentially more efficient. We trace this phenomenon to the fact that **KE** establishes a closer connection with the intended (classical, bivalent) semantics. In particular we point out, in Chapter 2, a basic 'redundancy' in tableau refutations (or cut-free Gentzen proofs) which depends on the form of the propositional inference rules and, therefore, affects *any* procedure based on them. In Chapter 3 we present the system **KE** and show that it is not affected by this kind of redundancy. An important property of **KE** is the *analytic cut property*: cuts cannot be eliminated but can be restricted to subformulae of the theorem, so that the subformula principle is still obeyed. We point out the close relationship between **KE** and the resolution method within the domain of formulae in clausal form. In Chapter 4 we undertake an analysis of the complexity of the propositional fragment of **KE** relative to the propositional fragments of other proof systems, including the tableau method and natural deduction, and prove some simulation and separation results. We show, among other things, that **KE** and the tableau method are separated with respect to the p-simulation relationship (**KE** can linearly simulate the tableau method, but the tableau method cannot p-simulate **KE**). In Chapter 5 we consider Belnap's four-valued logic, which has been proposed as an alternative to classical logic for reasoning in the presence of inconsistencies, and develop a tableau-based as well as a **KE**-based refutation system. Finally, in Chapter 6, we generalize on the ideas contained in the previous chapters and prove some more simulation and separation results.

Acknowledgements

This monograph consists of my thesis submitted for the degree of Doctor of Philosophy in the University of Oxford. I wish to thank my supervisors, Joe Stoy and Roberto Garigliano, for their encouragement, advice and guidance during the course of this thesis. Thanks are also due to Marco Mondadori for his invaluable comments and suggestions. I bave also benefitted from conversations with Ian Gent, Rajev Gore, Angus Macintyre, Paolo Mancosu, Bill McColl, Claudio Pizzi, Alasdair Urquhart, Lincoln Wallen and Alex Wilkie.

I also wish to thank Dov Gabbay for providing me with a draft of a forthcoming book of his; Peter Aczel and Alan Bundy for giving me the opportunity to discuss my work in seminars given at Manchester and Edinburgh, and Michael Mainwaring for doing his best to make my English more acceptable.

I am grateful to my parents and to all my friends, especially to Gabriella D'Agostino for her usual impassioned support.

This research has been supported by generous grants awarded by the Italian Ministry of Education, the City Council of Palermo ('Borsa di studio Ninni Cassarà') and CNR-NATO.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Traditionally, proof theory has been concerned with formal representations of the notion of proof as it occurs in mathematics or other intellectual activities, regardless of the efficiency of snch representations. The rapid development of computer science has bronght abont a dramatic cbange of attitude. Efficiency has become a primary concern and this fact has given rise to a whole new area of research in which the 'old' questions about provability, completeness, decidability, etc. have been replaced by new ones, with considerations of complexity playing a major role. A formal representation which is complete in principle may turn out to be 'practically incomplete' because in some cases it requires unrealistic resources (in terms of computer time and memory). As a result of this change of perspective, formal representations which are eqnivalent witb respect to the consequence relation that they represent, can be seen to be essentially separated as to their practical scope.

Nowhere has this shift of emphasis been more apparent than in the field of propositional logic. Here the neglected star is back in the limelight. Open questions of theoretical computer science like $\mathcal{P} =? \mathcal{NP}$ and $\mathcal{NP} =? co\text{-}\mathcal{NP}$ have revitalized a subject which seemed to be 'saturated' and to deserve only a brief mention as a stepping stone to the 'real thiug'. But the interest in propositional logic is not restricted to these fundamental open questions. There is a plethora of problems which are perhaps less fundamental but of practical and theoretical importance. Many of them arise when we consider logic as a *tool*. This was certainly the attitude of the pioneers of formal logic, from Aristotle to Leibniz. The latter dreamt of a *calculus ratiocinator* of which mathematical logic would be the heart, and explicitly related the

importance of his project to its potential 'usefulness':

> For if praise is given to the men who have determined the number of regular solids — which is of no use, except insofar as it is pleasant to contemplate — and if it is thought to be an exercise worthy of a mathematical genius to have brought to light the more elegant property of a conchoid or cissoid, or some other figure which rarely has any use, how much better will it be to bring under mathematical laws human reasoning, which is the most excellent and useful thing we have[1].

Later the role of logic in the foundations of mathematics took over and, while this opened up an entire world of conceptual and technical subtleties, it also brought about the neglect of many interesting questions[2]. Among such questions are all those related to the direct *use* of logic to solve problems about our 'world' (or our database), that is as a partial realization of Leibniz's dream. In the second half of this century, the revival of Leibniz's program (known as 'automated deduction') as well as the significant role played by a variety of logical methods in both theoretical and applied computer science, has resulted in a greater awareness of the computational aspects of logical systems and a closer (or at least a fresh) attention to their proof-theoretical representations. As Gabbay has stressed [Gab90a, Gab90b] logical systems 'which may be conceptually far apart (in their philophical motivation and mathematical definitions), when it comes to automated techniques and proof-theoretical presentation turn out to be brother and sister'[Gab90a, Section 1.1]. On the other hand, the same logical system (intended as a set-theoretical definition of a consequence relation) usually admits of a wide variety of proof-theoretical representations which may reveal or hide its similarities with other logical systems[3]. The consideration of these phenomena has prompted Gabbay to put forward the view that

[1] [Par66, page 166].

[2] In his survey paper [Urq90a] Alasdair Urquhart mentions the simplification of Boolean formulae. 'Here the general problem takes the form: how many logical gates do we need to represent a given Boolean function? This is surely as simple and central a logical problem that one could hope to find; yet, in spite of Quine's early contributions, the whole area has been simply abandoned by most logicians, and is apparently thought to be fit only for engineers.' He also remarks that 'our lack of understanding of the simplification problem retards our progress in the area of the complexity of proofs.'

[3] As an example of this approach, Gabbay observes that 'it is very easy to move from

> [...] a logical system **L** is not jnst the traditional consequence relation $\vdash$, but a pair $(\vdash, \mathbf{S}_{\vdash})$, where $\vdash$ is a mathematically defined consequence relation (i.e. a set of pairs (Δ, Q) such that $\Delta \vdash Q$) and $\mathbf{S}_{\vdash}$ is an algorithmic system for generating all those pairs. Thns, according to this definition, classical logic $\vdash$ perceived as a set of tautologies together with a Gentzen system $\mathbf{S}_{\vdash}$ is not the same as classical logic together with the two-valued truth-table decision procedure $\mathbf{T}_{\vdash}$ for it. In our conceptual framework, $(\vdash, \mathbf{S}_{\vdash})$ is *not the same logic* as $(\vdash, \mathbf{T}_{\vdash})$[4].

Gabbay's proposal seems even more suggestive when considerations of computational complexity enter the picture. Different proof-theoretical algorithms for generating the same consequence relation may have different complexities. Even more interesting: algorithmic representations which appear 'close' to each other from the proof-theoretical point of view may show dramatic differences as far as their efficiency is concerned. A significant part of this work will be devoted to illustrate this somewhat surprising phenomenon.

In the tradition which considers formal logic as an *organon* of thought[5] a central role has been played by the 'method of analysis', which amounts to what today, in the computer science circles, is called a 'bottom-up' or 'goal-oriented' procednre. Though the method was largely used in the math-

---

the truth-table presentation of classical logic to a truth-table system for Lukasiewicz's n-valued logic. It is not so easy to move to an algorithmic system for intuitionistic logic. In comparison, for a Gentzen system presentation, exactly the opposite is true. Intuitionistic and classical logic are neighbours, while Lukasiewicz's logics seem completeley different.'[Gab90a, Section 1.1].

[4][Gab90a, Section 1.1].

[5]Of conrse, as a matter of historical fact, logicians do not need to be full-time representatives of a single tradition. Leibniz is perhaps the best example: while he emphasized the practical utility of his dreamt-of calculus of reason to solve verbal disputes, he also claimed that the whole of mathematics conld be reduced to logical principles, indeed to the sole principle of identity; this made him a precursor of the logicist tradition in the philosophy of mathematics. Gentzen, who certainly belonged to the 'fonndational' tradition, showed some concern for the practical aspects of his own work. Among the advantages of his calculus of natural deduction he mentioned that 'in most cases the derivations for true formulae are *shorter* [...] than their counterparts in logistic calcnli'[Gen35, page 80]. A brilliant representative of both traditions is Beth: he was involved in the practical as well as the fonndational aspects of logic. His emphasis on the advantages of the tableau method as a tool for drawing inferences is pervasive and, as we will see, even excessive.

ematical practice of the ancient Greeks, its fullest description can be found in Pappus (3rd century A.D.), who writes:

> Now analysis is a method of taking that which is sought as though it were admitted and passing from it through its consequences in order, to something which is admitted as a result of synthesis; for in analysis we suppose that which is sought be already done, and we inquire what it is from which this comes about, and again what is the antecedent cause of the latter, and so on until, by retracing our steps, we light upon something already known or ranking as a first principle; and such a method we call analysis, as being a solution backwards[6].

This is the so-called *directional* sense of analysis. The idea of an 'analytic method', however, is often associated with another sense of 'analysis' which is related to the 'purity' of the concepts employed to obtain a result and, in the framework of proof-theory, to the *subformula principle* of proofs. In Gentzen's words: 'No concepts enter into the proof other than those contained in its final result, and their use was therefore essential to the achievement of that result'[Geu35, p. 69] so that 'the final result is, as it were, gradually built up from its constituent elements [Gen35, p.88]. Both meanings of analysis have been represented, in the last fifty years, by the notion of a *cut-free* proof in Gentzen's sense: not only cut-free proofs employ no concepts outside those contained in the statement of the theorem, but they can also be discovered by means of simple 'backwards' procedures like that described by Pappus. Therefore Gentzen-style cut-free methods (which include their semantic-flavoured descendant, known as 'the tableau method') were among the first used in automated deduction[7] and are today enjoying a deserved revival[8] which is threatening the unchallenged supremacy that resolution and its refinements[9] have had for over two decades in the area of

[6][Tho41] pp. 596–599.

[7]See, for example, [Bet58], [Wan60], [Kan63]. Cfr. also [Dav83] and [MMO83].

[8][Fit90] and [Gal86] are recent texthooks on 'logic for computer science' which are entirely based on the tableau method and on cut-free Gentzen systems respectively. [Wal90] is a book which combines the technical power of Bibel's connection method [Bib82] with the conceptual clarity of the tableau representation of classical as well as non-classical logic. [OS88] is a recent example of a complete theorem-prover based on the tableau method.

[9]See [CL73] and [Sti86] for an overview.

automated deduction and logic programming.

There are indeed several reasons which may lead us to prefer cut-free Gentzen methods to resolution as a basis for automated deduction as well as for other applications, like program verification. First, unlike resolution, these methods do not require reduction in any normal form, which fact allows them to exploit the full power of first order logic[10]. Second, the derivations obtained constitute an acceptable compromise between 'machine-oriented' *tests for validity*, like resolution derivations, and 'human-oriented'*proofs*, like those constructed in the framework of natural deduction calculi; this makes them more suitable for 'interactive' use. Third, they admit of natural extensions to a wide variety of non-classical logics[11] which appear less contrived than their resolution-based counterparts[12].

On the other hand, there is *one* reason, which usually leads us to prefer some form of resolution: efficiency. As will be shown in Chapter 4 cut-free methods are hopelessly inefficient and lead to combinatorial explosion in fairly simple examples which are easily solved not only by resolution but even by the old, despised truth-tables. What is the ultimate cause of this inefficiency? Can it be remedied? The main contribution of our work is a clarification of this issue and a positive proposal motivated thereby. The clarification can be expressed as a paradox: the ultimate cause of the inefficiency of cut-free methods is *exactly* what has traditionally made them so useful both from the theoretical and practical point of view, namely their being *cut-free*. But, as Smullyan once remarked, 'The real importance of cut-free proofs is not the elimination of cuts per se, but rather that such proofs obey the subformula principle.'[13]. Apart from Smullyan's short paper cited above, however, the proof-theory of *analytic cut* systems, i.e. systems which *allow* cuts but limit their application so that the resulting proofs still obey the subformula principle, has not been adequately studied, and the same is

[10]On this point see [Cel87].

[11]For Intuitionistic and Modal logics, see [Fit83]; see also [Wal90] for a tableau-oriented extension of Bibel's connection method to intuitionistic and modal logics; for relevance logics see [TMM88]; for many-valued logics see [Car87]; for linear logic see [Gir87b] and [Avr88].

[12]See [FE89] and [Fit87]. For a review of these and other resolution-based methods for non-classical logics, see [Wal90].

[13][Smu68b], p. 560.

true of their relative complexity[14]. In this work we shall take some steps in a forward direction. Moreover, while considerations of efficiency are nsually strictly technical and unconnected with more conceptual issues, for example semantical issues, we shall make a special and, we hope, successful effort to bring out a clear connection between computational efficiency on the one hand, and a close correspondence with the underlying (classical) semantics on the other.

We shall base our study on a particular system, the system **KE**, recently proposed by Mondadori [Mon88a, Mon88b] as an alternative to the tableau method for classical logic. As will be argued, **KE** is in some sense ideal for our purposes in that it constitutes a refutation system which, though being proof-theoretically 'close' to the tahleau method, is *essentially more efficient.* Moreover, it is especially suited to the purpose of illustrating the connection between efficiency, analytic cut and classical semantics, which is the object of our study.

In Chapter 2, we examine what we call the 'redundancy of cut-free proofs' as a phenomenon which depends on *the form* of the cut-free rules themselves and not *on the way in which they are applied:* this phenomenon affects cut-free systems intended as non-deterministic algorithms for generating proofs and therefore affects *every* proof generated by them. We relate this intrinsic redundancy to the fact that the cut-free rules do not reflect, in some sense, the (classical bivalent) structure of their intended semantics. In Chapter 3 we develop an alternative approach based on Mondadori's system **KE** which is not cut-free yet obeys the subformula principle (and also a stronger normal form principle), so preserving the most important property of cut-free systems. The system **KE** is a refutation system which, like the tableau method, generates trees of (signed or unsigned) formulae by means of tree expansion rules. Its distinguishing feature is that its *only* branching rule is the rule PB (Principle of Bivalence) which is closely related to the cut rule of Gentzen systems. So, in spite of all the similarities, **KE** hinges upon a rule whose absence is typical of the tableau method (and of all cut-free systems). An important related property of **KE** is the *analytic cut property*: cuts are not eliminable but can be restricted to *analytic* ones, involving only subformulae of the theorem. We also show that the set of formulae to be considered as

[14] For a recent exception see [Urq90b].

cut formulae can be further restricted as a result of a strong normal form theorem.

In Chapter 4, after briefly reviewing the most important results on the relative complexity of propositional calculi, we examine how the use of analytic cuts affects the complexity of proofs. We show that **KE** (and, in fact, *any* analytic cut system) is essentially more efficient than the tableau method (or *any* cut-free system). These results should be read in parallel with the semantic-oriented analysis carried out in Chapter 2: **KE** establishes a close connection with classical (bivalent) semantics which the tableau method fails to do, and this very fact has important consequences from the computational point of view.

Like the tableau method, **KE** can be adapted to a variety of non-classical logics including intuitionistic and modal logics. All these non-classical versions can be obtained trivially, given the corresponding tableau-based systems, as explained in Section 3.9. In Chapter 5 we shall examine a new case: Belnap's four-valued logic. This is a form of relevant logic which has also an interesting motivation from the point of view of computer science, since it is intended to formalize the notion of deduction from inconsistent databases. We formulate a tableau-based and a **KE**-based system both of which, as in the case of classical two-valued logic, generate simple *binary* trees. These systems are based on a reformulation of Belnap's four-valued semantics which 'mimics' the bivalent structure of classical semantics and, to the best of our knowledge, is new to the literature. Although Belnap's logic is a *restriction* of classical logic (intended as a consequence relation), our systems can be seen as *extensions* of the corresponding classical systems in that they allow us to characterize simultaneously the classical two-valued and Belnap's four-valued consequence relation using the same formal machinery.

Finally, in Chapter 6, we generalize on the ideas contained in the previous chapters. We show that, as far as the complexity of 'conventional' (analytic and non-analytic) proof systems is concerned, the cut rule is *all* that matters and the form of the logical rules (under certain conditions) does not play any significant role. This chapter finalizes the process started in Chapter 2: what in Gentzen-type systems is eliminable (and indeed its eliminability provides a major motivation for the form of the rules) becomes, in 'cut systems', the *only* essential feature.

Chapter 2

The redundancy of cut-free proofs

2.1 Invertible sequent calculi

Gentzen introduced the sequent calculi **LK** and **LJ** as well as the natural deduction calculi **NK** and **NJ** in his famous 1935 paper [Gen35]. Apparently he considered the sequent calculi as technically more convenient for metalogical investigation[1]. In particular he thought that they were 'especially suited to the purpose' of proving the *Hauptsatz* and that their form was 'largely determined by considerations connected with [this purpose]'[2]. He called these calculi 'logistic' because, unlike the natural deduction calculi, they do not involve the introduction and subsequent discharge of assumptions, but deal with formulae which are 'true *in themselves*, i.e. whose truth is no longer *conditional* on the truth of certain assumption formulae'[3]. Such 'unconditional' formulae are *sequents*, i.e. expressions of the form

$$A_1, \ldots, A_n \vdash B_1, \ldots, B_n \tag{2.1}$$

(where the A_i's and the B_i's are formulae) with the same informal meaning as the formula

$$A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m.$$

[1] [Gen35], p. 69.
[2] [Gen35, p. 89].
[3] [Gen35], p. 82.

The sequence to the left of the turnstyle is called 'the antecedent' and the sequence to the right is called 'the succedent'. In the case of intuitionistic logic the succedent may contain at most one formula. In this chapter we shall consider the classical system and focus on the *propositional* rules.

Although Gentzen considered the antecedent and the succedent as *sequences*, it is often more convenient to use *sets*, which eliminates the need for 'structural' rules to deal with permutations and repetitions of formulae[4]. Table 2.1 shows the rules of Gentzen's **LK** once sequences have been replaced by sets (we use Γ, Δ, etc. for sets of formulae and write Γ, A as an abbreviation of $\Gamma \cup \{A\}$). A proof of a sequent $\Gamma \vdash \Delta$ consists of a tree of sequents built up in accordance with the rules and on which all the leaves are axioms. Gentzen's celebrated *Hauptsatz* says that the cut rule can be eliminated from proofs. This obviously implies that the cut-free fragment is complete. Furthermore, one can discard the last structural rule left — the thinning rule — and do without structural rules altogether without affecting completeness, provided that the axioms are allowed to have the more general form

$$\Gamma, A \vdash \Delta, A.$$

This well-known variant corresponds to Kleene's system **G4** [Kle67, chapter VI].

Gentzen's rules have become a paradigm both in proof-theory and in its applications. This is not without reason. First, like the natural deduction calculi, they provide a precise analysis of the logical operators by specifying how each operator can be introduced in the antecedent or in the succedent of a sequent[5]. Second, their form ensures the validity of the *Hauptsatz*: each proof can be transformed into one which is cut-free, and cut-free proofs

[4]This reformulation is adequate for classical and intuitionistic logic, but not if one wants to use sequents for some other logic, like relevance or linear logic, in which the number of occurrences of formulae counts. For such logics the antecedent and the succedent are usually represented as *multisets*; see [TMM88] and [Avr88].

[5]Whereas in the natural deduction calculi there are, for each operator, an introduction and an elimination rule, in the sequent calculi there are only introduction rules and the eliminations take the form of introductions in the antecedent. Gentzen seemed to consider the difference between the two formulations as a purely technical aspect. See [Sun83]. He also suggested that the rules of the natural deduction calculus could be seen as *definitions* of the operators themselves. In fact he argued that the introduction rules alone are sufficient for this purpose and that the elimination rules are 'no more, in the final analysis, than

Axioms

$$A \vdash A$$

Structural rules

$$\frac{\Gamma \vdash \Delta}{\Gamma, \Theta \vdash \Delta, \Lambda}\ [\text{Thinning}] \qquad \frac{\Gamma, A \vdash \Delta \quad \Gamma \vdash \Delta, A}{\Gamma \vdash \Delta}\ [\text{Cut}]$$

Operational rules

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\ [\text{I-}\vee \textit{left}] \qquad \frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}\ [\text{I-}\wedge \textit{right}]$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ [\text{I-}\wedge \textit{left}] \qquad \frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B}\ [\text{I-}\vee \textit{right}]$$

$$\frac{\Gamma \vdash \Delta, A \quad \Gamma, B \vdash \Delta}{\Gamma, A \to B \vdash \Delta}\ [\text{I-}\to \textit{left}] \qquad \frac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta, A \to B}\ [\text{I-}\to \textit{right}]$$

$$\frac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta}\ [\text{I-}\neg \textit{left}] \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\ [\text{I-}\neg \textit{right}]$$

Table 2.1: Sequent rules for classical propositional logic.

enjoy the *subformula property*: every sequent in the proof tree contains only subformulae of the formulae in the sequent to be proved. From a conceptual viewpoint this property represents the notion of a purely analytic or 'direct' argument[6]: 'no concepts enter into the proof other than those contained in its final result, and their use was therefore essential to the achievement of that result'[Gen35, p. 69]. Third, the rules of a cut-free system, like Kleene's **G4**, seem to be particularly suited to a 'goal-oriented' search for proofs: instead of going from the axioms to the endsequent, one can start from the endsequent and use the rules in the reverse direction, going from the conclusion to suitable premises from which it can be derived. This method, which is clearly reminiscent of Pappus' 'theoretical analysis'[7], works only in virtue of an important property of the rules of **G4** described in Lemma 6 of [Kle67], namely their *invertibility*.

Definition 2.1.1 A rule is *invertible* if the provability of the sequent below the line in each application of the rule implies the provability of all the sequents above the line.

As was early recognized by the pioneers of Automated Deduction[8], if a logical calculus has to be employed for this kind of 'bottom-up' proof-search it is essential that its rules be invertible: this allows us to stop as soon as we reach a sequent that we can recognize as unprovable (for instance one containing only atomic formulae and in which the antecedent and the succedent are disjoint) and conclude that the initial sequent is also unprovable. We should notice that the absence of the thinning rule is crucial in this context because it is easy to see that *the thinning rule is not invertible*: the provability of its conclusion does not imply, in general, the provability of the premise.

consequences of these definitions'[Gen35, p. 80]. He also observed that this 'harmony' is exhibited by the intuitionistic calculus but breaks down in the classical case. For a thorough discussion of this subtle meaning-theoretical issue the reader is referred to the writings of Michael Dummett and Dag Prawitz, in particular [Dum78] and [Pra78].

[6]On this point see [Sta77].

[7]See the Introduction above.

[8]See for instance [Mat62].

2.2 From sequent proofs to tableau refutations

As far as classical logic is concerned, a system like G4 admits of an interesting semantic interpretation.

Let us say that a sequent $\Gamma \vdash \Delta$ is *valid* if every situation (i.e. a boolean valuation) which makes all the formulae in Γ true, also makes true at least one formula in Δ. Otherwise if some situation makes all the formulae in Γ true and all the formulae in Δ false, we say that the sequent is *falsifiable* and that the situation provides a *countermodel* to the sequent.

According to this semantic viewpoint we prove that a sequent is valid by ruling out all possible falsifying situations. So a sequent $\Gamma \vdash \Delta$ represents a *valuation problem*: find a boolean valuation which falsifies it. The *soundness* of the rules ensures that a valuation which falsifies the conclusion must also falsify at least one of the premises. Thns, if applying the rules backwards we reach an axiom in every branch, we are allowed to conclude that no falsifying valuation is possible (since no valuation can falsify an axiom) and that the endsequent is therefore valid. On the other hand, the *invertibility* of the rules allows us to stop as soon as we reach a falsifiahle sequent and claim that any falsifying valuation provides a countermodel to the endsequent. Again, if the thinning rule were allowed, we would not be be able, in general, to re-transmit falsifiability back to the endsequent. So, if employed in bottom-up search procedures, the thinning rule may result in the loss of crucial semantic information.

Beth, Hintikka, Schütte and Kanger (and maybe more) independently showed, in the '50s, how this semantic interpretation provided a strikingly simple and informative proof of Gödel's completeness theorem[9]. Their results suggested that the semantic interpretation could be presented as a proof method on its own, in which sequents do not appear and complex formulae are progressively 'analysed' into their successive components. This approach was later developed and perfected by Smullyan with his method of 'analytic tableaux' [Smu68a].

The construction of an analytic tableau for a sequent $\Gamma \vdash \Delta$ closely corresponds to the systematic search for a countermodel outlined above ex-

[9] [Bet55], [Hin55], [Sch56], [Kan57].

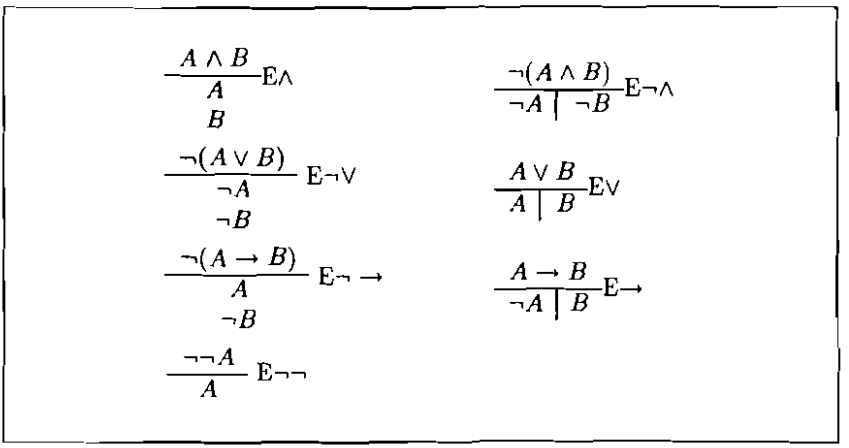


Table 2.2: Propositional tableau rules for unsigned formulae.

cept that Smullyan's presentation uses trees of formulae instead of trees of sequents. The correspondence between the tableau formulation and the sequent formulation is illustrated in detail in Smullyan's book [Smu68a] to which we refer the reader[10]. The propositional tableau rules (for unsigned formulae) are listed in Table 2.2.

2.3 Cut-free proofs and bivalence

This evolution from the original **LK** system to Smullyan's system of 'analytic tableaux' took over thirty years but did not change Gentzen's formulation significantly. However, Gentzen's rules are not the only possible way of analysing the classical operators and not necessarily the best for all purposes. The form of Gentzen's rules was influenced by considerations which were partly philosophical, partly technical. In the first place he wanted to set up a formal system which 'comes as close as possible to actual reasoning'[11]. In this context he introduced the natural deduction calculi in which the inferences are analysed essentially in a constructive way and classical logic is

[10] See also [Smu68c] and [Sun83]

[11] [Gen35], p. 68.

obtained by adding the law of excluded middle in a purely external manner. Then he recognized that the special position occupied by this law would have prevented him from proving the *Hauptsatz* in the case of classical logic. So he introduced the sequent calculi as a technical device in order to enunciate and prove the *Hauptsatz* in a convenient form[12] both for intuitionistic and classical logic. These calculi still have a strong deduction-theoretic flavour and Gentzen did not show any sign of considering the relationship between the classical calculus and the semantic notion of entailment which, at the time, was considered as highly suspicious.

The approach developed in the '50's by Beth and Hintikka was mainly intended to bridge the gap between classical semantics and the theory of formal derivability. In his 1955 paper, where he introduced the method of semantic tableaux, Beth thought he had reached a formal method which was 'in complete harmony with the standpoint of [classical bivalent] semantics' [Bet55, p. 317]. He also claimed that this method would allow for the 'purely mechanical' construction of proofs which were at the same time 'remarkably concise' and could even be 'proved to be, in a sense, the shortest ones which are possible' [Bet55, p. 323]. On the other hand Hintikka, who independently and simultaneously developed what amounts to the same approach, hoped that in this way he would 'obtain a semantical theory of quantification which satisfies the highest standard of constructiveness' [Hin55, p. 21]. Both authors explicitly stressed the correspondence between their own rules and those put forward by Gentzen[13]. So two apparently incompatible aims seemed to be achieved in the formal framework of Gentzen-type rules.

The solution to this puzzle is that the so-called semantic interpretation of Gentzen's rules does not establish those 'close connections between [classical] semantics and derivability theory' that Beth tried to point out. In fact if one takes the cut-free rules as a formal representation of *classical* logic, the situation seems odd. The rule which the *Hauptsatz* shows to be eliminable is the cut rule:

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma \vdash \Delta, A}{\Gamma \vdash \Delta}\,[\mathrm{CUT}]$$

If we read this rule upside-down, following the same semantic interpretation

[12] [Gen35], p. 69.

[13] [Bet55], p. 318 and p. 323; [Hin55], p.47.

that we adopt for the operational rules, then what the cut rule says is:

> *In all circumstances and for all propositions A, either A is true or A is false.*

But this is the Principle of Bivalence, namely the *basic* principle of classical logic. By contrast, none of the rules of the cut-free fragment implies bivalence (as is shown by the three-valued semantics for this fragment[14]). The elimination of cuts from proofs is, so to speak, the elimination of bivalence from the underlying semantics[15].

We have, then, a rather ambiguous situation: on the one hand we have a *complete* set of rules which are usually taken as a convenient analytic representation of classical logic; on the other hand, these rules do not assign to the basic feature of classical semantics — the Principle of Bivalence — any special role in the analysis of classical inferences[16]. We can ask ourselves two questions: is the elimination of bivalence (cut) necessary? Is it harmless?

As Smullyan once remarked: 'The real importance of cut-free proofs is not the elimination of cuts per se, but rather that such proofs obey the subformula principle.'[17] So, our two questions can be reformulated as follows: (1) Can we think of an analysis of classical inferences which gives the Principle of Bivalence the prominent role that it should have in a formal representation of *classical* logic? (2) Would such an analysis accomplish a more concise and efficient representation of classical proofs which preserves the most important property of the cut-free analysis (the subformula principle)?

The two questions are independent, but it is only to be expected that a positive answer to the second will be a by-product of a good answer to the first.

In the rest of this chapter we shall argue that the Principle of Bivalence (i.e. some form of cut) should indeed play a role in classical analytic

[14]See [Gir87a], chapter 3; see also [Sch77].

[15]Of course, the fact that the cut-free rules, in the semantic interpretation, are sound from the standpoint of classical semantics does not necessarely mean that they are in 'complete harmony' with this standpoint.

[16]Hintikka was fully aware that in his approach he was depriving the principle of bivalence of any role. In fact he did it *intentionally* in order to pursue his quasi-constructivist program. Cfr. [Hin55], chapter III, pp. 24–26

[17][Smu68b, p.560]. A problem: is there any application of cut-elimination which requires the proofs to be cut-*free*, not just to satisfy the subformula property?

deduction[18]; that the reintroduction of bivalence in the analysis not only does not affect the subformula principle, but *also allows for much shorter proofs*, because it eliminates a kind of redundancy which is inhereut to the cut-free analysis. In the next section we shall discuss a typical example of this redundancy.

2.4 The redundancy of cut-free proofs

Gentzen said that the essential property of a cut-free proof is that 'it is not roundabout' [Gen35, p.69]. By this he meant that: 'the final result is, as it were, gradually built up from its constituent elements. The proof represented by the derivation is not roundabout in that it contains only concepts which recur in the final result' [Gen35, p.88]. However there is a sense in which cut-free proofs *are* roundabout.

Let us consider, as a simple example, a cut-free proof of the sequent:

$$A \vee B, A \vee \neg B, \neg A \vee C, \neg A \vee \neg C \vdash \emptyset$$

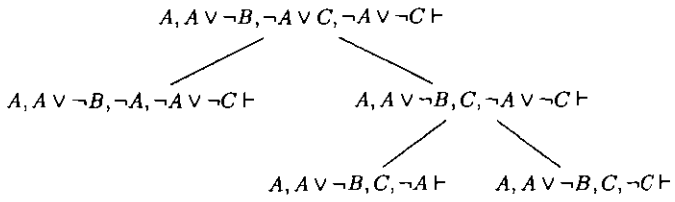
expressing the fact that the antecedent is inconsistent.

A minimal proof is as follows (we write the proof npside-down according to the interpretation of the sequent rules as redndction rules in the search for a counterexample; by $\Gamma \vdash$, we mean $\Gamma \vdash \emptyset$ and consider as axioms all the sequents of the form $\Gamma, A, \neg A \vdash$)):

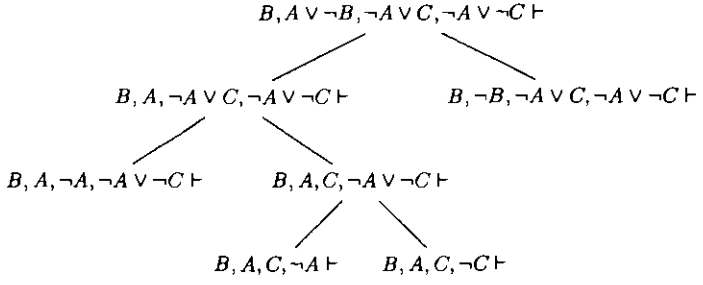
$$\begin{array}{ccc} & A \vee B, A \vee \neg B, \neg A \vee C, \neg A \vee \neg C \vdash & \\ \swarrow & & \searrow \\ A, A \vee \neg B, \neg A \vee C, \neg A \vee \neg C \vdash & & B, A \vee \neg B, \neg A \vee C, \neg A \vee \neg C \vdash \\ \mathcal{T}_1 & & \mathcal{T}_2 \end{array}$$

Where $\mathcal{T}_1 =$

[18] By this we do not mean that some form of cut should be 'thrown into' the cut-free system, but that some of the rules for the logical operators should be modified to allow the cut rule to play a role in the analysis.



and $\mathcal{T}_2 =$



Such a proof is, in some sense, redundant when it is interpreted as a (failed) systematic search for a countermodel to the endsequent (i.e. a model of the antecedent): the subtree $\mathcal{T}_1$ encodes the information that *there are no countermodels which make A true*, but this information cannot be used in other parts of the tree and, in fact, $\mathcal{T}_2$ still tries (in its left subtree) to construct a countermodel which makes A true, only to show again that such a countermodel is impossible.

Notice that (i) the proof in the example is minimal; (ii) the redundancy does not depend on our representation of the proof as *a tree*: the reader can easily check that all the sequents which label the nodes are different from each other and, as a result, the proof would have the same size if represented as a sequence or as a directed acyclic graph. The only way to obtain a non-redundant proof in the form of a sequence or a directed acyclic graph of sequents would be by using the thinning rule :

(1)	$A, C, \neg A \vdash$	Axiom
(2)	$A, C, \neg C \vdash$	Axiom
(3)	$A, C, \neg A \vee \neg C \vdash$	From (1) and (2)
(4)	$A, \neg A, \neg A \vee \neg C \vdash$	Axiom
(5)	$A, \neg A \vee C, \neg A \vee \neg C \vdash$	From (4) and (3)
(6)	$B, A, \neg A \vee C, \neg A \vee \neg C \vdash$	From (5) by thinning
(7)	$B, \neg B, \neg A \vee C, \neg A \vee \neg C \vdash$	Axiom
(8)	$B, A \vee \neg B, \neg A \vee C, \neg A \vee \neg C \vdash$	From (6) and (7)
(9)	$A, A \vee \neg B, \neg A \vee C, \neg A \vee \neg C \vdash$	From (5) by thinning
(10)	$A \vee B, A \vee \neg B, \neg A \vee C, \neg A \vee \neg C \vdash$	From (8) and (9)

In this case the proof obtained by employing thinning is not much shorter because of the simplicity of the example considered. Yet, it illustrates the use of thinning in *direct* proofs in order to eliminate redundancies. However, for the reasons discussed in section 2.1 the thinning rule is *not* suitable for bottom-up proof search.

The situation is perhaps clearer if we represent the proof in the form of a closed tableau *à la* Smullyan (see Fig. 2.1). It is easy to see that such a tableau shows the same redundancy as the sequent proof given before.

This intrinsic redundancy of the cut-free analysis is responsible in many cases for explosive growth in the size of the search tree. Moreover, it is *essential*: it does not depend on any particular proof-search procedure (it affects *minimal* proofs) but only on the use of the cut-free rules. In the rest of this chapter this point will be examined in detail.

2.5 The culprit

Can we think of a more economical way of organizing our search for a countermodel? of avoiding the basic redundancy of the cut-free analysis? We must first identify the culprit. Our example contains a typical pattern of cut-free refutations which is represented in figure 2.2. Here the subtree $\mathcal{T}_1$ searches for possibile countermodels which make A true. If the search is successful, the original sequent is not valid and the problem is solved. Otherwise there is *no* countermodel which makes A true (i.e. if we restrict ourselves to classical bivalent models, every countermodel, if any, must make A false). In both cases it is pointless, while building up $\mathcal{T}_2$, to try to construct (as we do

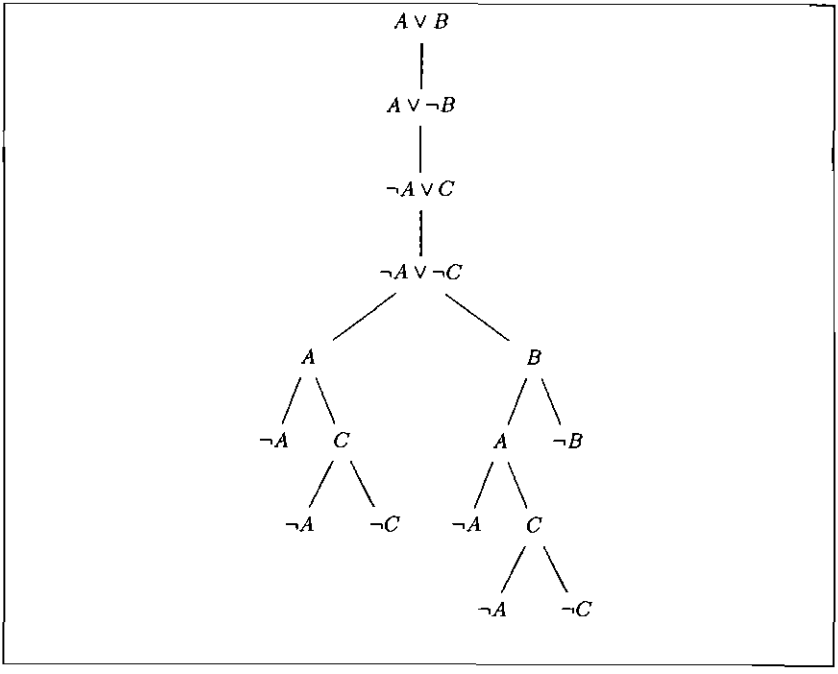


Figure 2.1: A closed tableau for $\{A \vee B, A \vee \neg B, \neg A \vee C, \neg A \vee \neg C\}$

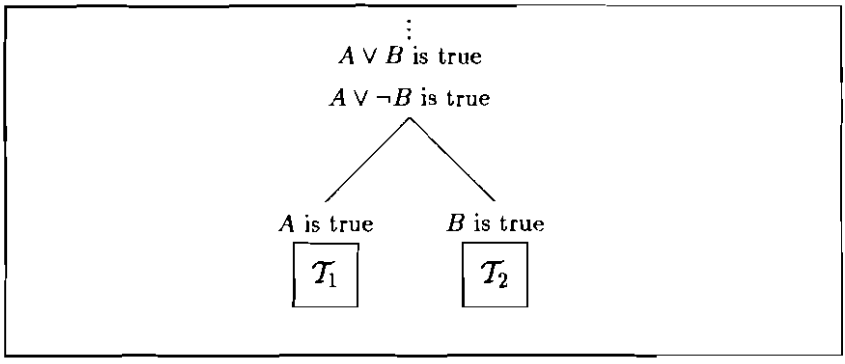


Figure 2.2: A typical pattern.

if our search is governed by Gentzen's rules) countermodels which make A true, because this kind of couutermodel is already sought in $\mathcal{T}_1$.

In general, we may have to reiterate this redundant pattern an arbitrary number of times, depending on the composition of our input set of formulae. For instance, if a branch contains n disjunctious $A \vee B_1, \ldots, A \vee B_n$ which are all to be analysed in order to obtain a closed subtableau, it is often the case that the *shortest* tablean has to contain higly redundant configurations like the one shown iu figure 2.3: where the subtree $\mathcal{T}^*$ has to be repeated n times. Each copy of $\mathcal{T}^*$ may, in turn, contain a similar pattern. It is not difficult to see how this may rapidly lead to a combinatorial explosion which is by no means related to auy 'intrinsic difficulty' of the problem considered but only to the redundant behaviour of the cut-free rules.

2.6 Searching for a countermodel

The example discussed in sectious 2.4 and 2.5 suggests that, in some sense, analytic tableaux coustructed according to the cut-free tradition are not well-suited to the nature of the problem they are intended to solve. Iu this section we shall render this claim more precise.

Let us call a *partial valuation* of Γ, where Γ is a set of formulae, any partial

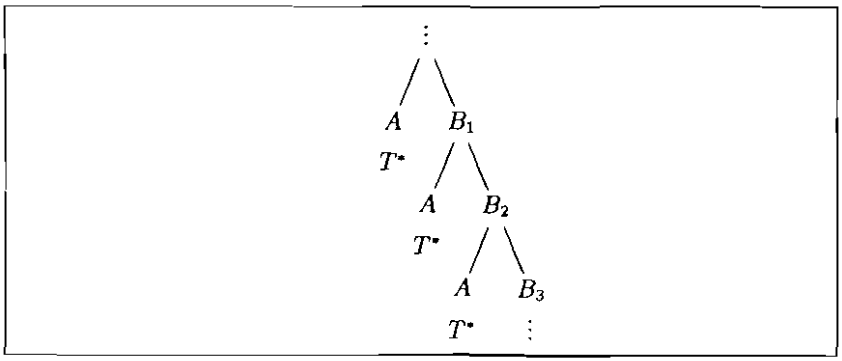


Figure 2.3: Redundancy of tableau refutations.

function $v\colon \Gamma \mapsto \{1,0\}$ where 1 and 0 stand as usual for the truth-values *true* and *false* respectively. It is convenient for our purposes to represent partial functions as total functions with values in $\{1,0,*\}$ where $*$ stands for the 'undefined' value. By a *total* valuation of Γ we shall mean a partial valuation which for no element of Γ yields the value $*$. For every element A of Γ we say that A *is true under* v if $v(A) = 1$, *false under* v if $v(A) = 0$ and *undefined under* v if $v(A) = *$. We say that a *sequent* $\Gamma \vdash \Delta$ *is true under* v if $v(A) = 0$ for some $A \in \Gamma$ or $v(A) = 1$ for some $A \in \Delta$. We say that it is *false under* v if $v(A) = 1$ for all $A \in \Gamma$ and $v(A) = 0$ for all $A \in \Delta$.

Let F denote the set of all formulae of propositional logic. A *boolean valuation*, defined as usual, is regarded from this point of view as a special case of a partial valuation of F, namely one which is total and is faithful to the usual truth-table rules.

In the no-countermodel approach to validity we start from a sequent $\Gamma \vdash \Delta$ intended as a valuation problem and try to find a countermodel to it — at the propositional level a boolean valuation which falsifies it. Here the *direction* of the procedure is characteristic: one moves from complex formulae to their components in a typical 'analytic' way. In this context it is sufficient to construct some partial valuation which satisfies certain closure conditions. Such partial valuations have been extensively studied in the literature and are known under different names and shapes. They constitute the basic idea

underlying the simple completeness proofs discovered in the '50's which have been mentioned in the first two sections of this chapter. Following Prawitz [Pra74] we shall call them 'semivaluations':

Definition 2.6.1 Let $\widehat{\Gamma}$ denote the closure of the set of formulae Γ under the subformula relation. A (hoolean) *semivaluation* of $\widehat{\Gamma}$ is a partial valuation v of $\widehat{\Gamma}$ which satisfies the following conditions, for all $A, B \in \widehat{\Gamma}$:

1. if $v(A \vee B) = 1$, then $v(A) = 1$ or $v(B) = 1$;
2. if $v(A \vee B) = 0$, then $v(A) = 0$ and $v(B) = 0$;
3. if $v(A \wedge B) = 1$, then $v(A) = 1$ and $v(B) = 1$;
4. if $v(A \wedge B) = 0$, then $v(A) = 0$ or $v(B) = 0$;
5. if $v(A \rightarrow B) = 1$, then $v(A) = 0$ or $v(B) = 1$;
6. if $v(A \rightarrow B) = 0$, then $v(A) = 1$ and $v(B) = 0$;
7. if $v(\neg A) = 1$, then $v(A) = 0$;
8. if $v(\neg A) = 0$, then $v(A) = 1$.

The property of semivaluations which justifies their use is that they can be readily extended to boolean valuations, i.e. models in the traditional sense:

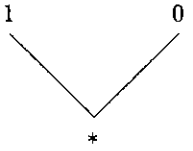
Lemma 2.6.1 *Every semivaluation of* $\widehat{\Gamma}$ *can be extended to a boolean valuation.*

Each stage of our attempt to construct a semivaluation which falsifies a given sequent $\Gamma \vdash \Delta$ can, therefore, be described as a partial valuation $v: \widehat{\Gamma} \cup \widehat{\Delta} \mapsto \{1, 0, *\}$. We start from the partial valuation which assigns 1 to all the formulae in Γ and 0 to all the formulae in Δ and try to refine it by extending step by step its domain of definition, taking care that the classical rules of truth are not infringed. If we eventually reach a partial valuation which is a semivaluation, we have successfully described a countermodel to the original sequent. Otherwise we have to ensure that no way of refining the initial partial valuation will ever lead to a semivaluation.

The search space, then, is a set of partial valuations which are naturally ordered by the approximation relationship (in Scott's sense [Sco70]):

$$v \sqsubseteq v' \text{ if and only if } v(A) \preceq v'(A) \text{ for all formulae } A$$

where $\preceq$ is the usual partial ordering over $\{1, 0, *\}$, namely



The set of all these partial valuations, together with the approximation relationship defined above, forms a complete semilattice. It can be convenient to transform this semilattice into a complete lattice by adding an 'overdefined' element $\top$. This 'fictitious' element of the lattice does not correspond to any real partial valuation and is used to provide a least upper bound for pairs of partial valuations which have no common refinement[19]. Hence we can regard the equation

$$v \sqcup v' = \top$$

as meaning intuitively that v and v' are *inconsistent*.

Having described the primitive structure of the search space, we are left with the problem of formulating efficient methods for exploring it. Constructing an analytic tableau is *one* method and, as we will see, certainly not the most efficient. This point is best seen by generalizing the basic idea underlying the tableau method. This is what will be done in the next two subsections.

2.6.1 Expansion systems

We assume a 0-order language defined as usual. We shall denote by X, Y, Z (possibly with subscripts) arbitrary *signed formulae* (s-formulae), i.e. expressions of the form $\mathrm{t}(A)$ or $\mathrm{f}(A)$ where A is a formula. The *conjugate* of an

[19]Namely partial valuations v and v' such that for some A in their common domain of definition $v(A) = 1$ and $v'(A) = 0$.

s-formula is the result of changing its sign (so $\mathrm{t}(A)$ is the conjugate of $\mathrm{f}(A)$ and viceversa). Sets of signed formulae will he denoted by S, U, V (possibly with subscripts). We shall use the upper case greek letters $\Gamma, \Delta, \ldots$ for sets of *unsigned* formulae. We shall often write S, X for $S \cup \{X\}$ and S, U for $S \cup U$. Given a formula A, the set of its *subformulae* is defined in the usual way. We shall call *subformulae of an s-formula* $\mathrm{s}(A)$ ($\mathrm{s} = \mathrm{t}, \mathrm{f}$) all the formulae of the form $\mathrm{t}(B)$ or $\mathrm{f}(B)$ where B is a subformula of A. For instance $\mathrm{t}(A), \mathrm{t}(B), \mathrm{f}(A), \mathrm{f}(B)$ will all be subformulae of $\mathrm{t}(A \vee B)$.

Definition 2.6.2 We say that an s-formula X is *satisfied* by a boolean valuation v if $X = \mathrm{t}(A)$ and $v(A) = 1$ or $X = \mathrm{f}(A)$ and $v(A) = 0$. A *set* S of s-formulae is *satisfiable* if there is a hoolean valuation v which satisfies all its elements.

A set of s-formulae S is *explicitly inconsistent* if S contains both $\mathrm{t}(A)$ and $\mathrm{f}(A)$ for some formula A. If S is not explicitly inconsistent we say that it is *surface-consistent.*

Sets of s-formulae correspond to the partial valuations of the previous section in the obvious way (we shall omit the adjective 'partial' from now on): given a surface-consistent set S of s-formulae its associated valuation is the valuation v_S defined as follows:

$$v_S(A) = \begin{cases} 1 & \text{if } \mathrm{t}(A) \in S \\ 0 & \text{if } \mathrm{f}(A) \in S \\ * & \text{otherwise} \end{cases}$$

(An explicitly inconsistent set of s-formulae is associated with the top element $\top$.) Conversely, given a partial valuation v its associated set of s-formulae will be the set S_v containing $\mathrm{t}(A)$ for every formula A such that $v(A) = 1$, and $\mathrm{f}(A)$ for every formula A such that $v(A) = 0$ (and nothing else).

So we can always regard the s-formulae $\mathrm{t}(A)$ and $\mathrm{f}(A)$ as 'meaning' $v(A) = 1$ and $v(A) = 0$ respectively.

The sets of s-formulae corresponding to semivaluations are known in the literature as *Hintikka sets.*

Definition 2.6.3 A set of s-formulae S is a (propositional) *Hintikka set* if it satisfies the following conditions (for every A, B):

1. For no variable P, $\mathrm{t}(P)$ and $\mathrm{f}(P)$ are both in S.

2. If $\mathrm{t}(\neg A) \in S$, then $\mathrm{f}(A) \in S$.
3. If $\mathrm{f}(\neg A) \in S$, then $\mathrm{t}(A) \in S$.
4. If $\mathrm{t}(A \vee B) \in S$, then $\mathrm{t}(A) \in S$ or $\mathrm{t}(B) \in S$.
5. If $\mathrm{f}(A \vee B) \in S$, then $\mathrm{f}(A) \in S$ and $\mathrm{f}(B) \in S$.
6. If $\mathrm{t}(A \wedge B) \in S$, then $\mathrm{t}(A) \in S$ and $\mathrm{t}(B) \in S$.
7. If $\mathrm{f}(A \wedge B) \in S$, then $\mathrm{f}(A) \in S$ or $\mathrm{f}(B) \in S$.
8. If $\mathrm{t}(A \rightarrow B) \in S$, then $\mathrm{f}(A) \in S$ or $\mathrm{t}(B) \in S$.
9. If $\mathrm{f}(A \rightarrow B) \in S$, then $\mathrm{t}(A) \in S$ and $\mathrm{f}(B) \in S$.

The sets of s-formulae corresponding, in a similar way, to a boolean valuation are often called *truth sets* or *saturated sets*. The translation of lemma 2.6.1 is known as the (propositional) Hintikka lemma.

Lemma 2.6.2 (Propositional Hintikka lemma) *Every propositional Hintikka set is satisfiable.*

In other words, every Hintikka set can be embedded in a truth set.

We shall now define the notion of *expansion system* which generalizes the tableau method.

Definition 2.6.4

1. An $n \times m$ *expansion rule* R is a relation between n-tuples of s-formulae and m-tuples of s-formulae, with $n \geq 0$ and $m \geq 1$. Expansion rules may be represented as follows:

$$\frac{\begin{array}{c}\chi_1 \\ \vdots \\ \chi_n\end{array}}{\upsilon_1 | \ldots | \upsilon_m}$$

where the χ_i's and the υ_i's are *schemes* of s-formulae. We say that the rule has n premises and m conclusions. If $m = 1$ we say that the rule is of *linear type*, otherwise we say that the rule is of *branching type*.

2. An *expansion system* S is a finite set of expansion rules.

3. We say that S, X is an *expansion* of S under an $n \times m$ rule R if there is an n-tuple a of elements of S such that X belongs to some m-tuple b in the set of the images of a under R.

4. Let R be an $n \times m$ expansion rule. A set U is *saturated under* R or *R-saturated*, if for every n-tuple a of elements of U and every m-tuple b in the set of the images of a under R, at least one element of b is also in U.

5. A set U is **S**-*saturated* if it is R-saturated for every rule R of **S**.

The rules of an expansion system are to be read as rules which allow us to turn a tree of s-formulae into another such tree. Suppose we have a finite tree $\mathcal{T}$ and ϕ is one of its branches. Let R be an $n \times m$ expansiou rule and $(Y_1, \ldots, Y_m)$ an image under R of some n-tuple of s-formulae occurring in ϕ. Then we can extend $\mathcal{T}$ by appending m immediate snccessors $(Y_1, \ldots, Y_m)$ (in different branches) to the end or ϕ. Let us call $\mathcal{T}'$ the result. We say that $\mathcal{T}'$ *results from* $\mathcal{T}$ *by an application of* R.

We also say that the application of R is *analytic* if it has the *subformula property*, i.e. all the new s-formulae appended to the end of ϕ are subformulae of s-formulae occurring in ϕ. A *rule* R is *analytic* if every application of R is analytic (i.e. all the conclusions are subformulae of the premises).

We can then use an expansion system **S** to give a recursive definition of the notion of (analytic) **S**-tree for S, where S is a finite set of s-formulae.

Definition 2.6.5 Let $S = \{X_1, \ldots, X_n\}$.

1. The following one branch tree is an (analytic) **S**-tree for S:

$$\begin{array}{c} X_1 \\ X_2 \\ \vdots \\ X_n \end{array}$$

2. If $\mathcal{T}$ is an (analytic) **S**-tree for S and $\mathcal{T}'$ results from $\mathcal{T}$ by an (analytic) application of an expansion rule of **S**, then $\mathcal{T}'$ is also an (analytic) **S**-tree for S.

3. Nothing else is an (analytic) **S**-tree for S.

Let ϕ be a branch of an **S**-tree. We say that ϕ is *closed* if the set of its nodes is explicitly inconsistent. Otherwise it is *open*. A tree $\mathcal{T}$ is *closed* if all its branches are closed and *open* otherwise. We also say that a branch ϕ is *complete* if it is closed or the set of its nodes is **S**-saturated. A tree $\mathcal{T}$ is completed if all its branches are complete.

As far as this chapter is concerned we are interested in expansion systems which represent (analytic) *refutation systems* for classical propositional logic, i.e. systems **S** such that, for every finite set of s-formulae S, S is classically unsatisfiable if and only if there is a closed **S**-tree for S.

2.6.2 Redundant trees

Any set of rules meeting our definition of analytic refutation system can be considered an adequate formalization of the idea of proving validity by a failed attempt to construct a countermodel. Each expansion step in such as system reduces a problem concerning a set S (is S satisfiable?) to a finite set of 'easier' subproblems of the same kind concerning supersets of S. Thinking in terms of partial valuations, each step yields a finite set of more accurate approximations to the sought-for semivaluation(s).

However, we have observed that the search space has a natural structure of its own. It is therefore reasouable to require that the rules we adopt in our systematic search reflect this structure. This can be made precise as follows: given an **S**-tree $\mathcal{T}$ we can associate with each node n of $\mathcal{T}$, the set of the s-formulae occurring in the path from the root to n or, equivalently, the partial valuation v_n which assigns 1 to all the formulae A such that $\mathrm{t}(A)$ occurs in the path to n, and 0 to all the formulae A such that $\mathrm{f}(A)$ occurs in the path to n (and leaves all the other formulae undefined). We can then require that the relations between the nodes in an **S**-tree correspond to the relations between the associated partial valuations.

Definition 2.6.6 Let $\mathcal{T}$ be an **S**-tree and let $\preceq_{\mathcal{T}}$ the partial ordering defined by $\mathcal{T}$ on the set of its nodes (i.e. for all nodes n_1, n_2, $n_1 \preceq_{\mathcal{T}} n_2$ if an only if n_1 is a predecessor of n_2). We say that a refutation system **S** is *non-redundant* if the following condition is satisfied for every **S**-tree $\mathcal{T}$ and every pair n_1, n_2 of nodes of $\mathcal{T}$:

$$n_1 \preceq_{\mathcal{T}} n_2 \iff v_{n_1} \sqsubseteq v_{n_2}.$$

Non-redundancy as defined above seems a very natural requirement on refutation systems: we essentially ask that our refutation system generate trees which follow the structure of the approximation problems they set out to solve. (Notice that the *if* part of the condition is trivial, but the *only-if* part is not so trivial.) One corollary of non-redundancy, which justifies this choice of terminology, is the following: let us say that an S-tree T is *redundant* if for some pair of nodes n_1, n_2 belonging to *different* branches of T, we have that $v_{n_1} \sqsubseteq v_{n_2}$. Such a tree is obviously redundant for the reasons discussed in section 2.5: if v_{n_1} can he extended to a semivaluation, we have found a countermodel to the original sequent and the problem is solved. Otherwise, if no extension of v_{n_1} is a semivaluation, the same applies to v_{n_2}. It immediately follows from our definitions that:

Corollary 2.6.1 *If* **S** *is a non-redundant system, no* **S***-tree is redundant.*

The importance of non-redundaucy for a system is both conceptual and practical. A redundant tree does *not* reflect the structure of the semantic space of partial valuations which it is supposed to explore and this very fact has disastrous *computational* consequences: redundant systems are ill-designed, from an algorithmic point of view, in that, in some cases, they force us to repeat over and over again what is essentially the same computational process.

It is easy to see that the non-redundancy condition *is not satisfied by the tableau method* (and in general by cut-free Gentzen systems). We can therefore say that, in some sense, *such systems are not natural for classical logic*[20].

[20]This suggestion may be contrasted with Prawitz's suggestion, advanced in [Pra74], that 'Gentzen's calculus of sequents may be understood as *the* natural system for generating logical truths'.

Chapter 3

An alternative approach

3.1 Bivalence restored

The main feature of the tableau method and of all the variants of Gentzen's cut-free systems is the close correspondence between their rules and the clauses of the semantic definition of semivaluation[1]. This is the reason that such systems are usually regarded as 'natural'. But, as we have argued, the tree-structure generated by the semivaluation clauses bears a tortuous relation to the structure of the space of partial valuations, i.e. the partial semantic objects by which we represent our successive approximations to the sought-for countermodel. So, our non-redundancy condition suggests that we should *not* use such clauses as rules in our search.

Is there a simple way out? There is. We need only to take into consideration that models, or countermodels, in classical logic, are *bivalent*. As seen is section 2.3, this crucial piece of semantic information is hidden by the cut-free Gentzen rules. So in order to establish a close correspondence between formal derivability and classical semantics we have to reintroduce the notion of bivalence in our analysis. It will transpire that this is in fact *all we need to do*.

The non-redundancy condition given in definition 2.6.6 is automatically satisfied by any refutation system **S** which satisfies the stronger condition that,

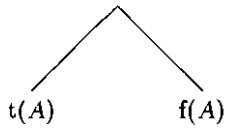
[1]This correspondence is discussed thoroughly in [Pra74].

for all S-trees $\mathcal{T}$ and all nodes n_1, n_2

$$(3.1) \qquad n_1 \not\preceq_{\mathcal{T}} n_2 \text{ and } n_2 \not\preceq_{\mathcal{T}} n_1 \Longrightarrow v_{n_1} \sqcup v_{n_2} = \top$$

i.e., any two different branches define *inconsistent* partial valuations.

It is obvious that the *only* rule of the branching type which generates trees with this property is a 0-premise rule, corresponding to the principle of bivalence:



So our discussion strongly suggests that the principle of bivalence should be *re*-introduced in some way as a rule in the search for a countermodel and that, indeed, it should be the *only* 'branching' rule to govern this search.

Three problems immediately arise in connection with the use of PB in a refutation system:

1. Are there simple analytic rules of linear type which combined with PB yield a refutation system for classical logic?

2. Since a 0-premise rule like PB can introduce arbitrary formulae, can we restrict ourselves to *analytic applications* of PB without affecting completeness?

3. Even if the previous question has a positive answer, we shall be left with a large choice of formulae to introduce via PB. This could be a problem from a practical point of view. Can we further restrict this choice?

The rest of this chapter will be devoted to studying an alternative approach based on the system **KE**, recently proposed in [Mon88a, Mon88b], which gives our three questions positive answers.

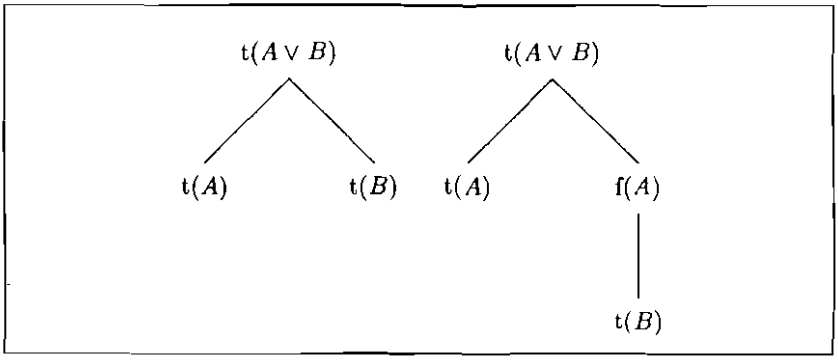


Figure 3.1: Two different analyses.

3.2 The system KE

Let us consider the case in which, at a certain point of the search tree, we examine a partial function which renders a disjunction, $A \vee B$, true. Then, instead of applying the tableau branching rule as in the left diagram of figure 3.1, we apply the rule PB as in the diagram on the right[2]. Next, we observe that any boolean valuation which makes $A \vee B$ true and A false must make B true.

If we compare the result of this way of analysing the disjunction with the result of applying the tableau branching rule, we notice that (i) the lefthand branch represents the same partial valuation, but (ii) the partial valuation represented by the righthand branch *is more defined*, i.e. contains more information: precisely the information that if A is not true, it must be false; so if there are no countermodels which make A true, every countermodel, if any, must make A false. We can then use this information to exclude from the search space associated with the righthand branch all the partial valuations which make A true, whereas such partial valuations are not excluded if we apply the standard tableau rule. In other words, our chances of closing

[2]In this example we apply PB to the formula A. A similar configuration is obtained by applying PB to the formula B. We need only *one* of these applications. Although the choice of the formula does not affect completeness, it may affect the complexity of the resulting refutation. See below p. 79.

branches are significantly increased.

The example suggests that we can find a set of rules of the linear type which, combined with PB, provides a complete refutation system for propositional logic. We need only to notice that the following eleven facts hold true (for any formulae A,B) under any boolean valuation:

1. If $A \vee B$ is true and A is false, then B is true.
2. If $A \vee B$ is true and B is false, then A is true.
3. If $A \vee B$ is false then both A and B are false.
4. If $A \wedge B$ is false and A is true, then B is false.
5. If $A \wedge B$ is false and B is true, then A is false.
6. If $A \wedge B$ is true then both A and B are true.
7. If $A \rightarrow B$ is true and A is true, then B is true.
8. If $A \rightarrow B$ is true and B is false, then A is false.
9. If $A \rightarrow B$ is false, then A is true and B is false.
10. If $\neg A$ is true, then A is false.
11. If $\neg A$ is false, then A is true.

These facts can immediately be nsed to provide a set of expansion rules of the linear type which, with the addition of PB, correspond to the the propositional fragment of the system **KE** proposed in [Mon88a, Mon88b]. The rules of **KE** are shown below. Notice that those with two s-formnlae below the line represent *a pair* of expansion rnles of the linear type, one for each s-formula.

Disjunction Rules

$$\frac{\begin{array}{l}t(A \vee B)\\ f(A)\end{array}}{t(B)}\ \text{Et}\vee 1 \qquad \frac{\begin{array}{l}t(A \vee B)\\ f(B)\end{array}}{t(A)}\ \text{Et}\vee 2 \qquad \frac{f(A \vee B)}{\begin{array}{l}f(A)\\ f(B)\end{array}}\ \textbf{Ef}\vee$$

Conjunction Rules

$$\frac{\mathrm{f}(A \wedge B)}{\begin{array}{l}\mathrm{t}(A)\\ \mathrm{f}(B)\end{array}}\ \mathrm{Ef}\wedge 1 \qquad \frac{\mathrm{f}(A \wedge B)}{\begin{array}{l}\mathrm{t}(B)\\ \mathrm{f}(A)\end{array}}\ \mathrm{Ef}\wedge 2 \qquad \frac{\mathrm{t}(A \wedge B)}{\begin{array}{l}\mathrm{t}(A)\\ \mathrm{t}(B)\end{array}}\ \mathrm{Et}\wedge$$

Implication Rules

$$\frac{\mathrm{t}(A \rightarrow B)}{\begin{array}{l}\mathrm{t}(A)\\ \mathrm{t}(B)\end{array}}\ \mathrm{Et}\rightarrow 1 \qquad \frac{\mathrm{t}(A \rightarrow B)}{\begin{array}{l}\mathrm{f}(B)\\ \mathrm{f}(A)\end{array}}\ \mathrm{Et}\rightarrow 2 \qquad \frac{\mathrm{f}(A \rightarrow B)}{\begin{array}{l}\mathrm{t}(A)\\ \mathrm{f}(B)\end{array}}\ \mathrm{Ef}\rightarrow$$

Negation Rules

$$\frac{\mathrm{t}(\neg A)}{\mathrm{f}(A)}\ \mathrm{Et}\neg \qquad \frac{\mathrm{f}(\neg A)}{\mathrm{t}(A)}\ \mathrm{Ef}\neg$$

Principle of Bivalence

$$\frac{}{\mathrm{t}(A) \mid \mathrm{f}(A)}\ \mathrm{PB}$$

The rules involving the logical operators will be called *(propositional) elimination rules* or *E-rules.*[3]

In contrast with the tableau rules for the same logical operators, the E-rules are all of the linear type and are *not* a complete set of rules for classical propositional logic. The reason is easy to see. The E-rules, intended as 'operational rules' which govern our use of the logical operators do not say anything about the bivalent structure of the intended models. If we add the rule PB as the only rule of the branching type, completeness is achieved. So

[3]Quite independently, and with a different motivation, Cellucci [Cel87] formulates the same set of rules (although he does not use signed formulae). Surprisingly, the two-premise rules in the above list were already discovered by Chrysippus who claimed them to be the fundamental rules of reasoning ('anapodeiktoi'), except that disjunction was interpreted by him in an exclusive sense. Chrysippus also maintained that his 'anapodeiktoi' formed a complete set of inference rules ('the indemonstrables are those of which the Stoics say that they need no proof to be maintained. [...] They envisage many indemonstrables but especially five, from which it seems all others can be deduced'. See [Bla70], pp.115-119 and [Boc61], p.126.

PB is *not eliminable* in the system **KE**.

We shall call an application of PB a *PB-inference* and the s-formulae which are the conclusions of the PB-inference *PB-formulae*. Finally, if $\mathrm{t}(A)$ and $\mathrm{f}(A)$ are the conclusions of a given PB-inference, we shall say that PB has been applied to the formula A.

Definition 3.2.1 Let $S = X_1, \ldots, X_m$. Then $\mathcal{T}$ *is a* **KE**-*tree for* S if there exists a finite sequence $(\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n)$ such that $\mathcal{T}_1$ is a one-branch tree consisting of the seqnence of $X_1, \ldots, X_m$, $\mathcal{T}_n = \mathcal{T}$ and for each $i < n$, $\mathcal{T}_{i+1}$ results from $\mathcal{T}_i$ by an application of a rule of **KE** (see section 2.6.1 for this terminology).

Definition 3.2.2

1. Given a tree $\mathcal{T}$ of s-formulae, a branch ϕ of $\mathcal{T}$ is *closed* if for some atomic formula P, both $\mathrm{t}(P)$ and $\mathrm{f}(P)$ are in ϕ. Otherwise it is *open*.

2. A tree $\mathcal{T}$ of s-formulae is *closed* if each branch of $\mathcal{T}$ is closed. Otherwise it is *open*.

3. A tree $\mathcal{T}$ is a **KE**-*refutation of* S if $\mathcal{T}$ is a closed **KE**-tree for S.

4. A tree $\mathcal{T}$ is a **KE**-*proof of* A *from a set* Γ *of formulae* if $\mathcal{T}$ is a **KE**-refutation of $\{\mathrm{t}(B) | B \in \Gamma\} \cup \{\mathrm{f}(A)\}$.

5. A is **KE**-*provable from* Γ if there is a **KE**-proof of A from Γ.

6. A is a **KE**-*theorem* if A is **KE**-provable from the empty set of formulae.

Remark: it is easy to prove that if a branch ϕ of $\mathcal{T}$ contains both $\mathrm{t}(A)$ and $\mathrm{f}(A)$ for some formnla A (not necessarily atomic), ϕ can be extended *by means of the E-rules only* to a branch ϕ' which is closed in the sense of the previous definition. Hence, in what follows we shall consider a branch closed as soon as both $\mathrm{t}(A)$ and $\mathrm{f}(A)$ appear in it.

As pointed out in Section 2.3, there is a close correspondence between the semantic rule PB and the cnt rule of the seqnent calculus. We shall return to this point in Section 4.5.2.

Disjunction Rules

$$\frac{A \vee B \quad \neg A}{B}\,\mathrm{E}\vee 1 \qquad \frac{A \vee B \quad \neg B}{A}\,\mathrm{E}\vee 2 \qquad \frac{\neg(A \vee B)}{\neg A \quad \neg B}\,\mathrm{E}\neg\vee$$

Conjunction Rules

$$\frac{\neg(A \wedge B) \quad A}{\neg B}\,\mathrm{E}\neg\wedge 1 \qquad \frac{\neg(A \wedge B) \quad B}{\neg A}\,\mathrm{E}\neg\wedge 2 \qquad \frac{A \wedge B}{A \quad B}\,\mathrm{E}\wedge$$

Implication Rules

$$\frac{A \rightarrow B \quad A}{B}\,\mathrm{E}\rightarrow 1 \qquad \frac{A \rightarrow B \quad \neg B}{\neg A}\,\mathrm{E}\rightarrow 2 \qquad \frac{\neg(A \rightarrow B)}{A \quad \neg B}\,\mathrm{E}\neg\rightarrow$$

Negation Rule

$$\frac{\neg\neg A}{\neg A}\,\mathrm{E}\neg\neg$$

Principle of Bivalence

$$\frac{}{A \mid \neg A}\,\mathrm{PB}$$

Table 3.1: **KE** rules for unsigned formulae.

We can give a version of **KE** which works with unsigned formulae. The rules are shown in Table 3.1. It is intended that all definitions be modified in the obvious way.

We can see from the unsigned version that the two-premise rules correspond to well-known principles of inference: *modus ponens, modus tollens, disjunctive syllogism* and the dual of disjunctive syllogism. This gives **KE** a certain natural-deduction flavour (see Section 3.8). However, the classical operators are analysed as such and not as 'stretched' versions of the constructive ones.

In Fig. 3.2 we give a **KE**-refutation (using unsigned formulae) of the

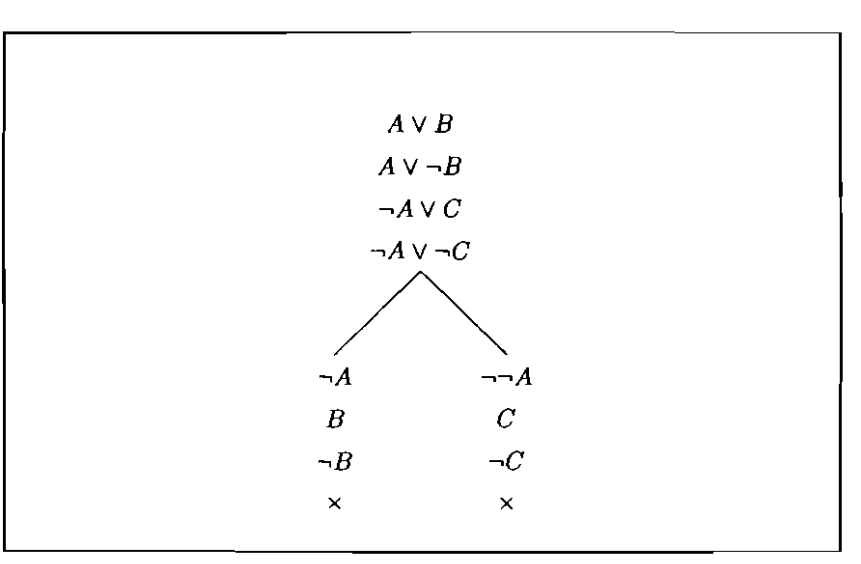


Figure 3.2: A **KE**-refutation of $\{A \vee B, A \vee \neg B, \neg A \vee C, \neg A \vee \neg C\}$

same set for which a minimal tableau was given on p. 23; the reader can compare the different structure of the two refutations and the crucial use of (the unsigned version of) PB to eliminate the redundancy exhibited by the tableau refutation.

3.3 Soundness and completeness of KE

We shall give the proofs for **KE**-trees using signed formulae. The modifications for **KE**-trees using unsigned formulae are obvious.

Proposition 3.3.1 (Soundness of KE) *If there is a closed* **KE***-tree for S, then S is unsatisfiable.*

Proof. The proof is essentially the same as the soundness proof for the tableau method. See [Smu68a, p. 25].

The completeness of **KE** can be shown in several ways. One is by proving

that the set of **KE**-theorems includes some standard set of axioms for propositional logic and is closed under *modus ponens*. Another way is by modifying the traditional completeness proof for the tableau method. We shall give both these proofs because they provide us with different kinds of information. (One can also obtain a proof *à la* Kalmar [Kal34]. See [Mon88a].)

3.3.1 Completeness of KE: proof one

Theorem 3.3.1 *If A is a valid formula than there is a* **KE***-proof of A.*

Proof. The theorem immediately follows from the following facts which at the same time provide examples of **KE**-refutations (we write just $\vdash$ for $\vdash_{\mathbf{KE}}$):

Fact 3.3.1 $\vdash A \rightarrow (B \rightarrow A)$

$\underline{\mathrm{f}(A \rightarrow (B \rightarrow A))}$
$\mathrm{t}(A)$
$\mathrm{f}(B \rightarrow A)$
$\mathrm{t}(B)$
$\mathrm{f}(A)$
$\times$

Fact 3.3.2 $\vdash (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

$\underline{\mathrm{f}((A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)))}$
$\mathrm{t}(A \rightarrow (B \rightarrow C))$
$\mathrm{f}((A \rightarrow B) \rightarrow (A \rightarrow C))$
$\mathrm{t}(A \rightarrow B)$
$\mathrm{f}(A \rightarrow C)$
$\mathrm{t}(A)$
$\mathrm{f}(C)$
$\mathrm{t}(B \rightarrow C)$
$\mathrm{f}(B)$
$\mathrm{t}(B)$
$\times$

Fact 3.3.3 $\vdash (\neg B \to \neg A) \to (A \to B)$

	$f((\neg B \to \neg A) \to (A \to B))$	
	$t(\neg B \to \neg A)$	
	$f(A \to B)$	
	$t(A)$	
	$f(B)$	
$t(\neg B)$	|	$f(\neg B)$
$t(\neg A)$		$t(B)$
$f(A)$		$\times$
$\times$		

Fact 3.3.4 *If* $\vdash A$ *and* $\vdash A \to B$, *then* $\vdash B$

Proof. It follows from the hypothesis that there are refutations $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively of $\{f(A)\}$ and $\{f(A \to B)\}$. Then the following tree:

	$f(B)$	
$t(A \to B)$	|	$f(A \to B)$
$f(A)$		$\mathcal{T}_2$
$\mathcal{T}_1$		

is a refutation of $\{f(B)\}$.□

This kind of proof provides a simulation of a standard axiomatic system. In this simulation it is essential to apply the rule PB to formulae which are not subformulae of the theorem to be proved. We notice that such a simulation cannot be directly obtained by the tableau method, since it is non-analytic. Are such non-analytic applications of PB necessary or can they be eliminated without loss of completeness? i.e. can we restrict ourselves to *analytic* applications? Let us say that a **KE**-tree $\mathcal{T}$ for Γ is *analytic* if PB is applied in $\mathcal{T}$ only to (proper) subformulae of formulae in Γ. Let us call **KE*** the *analytic restriction* of **KE**, i.e. the system in which the applications of PB are restricted to subformulae of the formulae occurring above in the same branch. So our problem becomes: is **KE*** complete?

A positive answer to this question can be obtained from the second completeness proof which follows the lines traced by Beth [Bet55] and Hintikka

[Hin55]. This proof not only ensures that we can restrict ourselves to analytic applications of PB so that our refutations will obey the subformula principle, but also provides more information concerning the formulae to be considered in these applications.

3.3.2 Completeness of KE: proof two

It is convenient to use Smullyan's unifying notation in order to reduce the number of cases to be considered.

We use the letter 'α' to stand for any signed formula of one of the forms: $\mathrm{t}(A \wedge B)$, $\mathrm{f}(A \vee B)$, $\mathrm{f}(A \rightarrow B)$, $\mathrm{t}(\neg A)$, $\mathrm{f}(\neg A)$.

For every such formula α, its *components* α_1 and α_2 are defined as in the following table:

α	α_1	α_2
$\mathrm{t}(A \wedge B)$	$\mathrm{t}(A)$	$\mathrm{t}(B)$
$\mathrm{f}(A \vee B)$	$\mathrm{f}(A)$	$\mathrm{f}(B)$
$\mathrm{f}(A \rightarrow B)$	$\mathrm{t}(A)$	$\mathrm{f}(B)$
$\mathrm{t}(\neg A)$	$\mathrm{f}(A)$	$\mathrm{f}(A)$
$\mathrm{f}(\neg A)$	$\mathrm{t}(A)$	$\mathrm{t}(A)$

We use 'β' to stand for any formula of one of the forms: $\mathrm{f}(A \wedge B)$, $\mathrm{t}(A \vee B)$, $\mathrm{t}(A \rightarrow B)$. For every such formula β, its *components* β_1 and β_2 are defined as in the following table:

β	β_1	β_2
$\mathrm{f}(A \wedge B)$	$\mathrm{f}(A)$	$\mathrm{f}(B)$
$\mathrm{t}(A \vee B)$	$\mathrm{t}(A)$	$\mathrm{t}(B)$
$\mathrm{t}(A \rightarrow B)$	$\mathrm{f}(A)$	$\mathrm{t}(B)$

So the E-rules of **KE** can be 'packed' into the following three rules (where β'_i, $i = 1, 2$ denotes the *conjugate* of β_i):

Rule A
$$\frac{\alpha}{\begin{array}{l}\alpha_1\\ \alpha_2\end{array}}$$

Rule B1
$$\frac{\begin{array}{l}\beta\\ \beta'_1\end{array}}{\beta_2}$$

Rule B2
$$\frac{\begin{array}{l}\beta\\ \beta'_2\end{array}}{\beta_1}$$

Remarks. The unifying notation can be easily adapted to unsigned formulae: simply delete all the signs 't' and replace all the signs 'f' by '¬'. The 'packed' version of the rules then suggests a more economical version of **KE** for unsigned formulae when β_i' is taken to denote the *complement* of β_i defined as follows: the *complement* of an *unsigned formula* A, is equal to $\neg B$ if $A = B$ and to B if $A = \neg B$. In this version the rules EV1, EV2 and E→2 become:

$$\frac{A \vee B \quad A'}{B} \qquad \frac{A \vee B \quad B'}{C} \qquad \frac{A \rightarrow B \quad B'}{A'}$$

This version is to be preferred for practical application. Notice also that the Rule A represents a pair of expansion rules, one with conclusion α_1 and the other with conclusion α_2.

In each application of the rules, the signed formulae α and β are called *major premises.* In each application of rules B1 and B2 the sigued formulae β_i', $i = 1,2$ are called *minor premises* (rule A has no minor premise).

We first define a notion akin to that of a Hintikka set:

Definition 3.3.1 Let us say that a set of s-formulae S is a *propositional analytic set* if and only if it satisfies the following conditions:

A_0: For no atomic formula P, t(P) and f(P) are both in S.

A_1: If $\alpha \in S$, then $\alpha_1 \in S$ and $\alpha_2 \in S$.

A_2: If $\beta \in S$ and $\beta_1' \in S$, then $\beta_2 \in S$.

A_3: If $\beta \in S$ and $\beta_2' \in S$, then $\beta_1 \in S$.

An analytic set differs from a Hintikka set in that it may be the case that for some β in the set neither β_1 nor β_2 is in the set.

Definition 3.3.2 We say that an analytic set S is *β-complete* if for every β in S one of the following two conditions is satisfied:

B_a: either $\beta_1 \in S$ or $\beta_1' \in S$;

B_b: either $\beta_2 \in S$ or $\beta_2' \in S$.

It is then easy to verify that:

Fact 3.3.5 *If S is a propositional analytic set and S is β-complete, then S is a propositional Hintikka set.*

Let us say that a branch ϕ of a **KE**-tree is *E-complete* if the set S_ϕ of the s-formulae occurring in it is saturated under the E-rules of **KE**. Let us also say that ϕ is *β-complete* if for every formula of type β occurring in it and some $i = 1, 2$, either β_i or β_i' occurs in ϕ. We say that ϕ is *complete* if it is E-complete and β-complete. Finally, we say that a **KE**-tree $\mathcal{T}$ is *completed* if every branch of $\mathcal{T}$ is either closed or complete. Of course if ϕ is an open branch in a completed **KE**-tree, the set S_ϕ of its s-formulae is a β-complete analytic set and, hence, a Hintikka set.

Thus, completeness follows from fact 3.3.5 and the propositional Hintikka lemma.

Theorem 3.3.2 (Completeness Theorem) *If $\Gamma \models A$, then there is a closed* **KE***-tree for $\{\mathrm{t}(B) | B \in \Gamma\} \cup \{\mathrm{f}(A)\}$. In particular every completed* **KE***-tree is closed.*

3.3.3 The subformula principle

Our proof of the completeness of **KE** yields the subformula principle as a corollary. Since all the rules except PB are analytic, we need only to observe that the completeness proof given above immediately implies:

Corollary 3.3.1 (Analytic cut property) *If S is unsatisfiable, then there is a closed* **KE***-tree $\mathcal{T}'$ for S such that all the applications of PB are analytic. Equivalently, the analytic restriction of* **KE** *is complete.*

A constructive proof of the subformula principle, which yields a procedure for transforming any **KE**-proof in an equivalent **KE**-proof which enjoys the subformula property, is given in [Mon88h].

Our completeness proof also gives us some additional information. Let us say that a formula of type β is *analysed in a branch ϕ* if either β_1 or β_2 occurs in ϕ. An application of PB in a branch ϕ of a **KE**-tree is *canonical* if

the s-formulae of this application are β_i and β'_i for some $i = 1, 2$ and some non-analysed formula of type β occurring in ϕ. A **KE**-*tree* is *canonical* if it contains only canonical applications of PB. It follows from our completeness proof that

Corollary 3.3.2 *If S is unsatisfiable, then there is a closed canonical* **KE**-*tree for S.*

In other words, our proof establishes the completeness of the restricted system **KE'** obtained by replacing the 'liberal' version of PB with one which allows only canonical applications.

There are several procedures which, given a finite or denumerable set of signed formulae S, generate a completed (canonical) **KE**-tree. Here is one possibility (adapted from [Smu68a, pp. 33–34]). Let us say that a node is *fulfilled* if (1) it is an atomic s-formula or (2) it is of type α and both α_1 and α_2 occur in all the branches passing through the node or (3) it is of type β and for every branch ϕ passing through the node all the following three conditions are satisfied: (3a) if β'_1 occurs in ϕ, β_2 also occurs in ϕ; (3b) if β'_2 occurs in ϕ, β_1 also occurs in ϕ; (3c) for some $i = 1, 2$ either β_i or β'_i occurs in ϕ. Obviously a **KE**-tree is completed if and only if every node is fulfilled.

We can now describe a simple procedure to construct a completed **KE**-tree for every denumerable set of s-formulae. Let S be arranged in a denumerable sequence $X_1, X_2, \ldots$. Start the tree with X_1. This node constitutes the *level 1*. Then fulfil[4] the origin and append X_2 to every open branch. Call all the nodes so obtained *nodes of level 2*. At the i-th step fulfil all the nodes of level $i - 1$ and append X_i to the end of each open branch. So every node gets fulfilled after a finite number of steps. The procedure either terminates with a closed **KE**-tree or runs forever. (If S is finite, the procedure always terminates). In this case we 'obtain' an *infinite* tree which is a completed **KE**-tree for S. Since the tree is finitely generated, by Kōnig's lemma it must contain an infinite branch which is obviously open. The set of s-formulae in this branch is a Hintikka set. Therefore S is satisfiable. If S is unsatisfiable the procedure generates a finite tree.

The procedure just described is a simple variant of a similar procedure based on the tableau method which is used to provide (among other things)

[4] By 'fulfilling a node' we mean applying the **KE**-rules so that the node becomes fulfilled.

a proof of the compactness theorem. Our adaptation is an example of how all the useful theoretical properties of cut-free systems can be almost immediately transferred to a system like **KE** which is *not* cut-free.

Remark. Like the tableau method, **KE** (with PB restricted to canonical applications) can be used as a method for turning arbitrary formulae into disjunctive normal form. However, the DNFs obtained by means of the tableau method may often be highly redundant and contain a number of conjunctions which are subsumed by other conjunctions (a conjunction C_1 *subsumes* another conjunction C_2, if all the literals occurring in C_1 occur also in C_2). By contrast the DNFs obtained by means of **KE** are in most cases remarkably concise. The dual versions of the rules (i.e. the rules obtained from rules A, B_1 and B_2 by swapping α and β) provide a method for turning an arbitrary formula into conjunctive (or 'clausal') normal form.

$\dfrac{\neg\neg A}{A}$	$\dfrac{\beta}{\beta_1 \mid \beta_2}$	$\dfrac{\alpha}{\alpha'_1 \\ \alpha_2}$	$\dfrac{\alpha}{\alpha'_2 \\ \alpha_1}$	$A \mid \neg A$

Table 3.2: CNF reduction rules.

Again, the CNFs yielded by this method are in most cases much more concise than those yielded by the analogous dual version of the tableau method (see [Fit90, p. 26]). It is a consequence of results in Section 4.5.2 that in the worst case the number of (disjunctions) conjunctions in the shortest[5] normal form of A yielded by the (duals of the) tableau rules is not even polynomially related to the corresponding number of (disjunctions) conjunctions yielded by the (duals of the) **KE** rules. By contrast the (duals of the) **KE** rules *never* yield normal forms which are longer than those yielded by the (duals of the) tableau rules.

[5]That is to say, we consider the non-deterministic version of the algorithm.

3.4 KE and the Davis-Putnam Procedure

The Davis-Putnam procedure was introduced in 1960 [DP60] and later refined in [DLL62]. It was meant as an efficient theorem proving method[6] for (prenex normal form) first-order logic, but it was soon recognized that it combined an efficient test for truth-functional validity with a wasteful search through the Herbrand universe[7]. This situation was later remedied by the emergence of unification. However, at the propositional level, the procedure is still considered among the most efficient, and is clearly connected with the resolution method, so that Robinson's resolution [Rob65] can be viewed as a (non-deterministic) combination of the Davis-Putnam propositional module and unification, in a single inference rule. It is not difficnlt to see that, if we extend our language to deal with 'generalized' disjunctions and conjunctions, the Davis-Putnam procedure (in the version of [DLL62] which is also the one exposed in [CL73, Section 4.6] and in Fitting's recent book [Fit90, Section 4.4]), can be represented as a special case of the canonical procedure for **KE** outlined in the previous section. So, from this point of view, **KE** provides a generalization of the Davis-Putnam procedure which does not require reduction in clausal form.

We enrich our langnage to include expression of the form $A_1 \vee \ldots \vee A_n$ and $A_1 \wedge \ldots \wedge A_n$, where $A_1, \ldots, A_n$ are formulae, and use the notation $A_1 \circ \ldots \circ A_n \setminus A_i$, where $\circ$ is $\vee$ or $\wedge$, $i = 1, \ldots, n$, to denote the result of removing A_i from $A_1 \circ \ldots \circ A_n$. We then replace the rules E$\vee i$, **E**$\neg\wedge i$, $i = 1, 2$, E$\neg\vee$, E$\wedge$ of **KE** with the generalized versions shown in Table 3.3 (we use the notation A' for the *complement* of A).

We can generalize Smullyan's notation to cover the new language including n-ary conjunctions and disjunctions: use the notation α^n for an n-ary conjunction or the negation of an n-ary disjunction, and the notation β^n for an n-ary disjunction or the negation of an n-ary conjunction (or an n-ary implication $A_1 \rightarrow (\ldots(A_{n-1} \rightarrow A_n)\ldots)$). The components α_i^n and β_i^n, for $i = 1, \ldots, n$ are defined in the obvious way. The notation $\beta^{n\setminus i}$, $i = 1, \ldots, n$

[6] The version given in [DP60] was not in fact a completely deterministic procedure: it involved the choice of which literal to eliminate at each step.

[7] See [Dav83].

$$\frac{A_1 \vee \ldots \vee A_n \quad A_i'}{(A_1 \vee \ldots \vee A_n) \setminus A_i} \qquad \frac{\neg(A_1 \wedge \ldots \wedge A_n) \quad A_i}{\neg((A_1 \wedge \ldots \wedge A_n) \setminus A_i)}$$

$$\frac{\neg(A_1 \vee \ldots \vee A_n)}{\begin{array}{c}\neg A_1 \\ \vdots \\ \neg A_n\end{array}} \qquad \frac{A_1 \wedge \ldots \wedge A_n}{\begin{array}{c}A_1 \\ \vdots \\ A_n\end{array}}$$

Table 3.3: Generalized **KE** rules.

is used as follows:

$$\beta^{n\setminus i} = \begin{cases} (A_1 \vee \ldots \vee A_n) \setminus A_i & \text{if } \beta^n = A_1 \vee \ldots \vee A_n \\ \neg((A_1 \vee \ldots \vee A_n) \setminus A_i) & \text{if } \beta^n = \neg(A_1 \wedge \ldots \wedge A_n) \end{cases}$$

Let us consider now the procedure given in the previous section and replace the definition of fulfilled node as follows: let us say that a node is *fulfilled* if (1′) it is a literal or (2′) it is of type α^n and, for every $i = 1, \ldots, n$, α_i^n occurs in all the branches passing through the node or (3′) it is of type β^n and for every branch ϕ passing through the node all the following conditions are satisfied: (3′a) if $\beta_i^{n\prime}$ occurs in ϕ, $\beta^{n\setminus i}$ also occurs in ϕ; (3′b) for some $i = 1, 2, \ldots, n$ either β_i^n or $\beta_i^{n\prime}$ occurs in ϕ.

It is not difficult to see that the same procedure described in the previous section is complete for this extended language. Moreover, if we restrict our attention to formulae in CNF and add two simplification rules corresponding to the 'affirmative-negative' rule and to the 'subsumption rule' (see [DLL62] and [Fit90]), the procedure becomes equivalent to the Davis-Putnam procedure. So **KE** incorporates the basic idea underlying resolution theorem proving in a tableau-like set up. From this point of view it is somehow related to non-clausal resolution [Mur82, MW80].

We also notice that the system resulting from this generalized version of **KE** by disallowing the branching rule PB includes, as a special case, the restrictiou of resolution known as *unit resolution* [Cha70]. This restricted version of **KE** can then be seen as an extension of unit resolution (it is therefore a complete system for Horn clauses, although its scope is not confined

to formulae in clausal form).

3.5 The first-order system KEQ

In this section we consider a standard first-order language with no functional symbols. We use the letters $x, y, z, \ldots$ (possibly with subscripts) as individual *variables* and the letters $a, b, c, \ldots$ (possibly with subscripts) as *parameters*. For any variable x and parameter a, $A(x/a)$ will be the result of substituting all the free occurrences of x in A with a. *Subformulae* are defined in the usual way, so that for every parameter a, $A(x/a)$ is a subformula of $\forall x A(x)$.

The rules of the first-order system **KEQ** consist of the rules of the propositional fragment (with the obvious assumptiou that the metavariables range over closed first-order formulae) plus the quantifier rules of the tableau method, namely:

Universal Quantifier Rules

$$\frac{\mathrm{t}((\forall x)A)}{\mathrm{t}(A(x/a))}\mathrm{Et}\forall \qquad \frac{\mathrm{f}((\forall x)A)}{\mathrm{f}(A(x/a))}\mathrm{Ef}\forall \quad \text{with } a \text{ new}$$

Existential Quantifier Rules

$$\frac{\mathrm{f}((\exists x)A)}{\mathrm{f}(A(x/a))}\mathrm{Ef}\exists \qquad \frac{\mathrm{t}((\exists x)A)}{\mathrm{t}(A(x/a))}\mathrm{Et}\exists \quad \text{with } a \text{ new}$$

The versions for unsigned formulae are obtained, as before, by changing the sign f into '$\neg$' and deleting the sign t.

We shall use Smullyan's nnifying notation for first-order formulae. Formulae of type α and β are defined as in the propositional case (except that formulae here means closed first-order formulae). In addition we use γ to denote any formula of one of the forms $\mathrm{t}(\forall x A(x))$, $\mathrm{f}(\exists x A(x))$, and by $\gamma(a)$ we shall mean, respectively, $\mathrm{t}(A(x/a))$, $\mathrm{f}(A(x/a))$. Similarly δ will denote any formula of one of the forms $\mathrm{t}(\exists x A(x))$, $\mathrm{f}(\forall x A(x))$ and by $\delta(a)$ we shall mean, respectively, $\mathrm{t}(A(x/a))$, $\mathrm{f}(A(x/a))$.

Using the unifying notation the quantifier rules are:

Rule C $\dfrac{\gamma}{\gamma(a)}$ for *any* parameter a

Rule D $\dfrac{\delta}{\delta(a)}$ for a *new* parameter a

3.6 Soundness and completeness of KEQ

The first-order system **KEQ** is obviously sound for the same reasons as the tableau method. It is easy to extend the first completeness proof given for the propositional fragment in section 3.3.1. We just need to observe that the set of **KEQ**-theorems is closed under the standard quantifier rules of a first-order axiomatic system.

Fact 3.6.1 *If* $\vdash_{\mathbf{KEQ}} A \rightarrow B(a)$ *and* a *does not occur in* A, *then* $\vdash_{\mathbf{KEQ}} A \rightarrow (\forall x)B(a/x)$

Fact 3.6.2 *If* $\vdash_{\mathbf{KEQ}} B(a) \rightarrow A$ *and* a *does not occur in* A, *then* $\vdash_{\mathbf{KEQ}} (\exists x)B \rightarrow A$.

The second completeness proof given in section 3.3.2 can also be readily extended to the first-order system.

We recall the notion of first-order Hintikka set.

Definition 3.6.1 By a first-order Hintikka set (for a universe U) we mean a set S of s-formulae (with constants in U) such that the following conditions are satisfied for every $\alpha, \beta, \gamma, \delta$ over U:

H_0: No signed variable and its complement are both in S.

H_1: If $\alpha \in S$, then $\alpha_1 \in S$ and $\alpha_2 \in S$.

H_3: If $\beta \in S$, then $\beta_1 \in S$ or $\beta_2 \in S$.

H_4: If $\gamma \in S$, then for *every* a in U, $\gamma(a) \in S$.

H_5: If $\delta \in S$, then for *some* a in U, $\delta(a) \in S$.

The notion of *first-order analytic set* over U is the analogous extension of the propositional notion:

Definition 3.6.2 By a *first-order analytic set* (for a universe U we mean a set S of s-formulae (with constants in U) such that the following conditions are satisfied for every $\alpha, \beta, \gamma, \delta$ over U:

A_0: No signed variable and its complement are hoth in X.

A_1: If $\alpha \in S$, then $\alpha_1 \in S$ and $\alpha_2 \in S$.

A_2: If $\beta \in S$ and $\beta_1' \in S$, then $\beta_2 \in S$.

A_3: If $\beta \in S$ and $\beta_2' \in S$, then $\beta_1 \in S$.

A_4: If $\gamma \in S$, then for *every* a in U, $\gamma(a) \in X$.

A_5: If $\delta \in S$, then for *some* a in U, $\delta(a) \in S$.

Again we observe that if S is a first-order analytic set and S is β-complete, S is a first-order Hintikka set.

It is not difficult to define a procedure which, given a set of s-formulae S, generates either a (canonical) closed **KEQ**-tree or an open **KEQ**-tree such that for every (possibly infinite) opeu branch ϕ, the set S_ϕ of the s-formulae occurring in ϕ is an analytic set which is also β-complete (the only tricky part of such a procedure concerns condition A_4 and can be dealt with as in [Smu68a, pp.58–60]). Thus, completeness follows from Fact 3.3.5 and Hintikka's lemma for first-order logic. This completeness proof establishes the analogs for the first-order system of Propositions 3.3.1 and 3.3.2. In formulating a refutation procedure to be used in practical applications, Skolem functions and unification can be employed exactly as with the standard tableau method. The reader is referred to [Fit90] on this topic. A 'naive' Prolog implementation of a **KE**-based theorem prover for classical first order logic has been developed by Rajev Gore [Gor90]. Comparisons with a similar implementation of the tableau method given in [Fit90] (see also [Fit88]) fully confirm our theoretical predictions about the relative efficiency of the two systems.

We have therefore shown that **KEQ** provides positive answers to the three

questions at the end of the previous section: that **KEQ** is complete for classical first-order logic; that we can restrict ourselves to analytic applications of PB; moreover, we are able to determine the formulae to be considered for these analytic applications with a degree of precision which leaves little room for guesswork[8].

3.7 A digression on direct proofs: the system KI

The abstract notion of expansion system outlined in Section 2.6.1 is by no means restricted to representing *refutation* systems. Nor is there any assumption that the expansion rules should be *elimination* rules. We can in fact obtain a *proof* system for classical propositional logic if, instead of considering the 'analytic' rules of **KE**, we consider tbe following 'synthetic' rules, corresponding to those of the system **KI** [Mon88d, Mon89, Mon88c]:

Disjunction Rules

$$\frac{\begin{array}{c}\mathrm{f}(A)\\ \mathrm{f}(B)\end{array}}{\mathrm{f}(A \vee B)}\ \mathrm{If}\vee \qquad \frac{\mathrm{t}(A)}{\mathrm{t}(A \vee B)}\ \mathrm{It}\vee 1 \qquad \frac{\mathrm{t}(B)}{\mathrm{t}(A \vee B)}\ \mathrm{It}\vee 2$$

Conjunction Rules

$$\frac{\begin{array}{c}\mathrm{t}(A)\\ \mathrm{t}(B)\end{array}}{\mathrm{t}(A \wedge B)}\ \mathrm{It}\wedge \qquad \frac{\mathrm{f}(A)}{\mathrm{f}(A \wedge B)}\ \mathrm{If}\wedge 1 \qquad \frac{\mathrm{f}(B)}{\mathrm{f}(A \wedge B)}\ \mathrm{If}\wedge 2$$

Implication Rules

$$\frac{\begin{array}{c}\mathrm{t}(A)\\ \mathrm{f}(B)\end{array}}{\mathrm{f}(A \rightarrow B)}\ \mathrm{If}\rightarrow \qquad \frac{\mathrm{f}(A)}{\mathrm{t}(A \rightarrow B)}\ \mathrm{It}\rightarrow 1 \qquad \frac{\mathrm{f}(B)}{\mathrm{t}(A \rightarrow B)}\ \mathrm{It}\rightarrow 2$$

[8] Of course there is still plenty of room for heuristics which help us to choose which formula should he analysed next.

Negation Rules

$$\frac{\mathrm{t}(A)}{\mathrm{f}(\neg A)}\,\mathrm{If}\neg \qquad \frac{\mathrm{f}(A)}{\mathrm{t}(\neg A)}\,\mathrm{It}\neg$$

Principle of Bivalence

$$\frac{}{\mathrm{t}(A) \mid \mathrm{f}(A)}\,\mathrm{PB}$$

The rules involving the logical operators will be called *introduction rules* or *I-rules*. A **KI**-tree for S will be, as usual, an expansion tree regulated by the rules of **KI** starting from s-formulae in S (when S is empty, the origin of the tree is labelled with $\oslash$). *Closed* and *open branches* are defined in the usual way. A **KI**-*proof* of A from Γ is a **KI**-tree for $\{\mathrm{t}(B) | B \in \Gamma\}$, such that $\mathrm{t}(A)$ occurs in every open branch. We say that A is a **KI**-*theorem* if A is provable from the empty set of formulae. A version for unsigned formulae can be obtained, as before, by changing all f's into $\neg$, and deleting all t's. Mondadori [Mon88c, Mon89] has also formulated a first-order version of the system which will not concern us here.

The I-rules of **KI** correspond to the notion of *inductive valuation* (see [Pra74]).

Definition 3.7.1 We shall call *inductive set* a set of s-formulae saturated under the I-rules. We shall call a set of s-formulae a *base* when it is used for the inductive definition of an inductive set. So *the inductive set generated by a base* U will be the smallest inductive set which includes U. Finally, a *synthetic set* will be an inductive set generated by a base U such that all the elements of U are atomic s-formulae and for no atomic formula P both $\mathrm{t}(P)$ and $\mathrm{f}(P)$ are in U.

The notion of a synthetic set bears the same relation to the calculus **KI** as does the notion of an analytic set to **KE**.

Our definitions make sense if, instead of considering the set E of all the formulae of propositional calculus, we restrict our attention to a subset which is closed under subformulae. If we denote by $\widehat{\Gamma}$ the set of all subformulae of formulae in Γ, by *a set of s-formulae over* $\widehat{\Gamma}$ we mean a set of s-formulae $\mathrm{s}(A)$ $(\mathrm{s} = \mathrm{t}, \mathrm{f})$, with $A \in \widehat{\Gamma}$. The related notions of a synthetic set, a truth

set and saturation under a rule, with reference to sets over $\widehat{\Gamma}$, are intended to be modified in the obvious way. For example, by saying that a set S over $\widehat{\Gamma}$ is saturated under PB, we shall mean that for every formula A in $\widehat{\Gamma}$ either $\mathrm{t}(A) \in S$ or $\mathrm{f}(A) \in S$.

It is easy to verify that:

Proposition 3.7.1 (Inversion Principle) *If S is a synthetic (analytic) set over $\widehat{\Gamma}$ and S is saturated under PB, then S is an analytic (synthetic) set over $\widehat{\Gamma}$.*

Corollary 3.7.1 *If S is a synthetic (analytic) set over $\widehat{\Gamma}$ and S is saturated under PB, then S is a truth set over $\widehat{\Gamma}$.*

The completeness of **KI** as well as the subformula principle follow immediately from corollary 3.7.1.

Let us say that an application of PB is *atomic* if the s-formulae resulting from the application are $\mathrm{t}(P)$ and $\mathrm{f}(P)$ for some atomic formula P. Let us denote by PB* the expansion rule resulting from restricting PB to atomic applications. Intuitively, it is obvious that **KI** is complete for classical logic even if we replace PB with PB*. Formally, this can be seen to be a consequence of the following lemma:

Lemma 3.7.1 (Truth-table lemma) *If S is a synthetic set over $\widehat{\Gamma}$, and S is saturated under PB*, then S is a truth set over $\widehat{\Gamma}$.*

So, given an arbitrary formula A to be tested for tautologyhood, we can use **KI** as a simulation in tree form of the familiar truth-tables. However, if we postpone the applications of PB* until no further application of an I-rule (over the set of subformulae of A) is possible, we may in many cases stop expanding a branch before PB* has been applied to *all* the atomic formulae. This kind of procedure amounts to a 'lazy evaluation' of boolean formulae via partial truth-assignments.

3.8 Analytic natural deduction

As mentioned before, the rules of **KE** have a certain natural deduction flavour. We can make this more evident by changing the format of the

rules (in the version for unsigned formulae). It is convenient to extend the language to include the constant F, standing for 'the absurd'. The natural deduction system **KEND** is characterized by the following rules:

Disjunction Rules

$$\frac{A \vee B \quad \neg A}{B} \qquad \frac{A \vee B \quad \neg B}{A} \qquad \frac{\neg(A \vee B)}{\neg A} \qquad \frac{\neg(A \vee B)}{\neg B}$$

Conjunction Rules

$$\frac{\neg(A \wedge B) \quad A}{\neg B} \qquad \frac{\neg(A \wedge B) \quad B}{\neg A} \qquad \frac{A \wedge B}{A} \qquad \frac{A \wedge B}{B}$$

Implication Rules

$$\frac{A \rightarrow B \quad A}{B} \qquad \frac{A \rightarrow B \quad \neg B}{\neg A} \qquad \frac{\neg(A \rightarrow B)}{A} \qquad \frac{\neg(A \rightarrow B)}{\neg B}$$

Negation Rule

$$\frac{\neg\neg A}{A}$$

Absurdum Rule

$$\frac{A \quad \neg A}{F}$$

PB

$$\frac{\begin{array}{cc}[A] & [\neg A] \\ \vdots & \vdots \\ F & F\end{array}}{F}$$

Every application of PB *discharges* the assumptions A and $\neg A$. A **KEND**-tree is a tree of formulae regulated by the above rules. As in natural deduc-

tion, a *proof of A from* Γ is a **KEND**-tree with origin A and such that its undischarged assumptions are in Γ.

The rules can be put in linear form by using a Fitch-style representation or nested boxes (as in [Smu65]). In this case instead of PB we can use a versiou of classical Reductio ad Absurdum:

$$\begin{array}{c} [\neg A] \\ \vdots \\ F \\ \hline A \end{array} \text{ [RA]}$$

In this last format the rules of **KEND** have been independently proposed by Cellucci [Cel87].

3.9 Non-classical logics

One of the attractions of the tableau method is that it can be adapted to a variety of non-classical logics in a relatively simple way. In all the cases in which a Kripke-style characterization exists[9], these adaptations essentially consist of systems for reasoning classically about Kripke models. For example, in the case of intuitionistic and standard modal logics, the tableau rules systematically search for a Kripke model which is a countermodel of the alleged theorem (whereas in the case of modal logics the use of signed formulae is optional, in the case of intuitionistic logic it is compulsory). As in the classical case, a proof of A is a frustrated attempt to construct a Kripke model in which A is not true. Signed formulae t(B) and f(B) are interpreted as 'B is true in the current world' and 'B is not true in the current world', respectively. The crucial feature of these non-classical tableaux is the use of 'up-dating' or 'branch modification rules' (see [Fit83]) to reflect the jump from one possible world to another. We just mention here that exactly the same devices can be used in the context of **KE**. The typical **KE** rule, namely PB, expresses the *classical* postulate that for every proposition A and every Kripke model $\mathcal{M}$, either A is true in $\mathcal{M}$ or A is not true in $\mathcal{M}$. Given the branch modification rules defined by Fitting, the **KE**-style versions of intuitionistic and analytic modal logics are obtained by means of

[9] See [Fit83].

trivial adaptations. Moreover, the rule PB is *necessary* in the context of non-analytic modal logics (like S5) so that the **KE**-version looks, in these cases, less *ad hoc*. The same redundancy of the cut-free rules pointed out in the case of classical logic can also be observed for these non-classical logics. Therefore the **KE**-versions also result in an improvement in efficiency. In Chapter 5 we shall study a non-classical logic, Belnap's four-valued logic, which has an interesting interpretation from a computer science viewpoint, and shall formulate two new methods corresponding to both kinds of tableaux (standard and **KE**-like) which provide particularly simple characterizations of it.

Chapter 4

Computational complexity

4.1 Absolute and relative complexity

The subject of computational complexity can be seen as a refinement of the traditional theory of computability. The refinement, which is motivated by practical considerations and above all by the rapid development of computer science, consists of replacing the fundamental question, 'Is the problem P computationally solvable?' with the question, 'Is P solvable within bounded resources (time and space)?'. Workers in computational complexity agree in identifying the class of 'practically solvable' or 'feasible' problems with the class $\mathcal{P}$ of the problems that can be solved by a Turing machine within polynomial time, i.e. time bounded above by a polynomial in the length of the input.

Most computational problems can be viewed as language-recognition problems i.e. problems which ask whether or not a word over a given alphabet is a member of some distinguished set of words. For instance, the problem of deciding whether a formula of the propositional calculus is a tautology can be identified with the set TAUT of all the words over the alphabet of propositional calculus which express tautologies, and an algorithm which solves the problem is one which decides, given a word over the alphabet, whether or not it belongs to TAUT. So the class $\mathcal{P}$ can be described as the class of the languages which can be recognized in polynomial time by a Turing machine.

The rationale of this identification of feasible problems with sets in $\mathcal{P}$ is that, as the length of the input grows, exponential time algorithms require

resources which quickly precipitate heyond any practical constraint. Needless to say, an exponential time algorithm may be preferable in practice to a polynomial time algorithm with running time, say, n^{1000}. However, the notion of polynomial time computability is theoretically nseful because it is particularly robust: it is invariant under any reasonable choice of models of computation. In fact, there is an analog of the Church-Turing thesis in the field of computational complexity, namely the thesis that a Turing machine can simulate any 'reasonable' model of computation with at most a polynomial increase in time and space. Moreover polynomial time computability is invariant under any reasonable choice of 'encoding scheme' for the problem under consideration. Finally, 'natural problems', i.e. problems which arise in practice and are not specifically constructed in order to defy the power of our computational devices, seem to show a tendency to be either intractable or solvable in time bounded by a polynomial of reasonably low degree.

The analog of the class $\mathcal{P}$, when non-deterministic models of computation are considered, for example non-deterministic Turing machines[1], is the class $\mathcal{NP}$ of the problems which are 'solved' in polynomial time by some non-deterministic algorithm. The class $\mathcal{NP}$ can be viewed as the class of all languages L such that, for every word $w \in L$, there is a 'short' proof of its membership in L, where 'short' means that the length of the proof is bounded above by some polynomial fnnction of the length of w. (See [GJ79] and [Sto87] for definitions in terms of non-deterministic Turing machines.) The central role played by propositional logic in theoretical computer science is related to the following well-known resnlts [Coo71, CR74]:

1. There is a deterministic polynomial time algorithm for the tautology problem if and only if $\mathcal{P} = \mathcal{NP}$.

2. There is a non-deterministic polynomial time algorithm for the tautology problem if and only if $\mathcal{NP}$ is closed under complementation.

As far as the first result is concerned, the theory of $\mathcal{NP}$-completeness[2] is providing growing evidence for the conjecture that $\mathcal{P} \neq \mathcal{NP}$, which wonld imply that no *proof procedure* can be uniformly feasible for the whole class of tautologies (it can of course be feasible for a number of infinite subclasses of this class).

[1] See [GJ79] and [Sto87].

[2] We refer the reader to [GJ79].

The second result involves the notion of a *proof system* rather than the notion of a proof procedure. The following definitions are due to Cook and Rechow [CR74] (Σ^* denotes the set of all finite strings or 'words' over the alphabet Σ):

Definition 4.1.1 If $L \subset \Sigma^*$, a *proof system* for L is a function $f : \Sigma_1^* \mapsto L$ for some alphabet Σ_1, where $f \in \mathcal{L}$ (the class of functions computable in polynomial time).

The condition that $f \in \mathcal{L}$ is intended to ensure that there is a feasible way, when given a string over Σ_1, of checking whether it represents a proof and what it is a proof *of*. So, for example, a proof system **S** is associated with a function f such that $f(x) = A$ if x is a string of symbols which represents a legitimate proof of A in **S**. If x does not represent a proof in **S**, then $f(x)$ is taken to denote some fixed tautology in L.

Definition 4.1.2 A proof system f is *polynomially bounded* if there is a polynomial $p(n)$ such that for all $y \in L$, there is an $x \in \Sigma_1^*$ such that $y = f(x)$ and $|x| \leq p(|y|)$, where $|z|$ is the length of the string z.

This definition captures the idea of a proof system in which, for every element of L, there *exists* a 'short' proof of its membership in L. If a proof system is polynomially bounded, this does not imply (unless $\mathcal{P} = \mathcal{NP}$) that there is a proof procedure based on it (namely a deterministic version) which is polynomially bounded. On the other hand if a proof system is *not* polynomially bounded, *a fortiori* there is no polynomially bounded proof procedure based on it.

The question of whether a proof system is polynomially bounded or not is one concerning its *absolute* complexity. As we will see, most conventional proof systems for propositional logic have been sbown *not* to be polynomially bounded by exhibiting for each system some infinite class of 'hard examples' which have no polynomial size proofs. One consequence of these results (which will be reviewed later), as far as the use of proof systems for automated deduction is concerned, is that we should not expect a complete proof system to be feasible and should be prepared either to give up completeness and restrict our language in order to attain feasibility (this is the line chosen for most of the resolution-based applications), or to appeal to suitable *heuristics*, namely *fallible* 'strategies' to guide our proofs. In fact the results mentioned

above imply that heuristics alone is not sufficient if we want to be ahle to obtain proofs expressible as formal derivations in some couveutional system. So we should be prepared to use heuristics *and* give up completeuess.

However the importance of the complexity analysis of proof systems is by no means restricted to the $\mathcal{P}$ versus $\mathcal{NP}$ question. Nor should we conclude that all conventional systems are to be regarded as equivaleut and that the only difference is caused by the heuristics that we use. Besides the questions concerning the absolute complexity of proof systems, there are many interesting ones concerning their *relative* complexity which are computationally significant even when the systems have been proved intractable. As far as automated deduction is concerned, such questions of relative complexity are relevant, before any heuristic considerations, to the choice of an appropriate formal system to start with.

4.2 Relative complexity and simulations

Let **S** be a proof system for propositional logic. We write

$$\Gamma \overset{n}{\vdash}_{\mathbf{S}} A$$

to mean that there is a proof π of A from Γ in the system **S** such that $|\pi| \leq n$ (where $|\pi|$ denotes as usual the *length* of π intended as a string of symbols over the alphabet of **S**).

Suppose that, given two systems **S** and **S'**, there is a fuuctiou g such that for all Γ, A:

$$\Gamma \overset{n}{\vdash}_{\mathbf{S}'} A \Longrightarrow \Gamma \overset{g(n)}{\vdash}_{\mathbf{S}} A \tag{4.1}$$

we are interested in the rate of growth of g for particular systems **S** and **S'**. Positive results about the above relation are usually obtained by means of *simulation procedures*:

Definition 4.2.1 If $f_1 : \Sigma_1^* \mapsto L$ aud $f_2 : \Sigma_2^* \mapsto L$ are proof systems for L, a *simulation* of f_1 in f_2 is a computable function $h : \Sigma_1^* \mapsto \Sigma_2^*$ such that $f_2(h(x)) = f_1(x)$ for all $x \in L$.

Negative results consist of *lower bounds* for the function g.

An important special case of the relation in (4.1) occurs when $g(n)$ is a polynomial in n. This can be shown by exhibiting a simulation function

(as in definition 4.2.1) h such that for some polynomial $p(n)$, $h(x) \leq p(|x|)$ for all x. In this case **S** is said to *polynomially simulate*[3], or shortly *p-simulate*, **S'**. A p-simulation is theu a mapping from proofs in **S'** to proofs in **S** which preserves feasibility: if **S'** is a polynomially bounded system for L, so is **S** (where L can be any infinite subset of TAUT). The p-simulation is obviously a partial ordering and its symmetric closure is an equivalence relation. We can therefore order proof systems and put them into equivalence classes with respect to their relative complexity. Systems belonging to the same equivalence class can be considered as having 'essentially' (i.e. up to a polynomial) the same complexity. Ou the other hand if **S** p-simulates **S'**, but there is no p-simulation in the reverse directiou, we can say that **S** is essentially more efficient than **S'** — **S** is polynomially bounded for every $L \subset$ TAUT for which **S'** is polynomially bounded but the opposite is not true; therefore **S** has a larger 'practical' scope than **S'**.

The study of the relative complexity of proof systems was started by Cook and Rechow [CR74, CR79]. Later on, some open questions were settled and new ones have been raised. In section 4.5 we shall analyse the relative complexity of **KE** and **KI** with respect to other proof systems. First, in the next section, we shall briefly review the most importaut results concerning the absolute and relative complexity of conventional proof systems.

4.3 An overview

In this section we shall use the notation $\mathbf{S'} \leq_p \mathbf{S}$, where **S** and **S'** are propositional proof systems for 'S' is p-reducible to **S**' or equivalently 'S p-simulates **S**''. The notations $\mathbf{S'} <_p \mathbf{S}$ and $\mathbf{S'} \equiv_p \mathbf{S}$ are used in the obvious way.

The tableau method was among the first systems to be recognized as intractable for fairly simple examples [CR74]. Cook and Rechow remarked that their hard examples had easy (linear) resolution proofs, i.e.

(4.2) Tableau method $\not\geq_p$ Resolution[4].

[3]Our definition of p-simulation is slightly different from the original one given, for instance, in [CR79]. However it is easy to see that it serves exactly the same purposes as far as the study of the relative complexity of proof systems is concerned. Our definition is the same as the one used in [Bus87].

[4]Because resolution is restricted to clausal form any comparison with other proof sys-

The intractability of the tableau method obviously implies the intractability of the Gentzen cut-free system in *tree form* (with or without thinning), given the easy correspondence between tableau refutations and cut-free proofs in tree form. A rigorous proof of the latter result, using a different class of hard examples, is contained in [Sta78]. Statman remarked that his examples had polynomial size proofs in the Gentzen system with cut. Hence

(4.3) Cut-free Gentzen (tree) $\not\geq_p$ Gentzen with cut (tree)

Since it is obvious that

(4.4) Gentzen with cut (tree) $\geq_p$ Cut-free Gentzen (tree)

Statman's result showed that the system with cut is strictly more powerful than the cut-free system, even at the propositional level. The intractability of the cut-free Gentzen system *without thinning*, when proofs are arranged as sequences or directed acyclic graphs (d.a.g.'s) of sequents instead of trees, was proved by Cook and Rackoff [Coo78] using a class of tautologies known as the 'pigeonhole principle' encoding the so-called 'occupancy problem' studied by Cook and Karp[5].

In [CR79] the authors consider families of proof systems which generalize the traditional Gentzen system with cut, Natural Deduction and the Hilbert-style axiomatic systems (called 'Frege systems'), and show that p-simulation is possible between any two members of each family. Moreover, they show that all three families are in the same complexity class, namely:

(4.5) Gentzen with cut (d.a.g.) $\equiv_p$ Natural Deduction (d.a.g.)

(4.6) $\equiv_p$ Frege systems

An exponential lower bonnd for cut-free Gentzen systems had been proved by Tseitin in [Tse68] (the first paper on the topic), with the proviso that proofs are *regular*, i.e. (moving from the root to the leaves) a formula which has been eliminated in a branch cannot be reintroduced in the same branch. Urquhart has recently proved an exponential lower bound for unrestricted

tems is intended to be made in the domain of formulae in clausal form (or some suitable syntactical variant).

[5]See [Coo71] and [Kar72]. See also [DH76], in which the authors give a decision procedure to solve this problem efficiently.

cut-free Gentzen systems (with proofs defined as sequences or directed acyclic graphs of sequents) using a class of examples involving the biconditional operator [Urq89]. Again Urquhart's examples have polynomial size proofs if cut is allowed. Hence

(4.7) Cut-free Gentzen (d.a.g.) $\not\geq_p$ Gentzen with cut (d.a.g.)

whereas there is a trivial p-simulation in the opposite direction. Therefore:

(4.8) Cut-free Gentzen (d.a.g.) $<_p$ Gentzen with cut (d.a.g.),

It then follows from (4.5) and (4.6) that

(4.9) Cut-free Gentzen (d.a.g.) $<_p$ Natural Deduction (d.a.g.)

(4.10) $<_p$ Frege systems.

As far as resolution is concerned[6], an early intractability result was proved by Tseitin [Tse68], again under the assumption that derivations are regular (a literal cannot be eliminated and reintroduced on the same path), and later refined by Galil [Gal77]. The intractability of unrestricted resolution has been proved by Hacken [Hac85] and Urquhart [Urq87]. Hacken's proof also uses 'the pigeonhole principle' and provides a lower bound on the complexity of resolution proofs of this class of tautologies which is expouential on the cube root of the input size. Hacken's technique has been extended by Buss and Turán to give an exponential lower bound for resolution proofs of the generalized pigeonhole principle [BG88]. Urquhart's proof is based on a different construction which evolves from an idea used by Tseitiu [Tse68] in order to prove an exponential lower bound for regular resolution, and it provides a lower bound which is exponential in the input size. Urquhart explicitly notices that his examples have polynomial size proofs in Frege systems. Since Frege systems can p-simulate resolution (see [CR74]), Urquhart's result implies

(4.11) Resolution $<_p$ Frege systems

which in turn, by (4.5) and (4.6) implies

(4.12) Resolution $<_p$ Natural deduction (d.a.g.)

(4.13) Resolution $<_p$ Gentzen with cut (d.a.g.)

[6]See footnote 4.

Hacken's and Urquhart's results show that there are families of tautologies which have no feasible resolution proofs. A recently result by Chvátal and Szemerédi [CS88] shows that these are not isolated examples. Using a probabilistic analysis they prove tbat randomly generated sparse sets of clauses are very likely to be 'hard' for resolution.

Cook and Rechow [CR79] had considered the pigeonhole principle in relation to Frege systems (as the Hilbert-style axiomatic systems are known in the complexity literature) and had conjectured that there were no polynomial size proofs. This conjecture has been recently disproved by Buss [Bns87] who has exhibited polynomial size Frege proofs of this class of tautologies, so establishing (4.11) in a different way. However Ajtai has more recently proved that no such polynomial size proofs are possible if the *depth* (that is the number of alternations of $\vee$ and $\wedge$) of the formulae is bounded [Ajt88].

This series of negative results leaves, among the conventional proof systems[7], only unrestricted Frege systems and natural deduction as well as Gentzen systems with cut, as possible candidates for a polynomially bounded system. However Ajtai's theorem offers little hope for any reasonable restriction of the search space which is compatible with the existence of polynomial-size proofs of the entire class of tautologies, not to mention that even if such a restricted system were possible, the widespread conjecture that $\mathcal{P} \neq \mathcal{NP}$ would imply, if true, that we would not be able to formulate a polynomial time proof procedure based on it.

As said before, the fact that the existence of a polynomially bounded proof system seems to be higly improbable does not decrease our interest in the relative efficiency of alternative formalizations. Indeed, considerations of relative efficiency are now even more important because intractable systems may have quite different 'practical scopes', i.e. may differ cousiderably with

[7] We have not mentioned *extended systems*, which involve the introduction of new variables as abbreviations of formulae. This device was first introduced and used by Tseitin [Tse68] in the context of resolution. Cook and Rechow [CR79] defined *extended Frege systems* in a similar way and showed that the pigeonhole priuciple has polynomial size proofs in such systems. It is open whetber or not conventional Frege systems can polynomially simulate extended Frege systems. In the cited paper Cook and Rechow had conjectured that the pigeonhole principle provided a class of examples sufficient to establish a separation result. But Buss has recently shown that the two families are not discriminated by this class of examples, since there are polynomial size proofs in conventional Frege systems as well.

respect to the extension and the type of the subsets of TAUT for which they *are* polynomially bounded. From this point of view a minimal requirement for a proof system seems to be that its practical scope *properly include* that of the truth-table method which consists of using the very semantic definition of tantology (combined with the semantic definitions of the logical operators) as a proof system. In the next section we shall see that this requirement is not as trivial as is usually assumed.

4.4 Are tableaux an improvement on truth-tables?

The truth-table method, introduced by Wittgenstein in his *Tractatus Logico-Philosophicus*, provides a decision procedure for propositional logic which is immediately implementable on a machine. However, in the literature on Automated Deduction, this time-honoured method is usually mentioned, only to be immediately dismissed because of its incurable inefficiency. Here is a typical quotation from a recent textbook on this topic [8]:

> Is there a better way of testing whether a proposition A is a tautology than computing its truth table (which requires computing at least 2^n entries where n is the number of proposition symbols occurring in A)? One possibility is to work backwards, trying to find a truth assignment which makes the proposition false. In this way, one may detect failnre much earlier. This is the essence of Gentzen [cut-free] systems ...

Similarly, in another recent textbook in which the anthor advocates the use of Smullyan's semantic tableaux, we find the remark that 'tableau proofs can be very much shorter than truth-table verifications'[9]. Beth himself, who was (in the '50's) one of the inventors of tableaux, also stressed that they 'may be considered in the first place as a more convenient presentation of the familiar truth-table analysis'[10] Richard Jeffrey, in his well-known book *Formal Logic*, takes over this point and says:

[8] [Gal86, pages 44–45].
[9] [Fit90, page 39].
[10] [Bet58, page 82].

> The truth-table test is straightforward but needlessly laborious when statement letters are numerous, for the number of cases to be searched doubles with each additional letter (so that, e.g., with 10 letters there are over 1,000 cases). The truth-tree [i.e. tableau] test [...] is equally straightforward but saves labor by searching whole blocks of cases at once[11].

This appraisal of the relative efficiency of tableau proofs with respect to truth-table verifications, however, may turn out to be rather unfair. In fact, the situation is not nearly as clear-cut as it appears. It is, of course, true that a complete truth-table always requires the computation of a number of rows which grows exponentially with the number of *variables* in the expression to be decided, and also that in some cases a tableau proof can be much shorter. However, what usually goes unnoticed is that in other cases tableau proofs can be *much longer than truth-table verifications*. The reason is simple: the complexity of tableau proofs depends essentially on the *length* of the formula to be decided, whereas the complexity of truth-tables depends essentially on the number of *distinct propositional variables* which occur in it. If an expression is 'fat'[12], i.e. its length is large compared to the number of distinct variables in it, the number of branches generated by its tableau analysis may be large compared to the number of rows in its truth-table. One can easily find fat expressions for which this is the case. If we consider the *asymptotic* behaviour of the two systems, we can observe that if there is an infinite sequence S of expressions for which the shortest tableau proofs are exponential in the input size, and the input size is k^c for some $c \geq 2$, the truth-table method would perform essentially better, even asymptotically, over this sequence than the tableau method: the complexity of the truth-table verifications would be $O(2^k \cdot k^c) = O(2^{k+c\log k})$. So, in this case, the size of the tableau proofs would not even be polynomially related to the size of the truth-tables. Moreover, if S is an infinite sequence of 'truly fat' expressions of length 2^k, where k is the number of variables involved, any tableau-based refutation procedure is very likely to be ridiculed by the old truth-table method: if $f(n)$ is a function expressing a *lower bound* on the number of branches in a closed tableau for an expression of length n, then the fastest possible procedure based on the tableau method will generate,

[11] [Jef81, page 18].

[12] This rather fancy terminology is used in [DH76].

for each element of S, at least $f(2^k)$ branches, whereas the 'slow' truth-table method will always require 2^k rows. The faster $f(n)$ grows, the worse for the tableau-based procedure[13].

An extreme example is represented by the sequence of truly fat expressions in conjunctive normal form, defined as follows: given a sequence of k atomic variables $P_1, \ldots, P_k$, consider all the possible clauses containing as members, for each $i = 1, 2, \ldots, k$, either P_i or $\neg P_i$ and no other member. There are 2^k of such clauses. Let $H_{P_1,\ldots,P_k}$ denote the conjunction of these 2^k clauses. The expression $H_{P_1,\ldots,P_k}$ is unsatisfiable. For instance, H_{P_1,P_2} is the following expression in CNF:

$$P_1 \vee P_2 \ \wedge \ P_1 \vee \neg P_2 \ \wedge \ \neg P_1 \vee P_2 \ \wedge \ \neg P_1 \vee \neg P_2$$

Notice that in this case the truth-table procedure contains as many rows as clauses in the expressions, namely 2^k. In other words, this class of expressions is not 'hard' for the truth-table method. However we claim that it is hard for the tableau method[14], and therefore:

Claim 4.4.1 *The tableau method cannot p-simulate the truth-table method.*

The nature of these considerations is not only technical but also conceptual. They confirm that there is something inadequate about the tableau (and the cut-free) analysis of classical inferences. It seems that, in order to pursue the 'ideal of purity of methods' the cut-free tradition has sacrificed aspects (bivalence, cut) that are compatible with this ideal and, indeed, essential to establish a close connection between (analytic) formal derivability and classical semantics. After all, truth-tables are nothing bnt a literally-minded implementation of the classical definition of logical truth (or falsity) as truth (or falsity) *in all possible worlds* (or under all possible truth-assignments to use a less suggestive terminology) combined with the classical definition of the logical operators. In general, given a decidable logic L which admits of characterization by means of m-valued truth-tables, the complexity of the semantic decision procedure for L is essentially $O(n \cdot m^k)$ where n is the

[13]In most interesting cases $f(n)$ grows at least linearly with n and so does the number of nodes in each branch, therefore for such a sequence the truth-table method is uniformly more efficient than the tableau method.

[14]In a recent personal communication Alasdair Urquhart has suggested a way of proving our claim.

length of the input formula and k is the number of distinct variables in it. This is an upper bound which can be derived immediately from the semantic characterization of L. In the case of classical logic, in which $m = 2$, this upper bound is essentially $O(n \cdot 2^k)$. This can be taken as a 'natural' upper bound on every proof system for classical logic. In other words, it is odd for a classical proof system to generate proofs which, in some simple cases, are much more complex than the direct, unimaginative, computation based on classical semantics. This applies especially to the tableau method which is usually claimed to be a direct translation of the semantics of classical logic into a formal system (this aspect was certainly one of the motivations for its inventors, in particular for Beth). So the point we are making may also be considered as an argument against this claim.

If we take the upper bound for the truth-table computation as a 'natural' upper bound on the tautology problem, we can turn our considerations into a precise requirement on a 'natural' proof system for classical logic:

> *A 'natural' proof system for classical logic should* never *generate proofs whose complexity (significantly) exceeds the complexity of the corresponding truth-tables.*

The example given above shows that the tableau method does not satisfy this requirement. As suggested before, its poor performance is related to the way in which the tableau rules analyse classical inferences and unfold the information content of the formulae. As we have shown in Chapter 2, the tableau rules 'hide' a considerable amount of information which is contained in the input formulae and this is indeed the origin of their odd (and sometimes monstrous) computational behaviour: they hide all the information pertaining to the 'bivalent' structure of classical logic.

So our negative considerations raise a positive problem. Are there proof systems which can be considered as a *real* (uniform) improvement on the truth-table method (and on the tableau method)?

Let us say that a proof system is *standard* if its complexity is $O(n \cdot 2^k)$ where n is the length of the input formula and k the number of distinct variable occurring in it. It is not difficult to show that the analytic restrictions[15]

[15] By 'analytic restriction' of **KE** and **KI** we mean the systems obtained by restricting the applications of PB in a proof of A from Γ to subformulae of A or of formulae in Γ.

of both **KE** and **KI** are standard proof systems. This is obvious in the case of **KI**: as seen in Section 3.7 **KI** can be viewed as a *uniform* improvement of the truth-table method. For **KE** it will appear as a corollary of the fact that this system can linearly simulate **KI**, to be shown in the next section.

4.5 The relative complexity of KE and KI

In this section we compare the propositional fragments of the systems **KE** and **KI** with the propositional fragments of other well-known proof systems. All the proof systems we shall consider here enjoy the so-called (weak) *subformula property*, that is: to prove a formula A one only needs to consider its weak subformulae, where a weak snbformula of A is either a subformula of A or the negation of a subformula of A. We call a proof *analytic* if it enjoys this property. Some systems of deduction (like the tableau method and Gentzen's sequent calculus without cut) yield only analytic proofs. Others (like Natural Deduction, Gentzen's sequent calculus with cut, **KE** and **KI**) allow for a more general notiou of proof which includes non-analytic proofs, although in all these cases the systems obtained by restricting the rules to analytic applications are still complete. Since we are interested, for theoretical and practical reasons, in analytic proofs, we shall pay special attention to simulation procedures which preserve the subformula property.

Definition 4.5.1 The *length* of a proof π, denoted by $|\pi|$ is the total number of symbols occurring in π (intended as a string).

The *λ-complexity*, of π, denoted by $\lambda(\pi)$, is the *number of lines* in the proof π (each 'line' being a sequent, a formula, or any other expression associated with an inference step, depending on the system under consideration). Finally the *ρ-complexity* of π, denoted by $\rho(\pi)$ is the length (total number of symbols) of a line of maximal length occurring in π.

Our complexity measures are obviously connected by the relation

$$|\pi| \leq \lambda(\pi) \cdot \rho(\pi).$$

Now, observe that the λ-measure is sufficient to establish negative results about the simulation relation in 4.1, but is not sufficient in general for positive results. It may, however, be adequate also for positive results whenever one

can show that the ρ-measure (the length of lines) is not significantly increased by the simulation procedure under consideration. All the procedures that we shall consider in the sequel will be of this kind. So we shall forget about the ρ-measure and restrict our attention to the λ-measnre.

As said before, we are interested in the complexity not of proofs in general but of *analytic proofs*. We shall then appeal to the notion of *analytic restriction* of a system: let **S** be a system which enjoys the subformula property. We can take **S** as defined not by its inference rules, but extensionally, by the set of proofs which it recognizes as *sound*. Then we can denote by $\mathbf{S}^*$ its *analytic restriction*, i.e. the subset of **S** consisting of the proofs which enjoy the subformula property. All the notions defined for proof systems can be extended to analytic restrictions in a natural way.

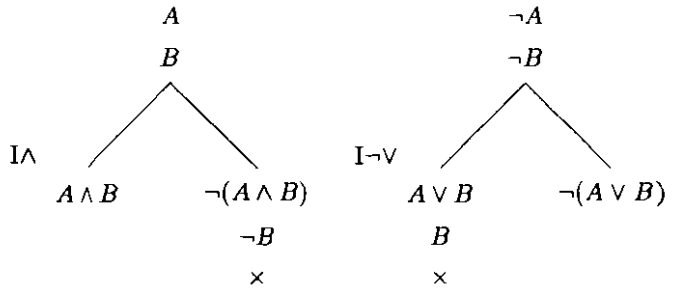
We shall consider the versions of **KE** and **KI** which use *unsigned* formulae.

4.5.1 KE versus KI

We start by showing that proofs in **KE** and **KI** have essentially the same complexity:

Theorem 4.5.1 **KE** *and* **KI** *can linearly simulate each other. Moreover, the simulation preserves the subformula property.*

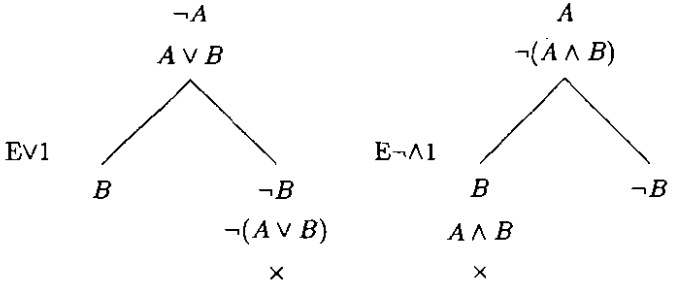
Proof. First, observe that the rules of **KI** can be simulated in **KE** as follows (we show the procedure only for the rules I$\wedge$ and I$\neg\vee$, the other cases being similar):



where $\times$ marks a closed branch.

Now, if $\mathcal{T}$ is a **KI**-tree with assumptions Γ, then replace each application of a **KI**-rule with its **KE**-simulation (the applications of PB can be left unchanged since PB is also a rule of **KE**). The result is a **KE**-tree $\mathcal{T}'$ containing at most $\lambda(\mathcal{T}) + c \cdot \lambda(\mathcal{T})$ nodes, where c is the maximum number of additional nodes generated by a **KE**-simulation of a **KI**-rule (namely 2). Thus, if $\mathcal{T}$ is a **KI**-proof of A from Γ, $\mathcal{T}'$ is a **KE**-tree such that all its open branches have A as terminal node. Moreover, $\lambda(\mathcal{T}') \leq 3\lambda(\mathcal{T})$ and $\mathcal{T}'$ does not contain any formulae which do not occur in $\mathcal{T}$. Simply adding $\neg A$ to the assumptions will provide a closed **KE**-tree for $\Gamma, \neg A$.

The inference rules of **KE** can be easily simulated in **KI** as follows (we show the procedure only for the rules E$\vee$1 and E$\neg\wedge$1, the other cases being similar):



Now, if $\mathcal{T}$ is a **KE**-tree with assumptions $\Gamma, \neg A$, then replace each application of a **KE**-rule with its **KI**-simulation (the applications of PB can be left unchanged since PB is also a rule of **KI**). The result is a **KI**-tree $\mathcal{T}'$ containing at most $\lambda(\mathcal{T}) + c \cdot \lambda(\mathcal{T})$ nodes, where c is the maximum number of additional nodes generated by a **KI**-simulation of a **KE**-rule (namely 2). Thus, if $\mathcal{T}$ is a closed **KE**-tree for $\Gamma, \neg A$, then $\mathcal{T}'$ is a closed **KI**-tree for $\Gamma, \neg A$. Moreover, $\lambda(\mathcal{T}') \leq 3\lambda(\mathcal{T})$ and $\mathcal{T}'$ does not contain any formula which does not occur in $\mathcal{T}$. Hence, by applying PB to A, one obtains the required **KI**-proof of A from Γ.□

Let us say that a **KI**-tree is atomic if PB is applied to atomic formulae only.

Proposition 4.5.1 *If A is a tautology of length n and containing k distinct variables, there is an* atomic **KI**-*proof $\mathcal{T}$ of A with $\lambda(\mathcal{T}) \leq n \cdot 2^k$.*

Proof. Trivial. (See above, Section 3.7.)

Theorem 4.5.2 **KE** *and* **KI** *are standard proof systems, i.e. for every tautology A of length n and containing k distinct variables, there is a* **KI**-*proof $\mathcal{T}$ of A and a* **KE**-*refutation $\mathcal{T}'$ of $\neg A$ with $\lambda(\mathcal{T}') = O(\lambda(\mathcal{T})) = O(n \cdot 2^k)$*

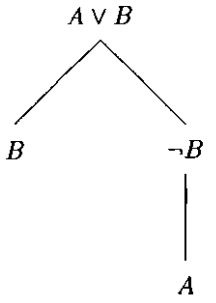
The theorem is obvious for **KI** (see above Section 3.7). For **KE** it is an immediate corollary of theorem 4.5.1.

4.5.2 KE versus the tableau method

First we notice that, given a tableau refutation $\mathcal{T}$ of Γ we can effectively construct an *analytic* **KE**-refutation $\mathcal{T}'$ of Γ which is not essentially longer.

Theorem 4.5.3 $\Gamma \overset{n}{\vdash}_{\mathbf{TM}} A \Longrightarrow \Gamma \overset{2n}{\vdash}_{\mathbf{KE}^-} A$

Proof. Observe that the connective rules of **KE**, combined with PB, can easily simulate the branching rules of the tableau method, as is shown below in the case of the branching rule for eliminating disjunctions (all the other cases are similar):



Such a simulation lengthens the original tableau by one node for each application of a branching rule. Since the linear rules of the tableau method are also rules of **KE**, it follows that there is a **KE**-refutation $\mathcal{T}'$ of Γ such that

$\lambda(\mathcal{T}') \leq \lambda(\mathcal{T}) + k$, where k is the number of applications of branching rules in $\mathcal{T}$. Since k is obviously $\leq \lambda(\mathcal{T})$, then $\lambda(\mathcal{T}') \leq 2\lambda(\mathcal{T})$.□

We have seen that in spite of their similarity, **KE** and the tableau method appear to be separated with respect to tbeir complexity. This fact is already suggested by Claim 4.4.1, and Theorems 4.5.2 and 4.5.3, and will later be proved as a consequence of a result in [CR74]. In accordance with Theorem 4.5.2, the 'hard examples' on which Claim 4.4.1 was based (see above Section 4.4) can be seen to have short **KE**-refutations. Figure 4.1 shows a **KE**-refutation of the set of clauses H_{P_1,P_2,P_3}. It is apparent that, in general, the number of branches in the **KE**-tree for $H_{P_1,\ldots,P_k}$, constructed according to the same pattern, is exactly 2^{k-1} (which is the number of clauses in the expression divided 2) and that the refutation trees have size $O(k \cdot 2^k)$.

While all the tableau rules can be easily simulated by means of **KE**-rules, **KE** includes a rule, namely PB, which cannot be easily simulated by means of the tableau rules. Although it is well-known that the addition of this rule to the tableau rules does not increase the stock of inferences that can be shown valid (since PB is classically valid and the tableau method is classically complete), its absence, in some cases, is responsible for an explosive growth in the size of tableau proofs. In chapter 2 we have given a semantic explanation of this combinatorial explosion. The syntactic counterpart to that semantic argument can be expressed in terms of 'analytic cut'. Suppose there is a tableau proof of A from Γ, i.e. a closed tableau $\mathcal{T}_1$ for $\Gamma, \neg A$ (where A is assumed to be atomic for the sake of simplicity) and a tableau proof of B from Δ, A, i.e. a closed tableau $\mathcal{T}_2$ for $\Delta, A, \neg B$; then it follows from the elimination theorem (see [Smu68a]) that there is also a closed tableau for $\Gamma, \Delta, \neg B$. This fact can be seen as a typical 'cut' inference:

$$\frac{\begin{array}{rcl}\Gamma & \vdash & A \\ \Delta, A & \vdash & B\end{array}}{\Gamma, \Delta \vdash B}$$

where '$\vdash$' stands for the tableau derivability relation. When a rule like PB is available, simulating this kind of 'cut' inference is relatively inexpensive in terms of proof size as is shown by the diagram below:

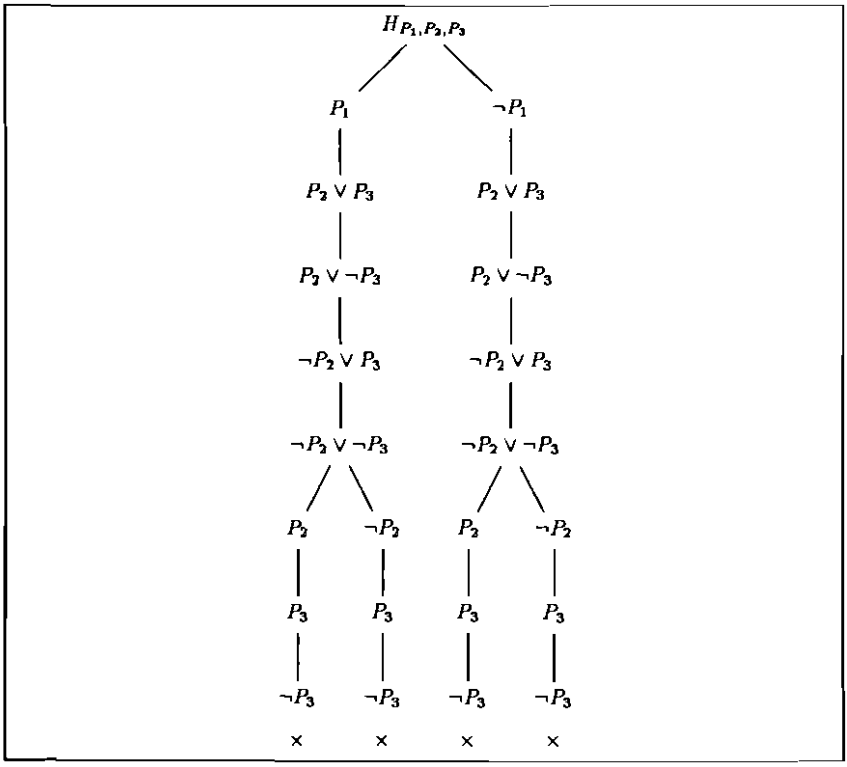
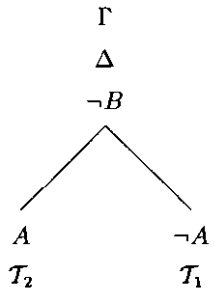
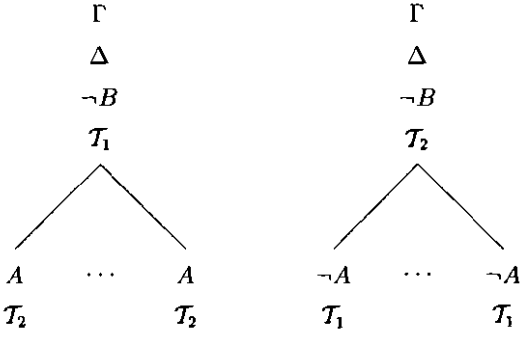


Figure 4.1: A **KE**-refutation of H_{P_1,P_2,P_3}.



But if PB is not in our stock of rules, reproducing the cut inference may be much more expensive. Let us assume, for instance, that $\neg A$ is used more than once, say n times, in $\mathcal{T}_1$ to close a branch, so that $\mathcal{T}_1$ contains n occurrences of A in *distinct* branches which will be left open if $\neg A$ is removed from the assumptions. Similarly, let A be used more than once, say m times, in $\mathcal{T}_2$ to close a branch, so that $\mathcal{T}_2$ contains m occurrences of $\neg A$ in *distinct* branches which will be left open if A is removed from the assumptions. Then, in some cases, the shortest tableau refutation of $\Gamma, \Delta, \neg B$ will have one of the following two forms:



where the subrefutation $\mathcal{T}_2$ is repeated n times in the lefthand tree and the subrefutation $\mathcal{T}_1$ is repeated m times in the righthand tree. The reader should notice that in this case the 'elimination of cuts' from the tableau proof does not remove 'impure' inferences, because A is assumed to be a snbformula of formulae in Γ, Δ. So the cut proof is analytic.

Because of this intrinsically inefficient way of dealing with analytic cut inferences, examples can be found which require a great deal of duplication in the construction of a closed tableau. We have already given a class of 'hard examples' for the tableau method which are easy not only for (the analytic restrictions of) **KE** and **KI** but also for the truth-table method. Another class of hard examples is described in [CR74]: Let

$$H_m = \{\pm A^1 \vee \pm A^2_{\pm} \vee \pm A^3_{\pm\pm} \vee \ldots \vee A^m_{\pm\ldots\pm}\}$$

where $+A$ means A and $-A$ means $\neg A$, and the snbscript of A^i is a string of $i-1$ $+$'s or $-$'s corresponding to the sequence of signs of the preceding A^j, $j < i$. Thus H_m contains 2^m disjunctions and $2^m - 1$ distinct atomic letters. For instance $H_2 = \{A^1 \vee A^2_+, A^1 \vee \neg A^2_+, \neg A^1 \vee A^2_-, \neg A^1 \vee \neg A^2_-\}$.

In [CR74] Cook and Rechow report without proof a lower bound of $2^{2^{cm}}$ on the number of nodes of a closed tableau for the conjunction of all disjunctions in H_m. Moreover, since there are 2^{m-1} distinct atomic letters, this class of examples is hard also for the truth-table method. In contrast, we can show that there is an 'easy' analytic **KE**-refntation of H_m which contains $2^m + 2^m m - 2$ nodes. Such a refutation has the following form: start with H_m. This will be a set containing $n(= 2^m)$ disjunctions of which $n/2$ start with A^1 and the remaining $n/2$ with its negation. Then apply PB to $\neg A^1$. This creates a branching with $\neg A^1$ in one branch and $\neg\neg A^1$ in the other. Now, on the first branch, by means of $n/2$ applications of the rule E$\vee$1 we obtain a set of formulae which is of the same form as H_{m-1}. Similarly on the second branch we obtain another set of the same form as H_{m-1}. By reiterating the same procedure we eventually produce a closed tree for the original set H_m. It is easy to see that the numher of nodes generated by the refutation can be calculated as follows (where n is the number of formulae in H_m, namely 2^m):

$$\begin{aligned}\lambda(T) = n + \sum_{i=1}^{\log n - 1} 2^i + n &= n + 2 \cdot \frac{1 - 2^{\log n - 1}}{1 - 2} + n \cdot (\log n - 1) \\ &= n + n \log n - 2.\end{aligned}$$

This is sufficient to establish:

Theorem 4.5.4 *The tableau method cannot p-simulate the analytic restriction of* **KE** *(and* **KI***).*

This result also shows that the truth-tables cannot p-simulate **KE** (and **KI**) in non-trivial cases[16]. Figure 4.2 shows the **KE**-refutation in the case $m = 3$. This class of examples also illustrates an interesting phenomenon: while the complexity of **KE**-refutations is not sensitive to the order in which the elimination rules are applied, it can be, in certain cases, highly sensitive to the choice of the PB formulae. If we make the 'wrong' choices, a combinatorial explosion may result when 'short' refutations are possible by making different choices. If, in Cook and Rechow's examples, the rule PB is applied always to the 'wrong' atomic variable, namely to the last one in each clause, it is not difficult to see that the size of the tree becomes exponential. To avoid this phenomenon an obvious criterion suggests itself from the study of this example: apply PB to a formula which has a large number of occurrences in the branch under consideration. So, as far as proof search is concerned, we need some further information about the composition of the formulae in the branch to guide our choice of the PB formulae. We expect that some variant of Bibel's connection method ([Bib82], see also [Wal90]) will prove useful in this context.

In any case, our discussion shows that analytic cuts are sometimes essential for the existence of short refutations *with the subformula property*[17].

4.5.3 KE versus Natural Deduction

It can also be shown that **KE** can linearly simulate natural deduction (in tree form). Moreover the simulation procedure preserves the subformula

[16]We mean that the exponential behaviour of the truth-tables in this case does not depend *only* on the large number of variables but also on the logical structure of the expressions. So these examples are essentially different from the examples which are usually employed in textbooks to show that the truth-tables are intractable (a favourite one is the sequence of expressions $A \vee \neg A$ where A contains an increasing number of variables).

[17]This can be taken as further evidence in support of Boolos' plea for not eliminating cut [Boo84]. In that paper he gives a natural example of a class of first order inference schemata which are 'hard' for the tableau method while admitting of 'easy' (*non-analytic*) natural deduction proofs. Boolos' example is a particularly clear illustration of the well-known fact that the elimination of cuts from proofs in a system in which cuts are eliminable can greatly increase the complexity of proofs. (For a related technical result see [Sta78].) **KE** and **KI** provide an elegant solution to Boolos' problem by making cut non-eliminable while preserving the subformula property of proofs. Our discussion also shows that eliminating *analytic* cuts can result in a combinatorial explosion.

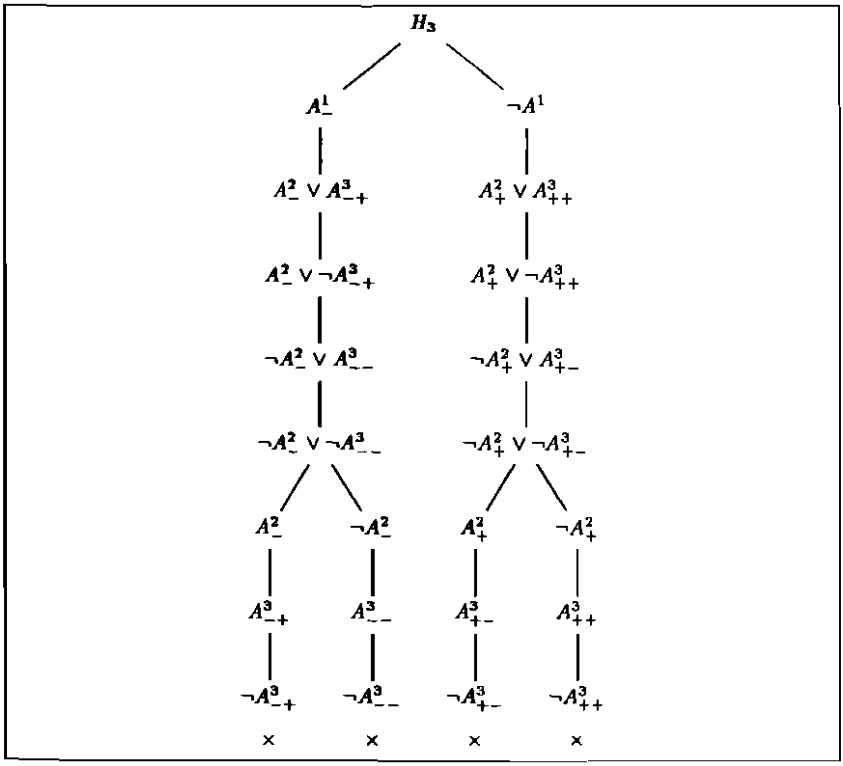


Figure 4.2: A **KE**-refutation of H_3.

property. We shall sketch this procedure for the natural deduction system given in [Pra65], the procedure being similar for other formulations.

We want to give an effective proof of the following theorem (where **ND** stands for Natural Deduction):

Theorem 4.5.5 *If there is an* **ND***-proof* $\mathcal{T}$ *of* A *from* Γ*, then there is a* **KE***-proof* $\mathcal{T}'$ *of* A *from* Γ *such that* $\lambda(\mathcal{T}') \leq 3\lambda(\mathcal{T})$ *and* $\mathcal{T}'$ *contains only formulae* A *such that* A *occurs in* $\mathcal{T}$*.*

Proof. By induction on $\lambda(\mathcal{T})$.
If $\lambda(\mathcal{T}) = 1$, then the **ND**-tree consists of only one node which is an assumption, say C. The corresponding **KE**-tree is the closed sequence $C, \neg C$.
If $\lambda(\mathcal{T}) = k$, with $k > 1$, then there are several cases depending on which rule has been applied in the last inference of $\mathcal{T}$. We shall consider only the cases in which the rule is elimination of conjunction (E$\wedge$) and elimination of disjunction (E$\vee$), and leave the others to the reader. If the last rule applied in $\mathcal{T}$ is E$\wedge$, then $\mathcal{T}$ has the form:

$$\mathcal{T} = \begin{array}{c} \Delta \\ \mathcal{T}_1 \\ A \wedge B \\ \hline A \end{array}$$

By inductive hypothesis there is a **KE**-refutation $\mathcal{T}_1'$ of $\Delta, \neg(A \wedge B)$ such that $\lambda(\mathcal{T}_1') \leq 3\lambda(\mathcal{T}_1)$. Then the following **KE**-tree:

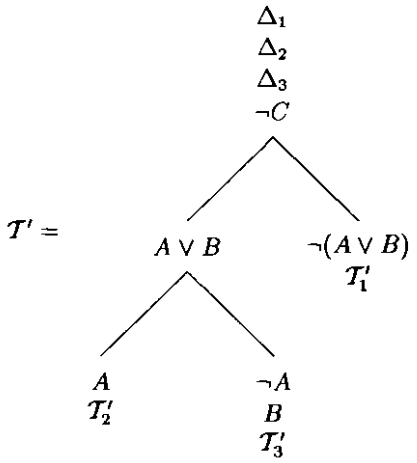
$$\mathcal{T}' = \begin{array}{ccc} & \Delta & \\ & \neg A & \\ & \swarrow \quad \searrow & \\ A \wedge B & & \neg(A \wedge B) \\ A & & \mathcal{T}_1' \\ \times & & \end{array}$$

is the required **KE**-proof and it is easy to verify that $\lambda(\mathcal{T}') \leq 3\lambda(\mathcal{T})$.

If the last rule applied in $\mathcal{T}$ is the rule of elimination of disjunction, then $\mathcal{T}$ has the form:

$$\mathcal{T} = \begin{array}{ccc} \Delta_1 & \Delta_2, [A] & \Delta_3, [B] \\ \mathcal{T}_1 & \mathcal{T}_2 & \mathcal{T}_3 \\ A \vee B & C & C \\ \hline & C & \end{array}$$

By inductive hypothesis there are **KE**-refutations $\mathcal{T}_1'$ of $\Delta_1, \neg(A \vee B)$, $\mathcal{T}_2'$ of $\Delta_2, A, \neg C$, and $\mathcal{T}_3'$ of $\Delta_3, B, \neg C$, such that $\lambda(\mathcal{T}_i') \leq 3\lambda(\mathcal{T}_i)$, $i = 1, 2, 3$. Then the following **KE**-tree:



is the required proof and it is easy to verify that $\lambda(\mathcal{T}') \leq 3\lambda(\mathcal{T})$.□

In [Cel88] Cellucci has proposed a new form of natural deduction which he has shown in several examples to produce shorter proofs than Prawitz's style natural deduction. In Cellucci's system, as in natural deduction, assumptions are introduced which may subsequently be discharged; however the inference rules involve sequents rather than single formulae. The rules of the propositional fragment are shown in table 4.1. (Δ and Λ represents sequences of

Structural rules

$$\frac{\Delta}{\Delta, A}\ [\text{Thin}] \qquad \frac{\Delta, A, B, \Lambda}{\Delta, B, A, \Lambda}\ [\text{Perm}] \qquad \frac{\Delta, A, A}{\Delta, A}\ [\text{Cont}]$$

Logical rules

$$\frac{\Delta, A \quad \Lambda, B}{\Delta, \Lambda, A \wedge B}\ [\wedge\text{I}] \qquad \frac{\Delta, A \wedge B}{\Delta, A}\ [\wedge\text{E-1}] \qquad \frac{\Delta, A \wedge B}{\Delta, B}\ [\wedge\text{E-2}]$$

$$\frac{\Delta, A, B}{\Delta, A \vee B}\ [\vee\text{I}] \qquad \frac{\Delta, A \vee B}{\Delta, A, B}\ [\vee\text{E}]$$

$$\frac{\begin{array}{c}[A] \\ \Delta, B\end{array}}{\Delta, A \rightarrow B}\ [\rightarrow\text{I}] \qquad \frac{\Delta, A \quad \Lambda, A \rightarrow B}{\Delta, \Lambda, B}\ [\rightarrow\text{E}]$$

$$\frac{\begin{array}{c}[A] \\ \Delta\end{array}}{\Delta, \neg A}\ [\neg\text{I}] \qquad \frac{\Delta, A \quad \Lambda, \neg A}{\Delta, \Lambda}\ [\neg\text{E}]$$

Table 4.1: A sequent-conclusion natural deduction system.

formulae, not sets. If they are taken to represent sets, the rules Perm and Cont become redundant.) According to Cellucci, derivations in his system are 'significantly simpler (i.e. contain fewer symbols) than in [Prawitz's style] system'[18]. It is not difficult to show, by means of an argument analogous to the one given above for Prawitz's style natural deduction, that:

Theorem 4.5.6 *If there is a proof π in Cellucci's system of the sequent Δ from assumptions Γ, then there is a* **KE***-refutation π' of $\Gamma, \overline{\Delta}$, where $\overline{\Delta}$ is equal to $\{\neg B \mid B \in \Delta\}$, such that $\lambda(\pi') \leq 3\lambda(\pi)$. Moreover, the simulation preserves the subformula property.*

[18][Cel88, p.29].

4.5.4 KE and resolution

The relationship between **KE** and resolution-like methods has been discussed in Section 3.4. We can easily adapt the **KE**-rules to deal with formulae in clausal form and, once this has been done, the system is seen to incorporate the basic idea underlying resolution theorem proving as a special case. As far as the relative complexity of **KE** and resolution is concerned, within the domain of clausal form logic, it depends on how refutations are represented. If propositional resolution is represented as in [DLL62], i.e. as a combination of unit resolution and the splitting rule, its complexity is the same as that of the clausal form fragment of **KE**. This way of representing resolution is much clearer and appears to be more convenient from a practical point of view, in that it seems to reduce the generation of redundant clauses. If the resolution rule is defined in the more usual way as a single rule, and refutations are represented as sequences or directed acyclic graphs of clauses, it is easy to see that **KE** can linearly simulate resolution with a procedure which involves only subformulae of the formulae occurring in the given resolution refutation. Such a refutation will be, strictly speaking, non-analytic because the resolution rule does not have the subformula property. An open question is whether or not the analytic restriction of **KE** can efficiently simulate the standard version of resolution. We conjecture that it can[19].

4.6 A more general view

As seen in section 4.5.2, the speed-up of **KE** over the tableau method can be traced to the fact that **KE**, unlike the tableau method, can easily simulate inference steps based on 'cuts', like the example on p.77. In other words, **KE** allows a uniform method for grafting proofs of subsidiary conclusions, or *lemmata*, in the proof of a theorem. However, the existence of such a uniform method is by no means sufficient. In Natural Deduction, for instance, replacing *every occurrence* of an assumption A with its proof provides an obvious grafting method which, though very perspicuous, is highly inefficient, leading to much unwanted duplication in the resulting proof-tree. We should then require that the method be also *efficient*. A way of making this requirement

[19] If resolution refutations are represented in tree form, it is not difficult to see that such a simulation is possible.

precise is to ask that the following condition be satisfied by a proof system S:

(C) Let π_1 be a proof of A from Γ and let π_2 be a proof of B from Δ, A, then there is a uniform method for constructing from π_1 and π_2 a proof π_3 of B from Γ, Δ such that $\lambda(\pi_3) \leq \lambda(\pi_1) + \lambda(\pi_2) + c$ for some constant c.

It can be easily seen that condition (C) is satisfied by **KE** and **KI**, the required method being the one described on p.77. In contrast, the standard way of grafting proofs of subsidiary conclusions iu Natural Deduction proofs, though providing a uniform method, does not satisfy the further condition on the complexity of the resulting proof. The rules of Natural Deduction, however, permit us to bypass this difficulty and produce a method satisfying the whole of condition (C). Consider the rule of *Non-Constructive Dilemma* (NCD):

$$\dfrac{\begin{array}{cc}\Gamma,[A] & \Delta,[\neg A] \\ \vdots & \vdots \\ B & B\end{array}}{B}$$

This is a derived rule in Prawitz's style Natural Deduction which yields classical logic if added to the intuitionistically valid rules (see [Ten78, section 4.5]). We can show Natural Deduction to satisfy condition (C) by means of the following construction:

$$\dfrac{\Gamma,[A] \quad \begin{array}{c}\Delta \\ \vdots \\ A\end{array} \quad [\neg A]}{\begin{array}{c}\vdots \\ B\end{array} \qquad \dfrac{F}{B}}{B}$$

Notice that the construction does not depend on the number of occurrences of the assumption A in the subproof of B from Γ, A.

In analogy to condition (C) we can formulate a condition requiring a proof system to simulate efficiently another form of cut which holds for classical systems and is closely related to the rule PB:

(C*) Let π_1 be a proof of B from Γ, A and π_2 be a proof of B from $\Delta, \neg A$. Then there is a uniform method for constructing from π_1 and π_2 a proof π_3 of B from Γ, Δ such that $\lambda(\pi_3) \leq \lambda(\pi_1) + \lambda(\pi_2) + c$ for some constant c.

Similarly, the next condition requires that proof systems can efficiently simulate the *ex falso* inference scheme:

(XF) Let π_1 be a proof of A from Γ and π_2 be a proof of $\neg A$ from Δ. Then there is a uniform method for constructing from π_1 and π_2 a proof π_3 of B from Γ, Δ, for any B, such that $\lambda(\pi_3) \leq \lambda(\pi_1) + \lambda(\pi_2) + c$ for some constant c.

So, let us say that a classical proof system is a *classical cut system* if it satisfies conditions (C*) and (XF). It is easy to show that every classical cut system satisfies also condition (C) above and, therefore, allows for an efficient implementation of the transitivity property of natural proofs. The next theorem shows that every classical cut system can simulate **KE** without a significant increase in proof complexity and with a procedure which preserves the subformula property[20]. This clearly shows that it is only the use of (analytic) *cut* and not the form of the *operational rules* which is crucial from a complexity viewpoint (this will be further discussed in chapter 6.).

Theorem 4.6.1 *If* **S** *is a classical cut system, then* **S** *can linearly simulate* **KE** *(and* **KI***) with a procedure which preserves the subformula property.*

To prove the theorem it is convenient to assume that onr language includes a 0-ary operator F (Falsum) and that the proof systems include suitable rules to deal with it[21]. For **KE** this involves only adding the obvious rule which allows us to append F to any branch containing both A and $\neg A$ for some formula A, so that every closed branch in a **KE**-tree ends with a node labelled with F. The assumption is made only for convenience's sake and can be dropped without consequence. Moreover we shall make the obvious assumption that, for every system **S**, the complexity of a proof of A from A in **S** is equal to 1.

[20] In this chapter we are restricting ourselves to analytic proof systems, i.e. proof systems which enjoy the subformula property.

[21] Systems which are not already defined over a language containing F can usually be redefined over such an extended language without difficulty.

Let $\tau(\pi)$ denote the number of nodes *generated* by a **KE**-refutation π of Γ (i.e. the assumptions are not counted)[22]. Let **S** be a classical cut system. If **S** is complete, then for every rule r of **KE** there is an **S**-proof π_r of the conclusion of r from its premises. Let $b_1 = Max_r(\lambda(\pi_r))$ and let b_2 and b_3 be the constants representing, respectively, the λ-cost of simulating classical cut in **S** — associated with condition (C*) above — and the λ-cost of simulating the *ex falso* inference scheme in **S** — associated with condition (XF) above. As mentioned before, every classical cut system satisfies also condition (C) and it is easy to verify that the constant associated witb this condition, representing the λ-cost of simulating 'absolute' cut in **S**, is $\leq b_2 + b_3 + 1$. We set $c = b_1 + b_2 + b_3 + 1$.

The theorem is an immediate consequence of the following Lemma:

Lemma 4.6.1 *For every classical cut system* **S**, *if there is a* **KE**-*refutation* π *of* Γ, *then there is an* **S**-*proof* π' *of* F *from* Γ *with* $\lambda(\pi') \leq c \cdot \tau(\pi)$.

Proof. The proof is by induction on $\tau(\pi)$, where π is a **KE**-refutation of Γ.

$\tau(\pi) = 1$. Then Γ is explicitly inconsistent, i.e. contains a pair of complementary formulae, say B and $\neg B$, and the only node generated by the refutation is F, which is obtained by means of an application of the **KE**-rule for F to B and $\neg B$. Since there is an **S**-proof of the **KE**-rule for F, we can obtain an **S**-proof π' of the particular application contained in π simply by performing the suitable substitutions and $\lambda(\pi') \leq b_1 < c$.

$\tau(\pi) > 1$. Case 1. The **KE**-refutation π has the form:

$$\begin{array}{c} \Gamma \\ C \\ \mathcal{T}_1 \end{array}$$

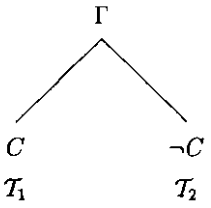
where C follows from premises in Γ by means of an E-rule. So there is a **KE**-refutation π_1 of Γ, C such that $\tau(\pi) = \tau(\pi_1) + 1$. By inductive hypothesis,

[22]The reader should be aware that our τ-measure applies to **KE**-refutations and *not* to trees: the same tree can represent different refutations yielding different values of the τ-measure.

there is an **S**-proof π_1' of F from Γ, C such that $\lambda(\pi_1') \leq c \cdot \tau(\pi_1)$. Moreover, there is an **S**-proof π_2 of C from the premises from which it is inferred in π such that $\lambda(\pi_2) \leq b_1$. So, from the hypothesis that **S** is a classical cut system, it follows that there is an **S**-proof π' of F from Γ such that

$$\begin{aligned} \lambda(\pi') &\leq c \cdot \tau(\pi_1) + b_1 + b_2 + b_3 + 1 \\ &\leq c \cdot \tau(\pi_1) + c \\ &\leq c \cdot (\tau(\pi_1) + 1) \\ &\leq c \cdot \tau(\pi) \end{aligned}$$

Case 2. π has the following form:



So there are **KE**-refutations π_1 and π_2 of Γ, C and $\Gamma, \neg C$ respectively such that $\tau(\pi) = \tau(\pi_1) + \tau(\pi_2) + 2$. Now, hy inductive hypothesis there is an **S**-proof π_1' of F from Γ, C and an **S**-proof π_2' of F from $\Gamma, \neg C$ with $\lambda(\pi_i') \leq c \cdot \tau(\pi_i), i = 1, 2$. Since **S** is a classical cut system, it follows that there is an **S**-proof π' of F from Γ such that

$$\begin{aligned} \lambda(\pi') &\leq c \cdot \tau(\pi_1) + c \cdot \tau(\pi_2) + b_2 \\ &< c \cdot \tau(\pi_1) + c \cdot \tau(\pi_2) + c \\ &< c \cdot (\tau(\pi_1) + \tau(\pi_2) + 2) \\ &< c \cdot \tau(\pi) \end{aligned}$$

□

It follows from Theorem 4.6.1 and Theorem 4.5.2 that:

Corollary 4.6.1 *Every analytic cut system is a standard proof system, i.e. for every tautology A of length n and containing k distinct variables there is a proof π, with $\lambda(\pi) = O(n \cdot 2^k)$.*

Moreover, Theorem 4.6.1, Theorem 4.5.3 and Theorem 4.5.4 imply that:

Corollary 4.6.2 *Every analytic cut system can linearly simulate the tableau method, but the tableau method cannot p-simulate any analytic cut system.*

Since (the classical version of) Prawitz's style natural deduction is a classical cut system, it follows that it can linearly simulate **KE**. The same holds also for Cellucci's system and for the 'natural deduction' variant of **KE** described in Section 3.8 under the name **KEND**. Moreover, for all these systems, the simulation preserves the subformula property (i.e. it maps analytic proofs to analytic proofs). Therefore Theorem 4.6.1, together with Theorems 4.5.1, 4.5.5 and 4.5.6, imply that Prawitz's style natural deduction, Cellucci's natural deduction, **KE**, **KEND** and **KI** can linearly simulate each other with a procedure which preserves the subformula property. Corollary 4.2 implies that all these systems are essentially more efficient than the tableau method, even if we restrict our attention to analytic proofs (the tableau method cannot p-simulate any analytic cut system). Finally Corollary 4.6.1 implies that, unlike the tableau method, all these systems are *standard* proof systems, i.e. have the same *upper bound* as the truth-table method.

Chapter 5

Belnap's four valued logic

5.1 Introduction

The study of first-degree entailment occupies a special position in the field of relevance logics: it can be seen either as the study of the validity of formulae of the form $A \rightarrow B$, where $\rightarrow$ is Andersou and Belnap's relevant implication and A, B are implication-free formulae, or as the study of the notion of *relevant deducibility* between standard formulae built-up from the usual connectives. In the latter interpretation it is associated with the problem — well-known to computer scientists who work in the area of Automated Deduction — of obtaining sound information from possibly inconsistent databases[1]. An interesting semantic characterization of first-degree entailment was given by Belnap in [Bel77] who also emphasized its connections with the problem of 'how a computer should think'[Bel76]. In [Dun76] Dunn presented a tableau system based on a modificatiou of Jeffrey's method of 'coupled trees' [Jef81]. In this chapter we study the consequence relation associated with Belnap's semantics and produce two different calculi which are sound and complete for it. Unlike Dunn's 'coupled tree' calculus, our calculi use *one* tree only. This simplification allows us to exploit fully the formal analogy with the corresponding classical calculi, and to obtain simple extensions to a first order version of Belnap's logic (ueither Jeffrey nor Dunn explain how the method of 'coupled trees' can be exteuded to deal with quantifiers).

[1] For recent contributions in this area and in the related one of logic programming, see [DCHLS90] and [BS89].

In sections 5.2 and 5.3 we briefly discuss the background problem and illustrate Belnap's semantics. In section 5.4 we formulate a tableau method which produces 'single' instead of 'coupled' tableaux, and prove it sound and complete for (the first-order version of) the consequence relation associated with Belnap's semantics. Then, in section 5.5, we define another calculus based on the classical systems **KE** which has been studied in Chapter 3.

5.2 'How a computer should think'

Deductive Reasoning is often described as a process of revealing 'hidden' information from explicit data and, as such, it is a basic *tool* in the area of 'intelligent' database management or question-answering systems. Unfortunately, the most time-honoured and well-developed framework for deductive reasoning — classical logic — is unsuitable to this purpose. The reason is that databases, especially large ones, have a great propensity to become *inconsistent*: first, the information stored is usually obtained from different sources which might conflict with each other; second, the information obtained from each source, even if it is not *obviously* inconsistent, may 'hide' contradictions. But it is well-known that classical two-valued logic is oversensitive to contradictions: if Γ is an inconsistent set of sentences, then — according to classical logic — *any sentence* follows from Γ. This does not imply, of course, that classical logic is *incorrect*, but only suggests that there are circumstances in which it is highly recommendable to abandon it and use another. A radical solution to this problem would be to require any database to be consistent *before* starting deductive processing. But, for practical reasons, this 'solution' is no better than the original problem: first, contradictions do not always lie on the surface, and the only way to detect such implicit contradictions is to apply deductive reasoning itself; second, even explicit contradictions may not be removable because they originate in conflicting data fed into the computer by different and equally reliable sources.

But if classical logic is not to be recommended for application in deductive database management, what kind of logic is the one with which a computer should 'think'?

5.3 Belnap's four-valued model

In this chapter we shall make use of the approach developed by Belnap [Bel77, Bel76] on the basis of a work of Dunn [Dun76]. Let **4** denote the set {**T**, **F**, **Both**, **None**}. The elements of **4** are called *truth-values*. Belnap calls them 'told values' to emphasize their epistemic character. We may think of them as the four possible ways in which an atomic sentence P can belong to the 'present state of information' : (1) the computer is told that P is true (and is not told that P is false); (2) the computer is told that P is false (and is not told that P is true); (3) the computer is told that P is both true and false (perhaps from different sources, or in different instants of time); (4) the computer is not told anything about the truth value of P.

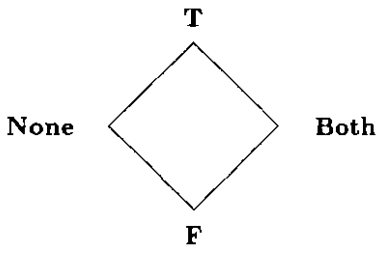
The values of complex sentences are obtained by Belnap by means of monotony considerations, based on Scott's approximation lattices, resulting in the following tables:

	None	F	T	Both
¬	None	T	F	Both

∧	None	F	T	Both
None	None	F	None	F
F	F	F	F	F
T	None	F	T	Both
Both	F	F	Both	Both

∨	None	F	T	Both
None	None	None	T	T
F	None	F	T	Both
T	T	T	T	T
Both	T	Both	T	Both

These tables represent a lattice which is called **L4**:



We define a *set-up* as a mapping of atomic formulae into **4**. Using the truth-tables given above, every set-up can be extended to a mapping of *all formulae* into **4** in the usual inductive way. We shall call such an extended mapping a **4**-*valuation* .

Definition 5.3.1 We say that A *entails* B, and write $A \rightarrow B$, if for all **4**-valuations v, $v(A) \preceq v(B)$, where $\preceq$ is the partial ordering associated with the lattice **L4**. We also say that a *non empty set* of formulae Γ *entails* A, and write $\Gamma \vdash A$, if the conjunction of all formulae in Γ entails A.

Notice that the relation $\vdash$ mentioned in Def. 5.3.1 is a *monotonic* consequence relation.

The logic characterized by Belnap's semantics corresponds to the logic of first-degree entailment (see [AB75, section 15.2]). This system admits of the following Hilbert-style formulation:

Axioms:

(5.1) $A \wedge B \rightarrow A$

(5.2) $A \wedge B \rightarrow B$

(5.3) $A \rightarrow A \vee B$

(5.4) $B \rightarrow A \vee B$

(5.5) $A \wedge (C \vee B) \rightarrow (A \wedge B) \vee C$

(5.6) $A \rightarrow \neg\neg A$

(5.7) $\neg\neg A \rightarrow A$

Rules:

$$(5.8) \qquad A \rightarrow B, B \rightarrow C \ \vdash \ A \rightarrow C$$

$$(5.9) \qquad A \rightarrow B, A \rightarrow C \ \vdash \ A \rightarrow B \wedge C$$

$$(5.10) \qquad A \rightarrow C, B \rightarrow C \ \vdash \ A \vee B \rightarrow C$$

$$(5.11) \qquad A \rightarrow B \ \vdash \ \neg B \rightarrow \neg A$$

Let us now introduce some useful terminology:

Definition 5.3.2 Let us say, given a **4**-valuation v, that a formula A is:

1. *at least true* under v if $v(A) = \mathbf{T}$ or $v(A) = \mathbf{Both}$.
2. *non-true* under v if $v(A) = \mathbf{F}$ or $v(A) = \mathbf{None}$.
3. *at least false* under v if $v(A) = \mathbf{F}$ or $v(A) = \mathbf{Both}$.
4. *non-false* under v if $v(A) = \mathbf{T}$ or $v(A) = \mathbf{None}$.

It is not difficult to see that the defiuition of $\Gamma \vdash A$ given in Definition 5.3.1 is equivalent to the following one:

Definition 5.3.3 $\Gamma \vdash A$ if and only if for every **4**-valuation v, (i) if all the elements of Γ are at least true under v, then A is at least true under v *and* (ii) if all the elements of Γ are non-false under v, then A is non-false under v.

5.4 Semantic tableaux for Belnap's logic

In this section we suggest a tableau method which, unlike Dunu's one, uses only one tree. The resulting tableaux are *binary* trees and are identical to classical tableaux of *signed* formulae except that they contain four types of s-formulae instead of two.

5.4.1 Propositional tableaux

Signed Formulae. We introduce the symbols t, f, t*, f* and define a *signed* formula as an expression of the form t(A), f(A), t*(A) or f*(A), where A is

$\frac{\mathrm{t}(A \wedge B)}{\mathrm{t}(A)\ \mathrm{t}(B)}$	$\frac{\mathrm{t}^*(A \wedge B)}{\mathrm{t}^*(A)\ \mathrm{t}^*(B)}$	$\frac{\mathrm{f}(A \vee B)}{\mathrm{f}(A)\ \mathrm{f}(B)}$	$\frac{\mathrm{f}^*(A \vee B)}{\mathrm{f}^*(A)\ \mathrm{f}^*(B)}$
$\frac{\mathrm{f}(A \wedge B)}{\mathrm{f}(A) \mid \mathrm{f}(B)}$	$\frac{\mathrm{f}^*(A \wedge B)}{\mathrm{f}^*(A) \mid \mathrm{f}^*(B)}$	$\frac{\mathrm{t}(A \vee B)}{\mathrm{t}(A) \mid \mathrm{t}(B)}$	$\frac{\mathrm{t}^*(A \vee B)}{\mathrm{t}^*(A) \mid \mathrm{t}^*(B)}$
$\frac{\mathrm{t}(\neg A)}{\mathrm{f}^*(A)}$	$\frac{\mathrm{t}^*(\neg A)}{\mathrm{f}(A)}$	$\frac{\mathrm{f}(\neg A)}{\mathrm{t}^*(A)}$	$\frac{\mathrm{f}^*(\neg A)}{\mathrm{t}(A)}$

Table 5.1: Propositional tableau rules for Belnap's four valued logic.

an unsigned formula. Intuitively we interpret '$\mathrm{t}(A)$' as 'A is at least true', '$\mathrm{f}(A)$' as 'A is non-true', '$\mathrm{t}^*(A)$' as 'A is non-false' and '$\mathrm{f}^*(A)$' as 'A is at least false'.

The *conjugate* of an s-formula $\mathrm{s}(A)$ is:

$\mathrm{f}(A)$ if $\mathrm{s} = \mathrm{t}$
$\mathrm{t}(A)$ if $\mathrm{s} = \mathrm{f}$
$\mathrm{f}^*(A)$ if $\mathrm{s} = \mathrm{t}^*$
$\mathrm{t}^*(A)$ if $\mathrm{s} = \mathrm{f}^*$

The *converse* of an s-formula $\mathrm{s}(A)$ is:

$\mathrm{t}^*(A)$ if $\mathrm{s} = \mathrm{t}$
$\mathrm{t}(A)$ if $\mathrm{s} = \mathrm{t}^*$
$\mathrm{f}^*(A)$ if $\mathrm{s} = \mathrm{f}$
$\mathrm{f}(A)$ if $\mathrm{s} = \mathrm{f}^*$

We are now in a position to formulate our tableau rules which are given in Table 5.1. The reader will notice the formal analogy with the classical rules.

It may be useful to extend Smullyan's unifying notation to cover the new types of s-formulae which occur in our language.

We use the letter 'α' to stand for any s-formula of one of the forms: $\mathrm{t}(A \wedge B)$, $\mathrm{t}^*(A \wedge B)$, $\mathrm{f}(A \vee B)$, $\mathrm{f}^*(A \vee B)$, $\mathrm{t}(\neg A)$, $\mathrm{t}^*(\neg A)$, $\mathrm{f}(\neg A)$ and $\mathrm{f}^*(\neg A)$. For every such formula α, its *components* α_1 and α_2 are defined as in Table 5.2. We use 'β' to stand for any formula of one of the forms: $\mathrm{f}(A \wedge B)$, $\mathrm{f}^*(A \wedge B)$, $\mathrm{t}(A \vee B)$ or $\mathrm{t}^*(A \vee B)$. For every such formula β, its *components* β_1 and β_2

α	α_1	α_2
$t(A \wedge B)$	$t(A)$	$t(B)$
$t^*(A \wedge B)$	$t^*(A)$	$t^*(B)$
$f(A \vee B)$	$f(A)$	$f(B)$
$f^*(A \vee B)$	$f^*(A)$	$f^*(B)$
$t(\neg A)$	$f^*(A)$	$f^*(A)$
$t^*(\neg A)$	$f(A)$	$f(A)$
$f(\neg A)$	$t^*(A)$	$t^*(A)$
$f^*(\neg A)$	$t(A)$	$t(A)$

Table 5.2: Formulae of type α.

β	β_1	β_2
$f(A \wedge B)$	$f(A)$	$f(B)$
$f^*(A \wedge B)$	$f^*(A)$	$f^*(B)$
$t(A \vee B)$	$t(A)$	$t(B)$
$t^*(A \vee B)$	$t^*(A)$	$t^*(B)$

Table 5.3: Formulae of type β.

are defined as in the Table 5.3. So our tableau rules can be 'packed' into the following two rules:

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

We say that a branch ϕ of a tableau is *closed* if it contains both an s-formula and its conjugate. Otherwise we say that ϕ is *open*. A tableau is *closed* if all its branches are closed. Otherwise it is *open*. A formula A is *provable* from the set of formulae Γ if and only if there is a closed tableau for $\{t(B) | B \in \Gamma\} \cup \{f(A)\}$.

An example of a closed tableau representing a proof of $((A \vee B) \wedge \neg A) \rightarrow$

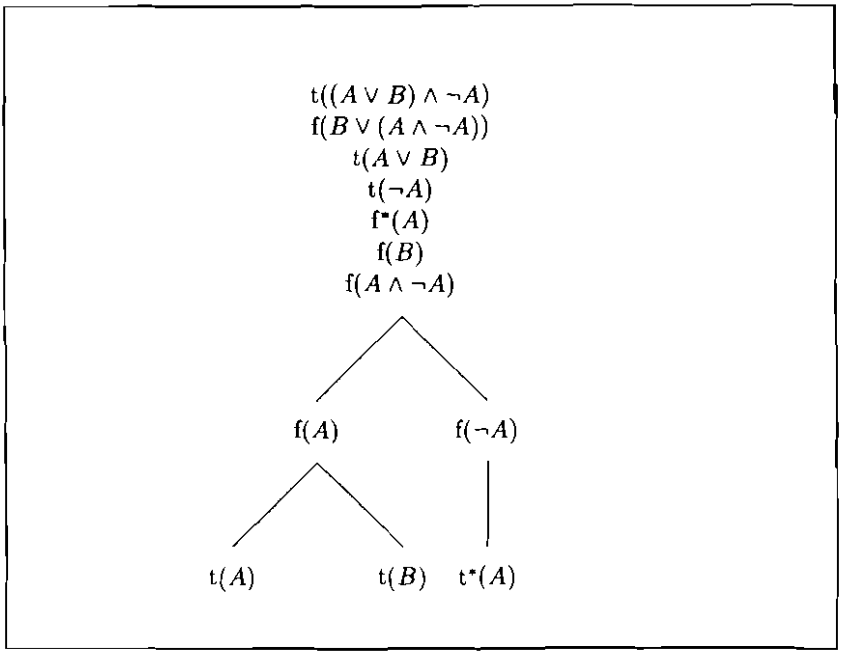


Figure 5.1: A proof of $((A \vee B) \wedge \neg A) \rightarrow (B \vee (A \wedge \neg A))$

$(B \vee (A \wedge \neg A))$ is given in Fig. 5.1. A failed attempt at proving the disjunctive syllogism is shown in Fig. 5.2.

Soundness.

Definition 5.4.1 Let us say that a **4**-valuation v *realizes* $s(A)$ if

1. $s(A) = t(A)$ and A is at least true under v.
2. $s(A) = f(A)$ and A is non-true under v.
3. $s(A) = t^*(A)$ and A is non-false under v
4. $s(A) = f^*(A)$ and A is at least false under v.

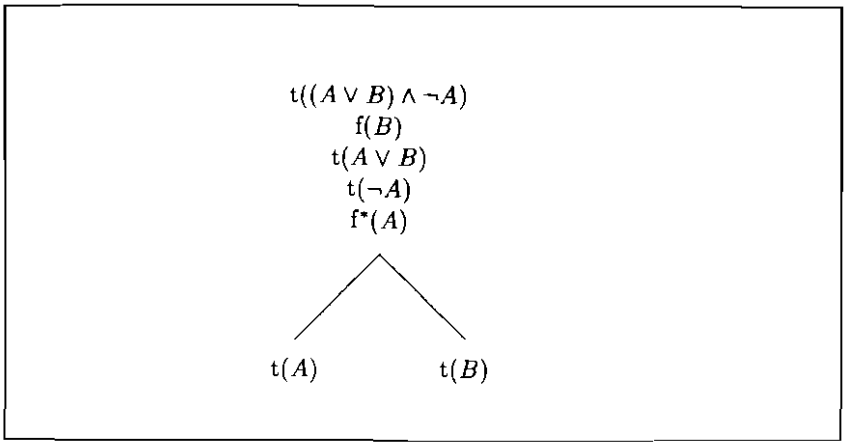


Figure 5.2: A failed attempt to prove the disjunctive syllogism

A set U of s-formulae is said to be *realizable* if there is a 4-valuation v which realizes every element of U.

In order to prove soundness we need the following lemma:

Lemma 5.4.1 *If $\mathcal{T}$ is a closed tableau for U, where U is a set of signed formulae, then the tableau $\mathcal{T}'$ obtained from $\mathcal{T}$ by replacing every (occurrence of an) s-formula with (an occurrence of) its converse is a closed tableau for the set U^* of the* converses *of the s-formulae in U.*

The proof is a straightforward induction on the number of nodes in a closed tableau.

Now, it is easy to verify that our tableau rules are *correct* in the sense that every 4-valuation which realizes the premise of the rule A realizes also both the conclusions of the rule, and every 4-valuation which realizes the premise of the rule B realizes also at least one of the conclusions of the rule. Therefore it follows, by an elementary inductive argument, that if a 4-valuation v realizes all the initial s-formulae of a tableau $\mathcal{T}$, then there is at least one branch ϕ of $\mathcal{T}$ such that v realizes all the s-formulae occurring

in ϕ. But, of course, no **4**-valuation can realize two conjugate s-formulae simultaneously. Therefore if $\mathcal{T}$ is a a closed tableau, no **4**-valuation can realize all tbe initial s-formulae of $\mathcal{T}$. So, if $\mathcal{T}$ is a closed tableau for $\{\mathrm{t}(B)|B \in \Gamma\} \cup \{\mathrm{f}(A)\}$, it follows that for every **4**-valuation v, A is at least true in v whenever all formulae in Γ are. Moreover it follows from lemma 5.4.1 that no **4**-valuation can realize the *converses* of all the initial signed formulae of $\mathcal{T}$, i.e. no **4**-valuation can realize $\{\mathrm{t}^*(B)|B \in \Gamma\} \cup \{\mathrm{f}^*(A)\}$. Hence for every **4**-valuation v, A is non-false in v whenever all the formulae in Γ are. So, by def. 5.3.3, $\Gamma \vdash A$. This concludes the proof.

Completeness. Let us say that a branch ϕ of a tableau $\mathcal{T}$ is *eomplete* if (i) for every α in ϕ both α_1 and α_2 occur in ϕ and (ii) for every β in ϕ at least one of β_1, β_2 occurs in ϕ. Let us also say that a tableau $\mathcal{T}$ is *completed* when every branch of $\mathcal{T}$ is complete. We have the following theorem:

Theorem 5.4.1 *Every complete* open *branch of any tableau is realizable.*

We shall first define the analog of Hintikka sets within our framework. The theorem will then immediately follow from the analog of Hintikka's lemma.

Definition 5.4.2 Let us say that a set of *signed* formulae U is an *R-Hintikka set* if and only if it satisfies the following conditions:

H_0: No signed variable and its conjugate are both in U.

H_1: If $\alpha \in U$, then $\alpha_1 \in U$ and $\alpha_2 \in U$.

H_2: If $\beta \in U$, then $\beta_1 \in U$ or $\beta_2 \in U$.

It follows from our definitions that the set of s-formulae in a complete *open branch* of any tableau is an R-Hintikka set. Then the theorem is an immediate consequence of the following lemma:

Lemma 5.4.2 *Every R-Hintikka set is realizable.*

Proof. Let U be an R-Hintikkka set. Let ns assign to each variable P which occurs in at least an element of U a value in **4** as follows:

(1) If $\mathrm{t}(P) \in U$, and $\mathrm{f}^*(P) \notin U$, give P the value **T**.
(2) If $\mathrm{t}(P) \in U$, and $\mathrm{f}^*(P) \in U$, give P the value **Both**.

(3) If $\mathrm{f}(P) \in U$, and $\mathrm{t}^*(P) \notin U$, give P the value **F**.
(4) If $\mathrm{f}(P) \in U$, and $\mathrm{t}^*(P) \in U$, give P the value **None**.
(5) if $\mathrm{t}^*(P) \in U$, and $\mathrm{f}(P) \notin U$, give P the value **T**.
(6) if $\mathrm{f}^*(P) \in U$, and $\mathrm{t}(P) \notin U$, give P the value **F**.

Now it is obvious that the **4**-valuation induced by this assignment realizes all the signed variables occurring in U. Then we only need to observe that

(i) If a **4**-valuation realizes both α_1 and α_2, then it realizes also α.

(ii) If a 4-valuation realizes at least one of β_1,β_2 then it realizes also β.

The lemma then follows from an easy induction on the complexity of the s-formulae in U.

Theorem 5.4.1 implies

Theorem 5.4.2 (Completeness Theorem) *If* $\Gamma \vdash A$ *then there is a closed tableau for* $\{\mathrm{t}(B) | B \in \Gamma\} \cup \{\mathrm{f}(A)\}$.

5.4.2 Detecting inconsistencies

Although we do not want inconsistencies to have the catastrophic effect that they have in classical logic, it would be desirable to be able to check our databases for consistency. After all, we do not want to 'glorify' contradictions. On the contrary we usually want to remove them.

Fortunately, consistency checks are quite easy to carry out without modifying our framework. We shall show that, in fact, our tableaux 'contain' their classical version: to obtain classical tableaux we only need to modify the closure condition on a branch.

Definition 5.4.3 The *weak conjugate* of an s-formula is defined as in the following table:

Signed formula	Weak conjugate
$\mathrm{t}(A)$	$\mathrm{f}^*(A)$
$\mathrm{f}(A)$	$\mathrm{t}^*(A)$
$\mathrm{t}^*(A)$	$\mathrm{f}(A)$
$\mathrm{f}^*(A)$	$\mathrm{t}(A)$

We say that a branch ϕ of a tableau $\mathcal{T}$ is *weakly closed* if ϕ contains an s-formula and its *weak conjugate*. The tableau $\mathcal{T}$ is *weakly closed* if all its branches are weakly closed.

Let us use s as a variable ranging over *signs*, i.e. over the symbols $\{\mathrm{t}, \mathrm{f}, \mathrm{t}^*, \mathrm{f}^*\}$. By $\overline{s}$ we mean:

f if $s = \mathrm{t}$
t if $s = \mathrm{f}$
f^* if $s = \mathrm{t}^*$
t^* if $s = \mathrm{f}^*$

By s^* se mean:

t^* if $s = \mathrm{t}$
f^* if $s = \mathrm{f}$
t if $s = \mathrm{t}^*$
f if $s = \mathrm{f}^*$

We have the following lemma:

Lemma 5.4.3 *Let us say that a set of s-formulae S is* homogeneous *if all the s-formulae in it have the same sign s. Then if S is a homogeneous set of s-formulae, every tableau $\mathcal{T}$ for S contains only s-formulae with sign s or $\overline{s^*}$.*

Proof: By inspection of the rules and induction on the rank of the s-formulae occurring in $\mathcal{T}$.

The following theorem states the connection between classical and relevant derivability:

Theorem 5.4.3 *Γ is a (classically) unsatisfiable set of formulae if and only if there is a weakly closed tableau for $\{\mathrm{t}(B)|B \in \Gamma\}$. Therefore A is classically deducible from Γ if and only if there is a weakly closed tableau for $\{\mathrm{t}(B)|B \in \Gamma\} \cup \{\mathrm{t}(\neg A)\}$.*

Proof: If Γ is classically unsatisfiable, there is a closed classical tableau for $\{\mathrm{t}(B)|B \in \Gamma\}$. It is easy to see that the tableau obtained from it by replacing each sign f with f^* is a weakly closed tableau for $\{\mathrm{t}(B)|B \in \Gamma\}$.

Suppose there is a weakly-closed tableau $\mathcal{T}$ for $\{t(B)|B \in \Gamma\}$. By Lemma 5.4.3 this tableau can contain only formulae signed with t or f*. Therefore every weakly-closed brauch contains $t(A)$ and $f^*(A)$ for some A. It follows that every **4**-valuation which realizes the initial formulae, must realize both $t(A)$ and $f^*(A)$ for some A, i.e. it must assign **Both** to A for some A. Hence there is no boolean valuation which satisfies Γ.

As we said before, the notion of weakly closed tableau is useful in that a weakly closed tableau shows an inconsistency in our database and, therefore, the need for revision. So our tableaux are sensitive to contradictions like classical tableaux; unlike classical tableaux, however, when they detect a contradiction they do not panic but keep on making 'sensible' deductions.

We observe that, in our approach, an initial formula of the form $f(A)$ always represents a 'query', whereas data are represented by initial formulae of the form $t(A)$. So, checking our data for consistency aud answering questions on the base of our data are *formally* distiuct tasks.

5.4.3 First-order tableaux

Another advantage of our single-tableau formulation is in the treatment of quantifiers. These are dealt with by means of rules obtained, as in the case of the binary connective rules, by taking over the standard classical rules (for s-formulae) and supplementing them with their 'starred' versions. By contrast, the suitable quantifier rules for the first-order version of the method of coupled-trees are not so straightforward.

We consider a standard first-order language with no functional symbols. We use the letters $x, y, z, \ldots$ (possibly with subscripts) as individual *variables* and the letters $a, b, c, \ldots$ (possibly with subscripts) as *parameters*. For any variable x and parameter a, $A(x/a)$ will be the result of substituting all the free occurrences of x in A with a. *Subformulae* are defined in the usual way, so that for every parameter a, $A(x/a)$ is a subformula of $\forall x A(x)$.

An obvious way of extending Belnap's semantics to formulae containing quantifiers is the following:

Let **U** be a non-empty universe. We assume a set U of *constants* naming the elements of **U** (of course neither **U** nor U need to be denumerable). By a U-formula we mean a formula built up from the logical operators, the relation symbols, the individual variables and the constants in U. (Thus a U-

formula is like a formula with parameters but with elements of U in place of parameters.) Let F^U be the set of *closed* U-formulae. A *first-order valuation over* U is defined as a mapping v of all elements of F^U into **4** such that (i) v is a **4**-valuation (see above p. 93) and (ii) v satisfies the following additional conditions for quantifiers (where $\sqcap V$ and $\sqcup V$ denote, respectively, the g.l.b. and the l.u.b. of the set V in **L4**):

$$\begin{aligned} v(\forall x(A(x)) &= \sqcap\{v(A(k/x))|k \in U\} \\ v(\exists x(A(x)) &= \sqcup\{v(A(k/x))|k \in U\} \end{aligned}$$

The first-order consequence relation associated with this extended semantics is then:

Definition 5.4.4 $\Gamma \vdash A$ if and only if for every universe U and all first-order valuations v over U, (i) if all elements of Γ are at least true under v, then A is at least true under v *and* (ii) if all the elements of Γ are non-false under v, then A is non-false under v.

The quantifier rules are the expected ones:

$$\frac{\mathrm{t}(\forall x A(x))}{\mathrm{t}(A(a/x))} \quad \text{for } \textit{all } a \quad \frac{\mathrm{t}^*(\forall x A(x))}{\mathrm{t}^*(A(a/x))}$$

$$\frac{\mathrm{f}(\forall x A(x))}{\mathrm{f}(A(a/x))} \quad \text{with } a \textit{ new} \quad \frac{\mathrm{f}^*(\forall x A(x))}{\mathrm{f}^*(A(a/x))}$$

$$\frac{\mathrm{t}(\exists x A(x))}{\mathrm{t}(A(a/x))} \quad \text{with } a \textit{ new} \quad \frac{\mathrm{t}^*(\exists x A(x))}{\mathrm{t}^*(A(a/x))}$$

$$\frac{\mathrm{f}(\exists x A(x))}{\mathrm{f}(A(a/x))} \quad \text{for } \textit{all } a \quad \frac{\mathrm{f}^*(\exists x A(x))}{\mathrm{f}^*(A(a/x))}$$

It is convenient to use Smullyan's notation for quantified formulae. So γ will denote any formula of one of the four forms $\mathrm{t}(\forall x A(x))$, $\mathrm{t}^*(\forall x A(x))$, $\mathrm{f}(\exists x A(x))$, $\mathrm{f}^*(\exists x A(x))$ and by $\gamma(a)$ we shall mean, respectively, $\mathrm{t}(A(x/a))$, $\mathrm{t}^*(A(x/a))$, $\mathrm{f}(A(x/a))$, $\mathrm{f}^*(A(x/a))$. Similarly δ will denote any formula of one of the four forms $\mathrm{t}(\exists x A(x))$, $\mathrm{t}^*(\exists x A(x))$, $\mathrm{f}(\forall x A(x))$, $\mathrm{f}^*(\forall x A(x))$ and and by $\delta(a)$ we shall mean, respectively, $\mathrm{t}(A(x/a))$, $\mathrm{t}^*(A(x/a))$, $\mathrm{f}(A(x/a))$, $\mathrm{f}^*(A(x/a))$.

Thus, our first-order rules can be succintly expressed by the following two rules:

Rule C $\dfrac{\gamma}{\gamma(a)}$ for *any* parameter a

Rule D $\dfrac{\delta}{\delta(a)}$ for a *new* parameter a

Soundness and completeness of this first-order system can be proved — given the corresponding proofs for the propositional fragment — by means of straightforward adaptations of the standard proofs for classical tableaux (as given in [Smu68a]). In fact, in our framework, quantifiers do not involve any non-standard notion which does not arise already at the propositional level. For instance, in order to prove completeness, we define the first-order analog of a complete tableau (where, of course, a complete branch may be infinite) and a procedure which generates a complete tableau for a set U of s-formulae. We can then define first-order R-Hintikka sets (over a universe $\mathbf{U}$) by adding to the clauses in definition 5.4.2 the following two clauses:

H_3: If $\gamma \in U$, then for *every* k in U, $\gamma(k) \in U$.

H_4: If $\delta \in U$, then for *some* k in U, $\delta(k) \in U$.

It is easy to see that the set of s-formulae occurring in every *open* branch of a complete tableau is an R-Hintikka set. Then, completeness follows from the lemma:

Lemma 5.4.4 *Every R-Hintikka set for a universe* $\mathbf{U}$ *is realizable.*

The previous discussion about the completeness of the first-order systems shows the *heuristic* advantage of this approach. The proofs of the analogs of most of the theorems which hold for the classical version can simply be *carried over* to the non-standard version with minor modifications. In fact this is true of most of the theorems included in [Smu68a]. We just mention here that the proofs of the compactness theorem and of the analog of Lowenheim-Skolem theorem require virtually no modification. Moreover, any implementation of a classical tableau-based theorem prover can be easily adapted to our framework, so providing a theorem prover for Belnap's logic *and* classical logic (via the notion of 'weak closure') simultaneously. A 'naive' implementation in Prolog, adapted from a program by Fitting, is given in [Gor90]

$\mathrm{t}(A \wedge B)$	$\mathrm{t}^*(A \wedge B)$	$\mathrm{f}(A \wedge B)$	$\mathrm{f}(A \wedge B)$
$\mathrm{t}(A)$	$\mathrm{t}^*(A)$	$\mathrm{t}(A)$	$\mathrm{t}(B)$
$\mathrm{t}(B)$	$\mathrm{t}^*(B)$	$\mathrm{f}(B)$	$\mathrm{f}(A)$
$\mathrm{f}^*(A \wedge B)$	$\mathrm{f}^*(A \wedge B)$	$\mathrm{f}(A \vee B)$	$\mathrm{f}^*(A \vee B)$
$\mathrm{t}^*(A)$	$\mathrm{t}^*(B)$	$\mathrm{f}(A)$	$\mathrm{f}^*(A)$
$\mathrm{f}^*(B)$	$\mathrm{f}^*(A)$	$\mathrm{f}(B)$	$\mathrm{f}^*(B)$
$\mathrm{t}(A \vee B)$	$\mathrm{t}(A \vee B)$	$\mathrm{t}^*(A \vee B)$	$\mathrm{t}^*(A \vee B)$
$\mathrm{f}(A)$	$\mathrm{f}(B)$	$\mathrm{f}^*(A)$	$\mathrm{f}^*(B)$
$\mathrm{t}(B)$	$\mathrm{t}(A)$	$\mathrm{t}^*(A)$	$\mathrm{t}^*(A)$
$\mathrm{t}(\neg A)$	$\mathrm{f}(\neg A)$	$\mathrm{t}^*(\neg A)$	$\mathrm{f}^*(\neg A)$
$\mathrm{f}^*(A)$	$\mathrm{t}^*(A)$	$\mathrm{f}(A)$	$\mathrm{t}(A)$
$\mathrm{t}(A) \mid \mathrm{f}(A)$	$\mathrm{t}^*(A) \mid \mathrm{f}^*(A)$		

Table 5.4: The rules of $\mathbf{RE}_{\mathrm{fde}}$.

5.5 An efficient alternative

Another simple tree method for first-degree entailment is obtained by adapting the rules of the system **KE** (see above, Chapter 3). The system so obtained will be baptised $\mathbf{RE}_{\mathrm{fde}}$. The propositional rules of $\mathbf{RE}_{\mathrm{fde}}$ are given in Table 5.4. The quantifier rules are the same as the rules for the tableau method given in the previous section.

Again, the logical rules can be expressed in a succint form by means of our extended use of Smullyan's notation (where β_i', $i = 1, 2$ denotes the *conjugate* of β_i):

Rule A $\dfrac{\alpha}{\begin{array}{l}\alpha_1\\ \alpha_2\end{array}}$

Rule B1 $\dfrac{\beta \quad \beta_1'}{\beta_2}$ Rule B2 $\dfrac{\beta \quad \beta_2'}{\beta_1}$

Rule C $\dfrac{\gamma}{\gamma(a)}$ for *any* parameter a

Rule D $\dfrac{\delta}{\delta(a)}$ for a *new* parameter a

In each application of the rules, the s-formulae α, β, γ and δ are called *major premises*. In each application of rules B1 and B2 the s-formulae β_i', $i = 1, 2$ are called *minor premises* (rules A, C and D have no minor premises).

In the system $\mathbf{RE}_{\mathrm{fde}}$ all the logical rules have a linear format. However, they are not sufficient for completeness: our 'cut-rules', PB and PB*, are *not* eliminable.

Definition 5.5.1 An $\mathbf{RE}_{\mathrm{fde}}$*-tree for* U, where U is a set of s-formulae, is a tree of s-formulae constructed in accordance with the rules above starting from s-formulae in U. A branch of an $\mathbf{RE}_{\mathrm{fde}}$-tree is *closed* when it contains a signed formula and its conjugate. Otherwise it is *open*. The tree itself is said to be *closed* when all its branches are closed.

An s-formula occurring in a $\mathbf{RE}_{\mathrm{fde}}$-tree is said to be *analysed* if it has been used at least once as *major* premise of one of the rules.

In Fig. 5.3 we give an $\mathbf{RE}_{\mathrm{fde}}$-proof of the same example of which a tableau proof was given in Fig. 5.1. The reader can compare the structure of proofs in the two methods. Notice that the $\mathbf{RE}_{\mathrm{fde}}$-tree contains no branching. Indeed, the system $\mathbf{RE}_{\mathrm{fde}}$ is more efficient than the tableau method formulated in the previous section for much the same reason as the classical system **KE** is more efficient than the classical tableau method (see Chapters 2 and 4 above): the use of PB and PB* — our 'cut rules' — allow us to avoid many redundant branchings in the refutation tree.

A lemma analogous to Lemma 5.4.1 holds:

Lemma 5.5.1 *If $\mathcal{T}$ is an $\mathbf{RE}_{\mathrm{fde}}$-tree for U, where U is a set of signed formulae, then the tree $\mathcal{T}'$ obtained from $\mathcal{T}$ by replacing every (occurrence of an) s-formula with (an occurrence of) its converse is an $\mathbf{RE}_{\mathrm{fde}}$-tree for the set U^* of the* converses *of the s-formulae in U.*

Soundness. The proof is strictly analogous to the one given for the tableau method.

Completeness. We define a notion akin to the notion of R-Hintikka set :

$$
\begin{array}{c}
\mathrm{t}((A \vee B) \wedge \neg A) \\
\mathrm{f}(B \vee (A \wedge \neg A)) \\
\mathrm{t}(A \vee B) \\
\mathrm{t}(\neg A) \\
\mathrm{f}^*(A) \\
\mathrm{f}(B) \\
\mathrm{f}(A \wedge \neg A) \\
\mathrm{t}(A) \\
\mathrm{f}(\neg A) \\
\mathrm{t}^*(A)
\end{array}
$$

Figure 5.3: An $\mathbf{RE}_{\mathrm{fde}}$-proof of $((A \vee B) \wedge \neg A) \rightarrow (B \vee (A \wedge \neg A))$

Definition 5.5.2 Let us say that a set of *signed* formulae U is an *R-analytic set* if and only if it satisfies the following conditions:

A_0: No signed variable and its conjugate are both in U.

A_1: If $\alpha \in U$, then $\alpha_1 \in U$ and $\alpha_2 \in U$.

A_2: If $\beta \in U$ and $\beta'_1 \in U$, then $\beta_2 \in U$.

A_3: If $\beta \in U$ and $\beta'_2 \in U$, then $\beta_1 \in U$.

A_4: If $\gamma \in U$, then for *every* k in U, $\gamma(k) \in U$.

A_5: If $\delta \in U$, then for *some* k in U, $\delta(k) \in U$.

An R-analytic set differs from a R-Hintikka set in that it may be the case that for some β in the set neither β_1 nor β_2 are in the set.

Definition 5.5.3 We say that an R-analytic set is *β-complete* if for every β in U either of the following two conditions is satisfied:

1. either $\beta_1 \in U$ or $\beta'_1 \in U$;

2. either $\beta_2 \in U$ or $\beta_2' \in U$;

It is then easy to verify that:

Fact 5.5.1 *If U is an R-analytic set and U is β-complete, then U is an R-Hintikka set.*

It is not difficult to define a procedure which, given a set of s-formulae U, generates either a closed $\mathbf{RE}_{\mathrm{fde}}$-tree or an open $\mathbf{RE}_{\mathrm{fde}}$-tree such that for every (possibly infinite) open branch ϕ the set of all the s-formulae occurring in ϕ is a R-analytic set which is also β-complete (the only tricky part of such a procedure concerns condition A_4 and can be dealt with as in [Smu68a, pp.58–60]). Thus, completeness follows from fact 5.5.1 and lemma 5.4.4.

We can define the notions of *weakly closed branch* and *weakly closed tree* in exactly the same way as we did for the tableau method discussed in the previous section. Again the relation between classical and relevant deducibility is the same *mutatis mutandis*.

Our proof of the completeness of $\mathbf{RE}_{\mathrm{fde}}$ yields the subformula principle as a corollary:

Corollary 5.5.1 (Analytic Cut Property) *If there is a closed $\mathbf{RE}_{\mathrm{fde}}$-tree $\mathcal{T}$ for U, then there is a closed $\mathbf{RE}_{\mathrm{fde}}$-tree $\mathcal{T}'$ for U such that the rules PB and PB* are applied only to subformulae of s-formulae in U.*

In fact, the proof shows that, when applying PB or PB*, we need to consider only the immediate signed subformulae of signed formulae of type β occurring above in the same branch and which have not been already 'analysed'. Since all the logical rules preserve the subformula property, we have:

Corollary 5.5.2 (Subformula Principle) *If there is a closed $\mathbf{RE}_{\mathrm{fde}}$-tree $\mathcal{T}$ for U, then there is a closed $\mathbf{RE}_{\mathrm{fde}}$-tree $\mathcal{T}'$ for U such that every s-formula occurring in $\mathcal{T}'$ is a signed subformula of s-formulae in U*

A constructive proof of the subformula principle, which yields a procedure for transforming any $\mathbf{RE}_{\mathrm{fde}}$-proof in an equivalent $\mathbf{RE}_{\mathrm{fde}}$-proof which enjoys the subformula property, can be obtained by adapting the proof given by Mondadori for the system **KE** [Mon88b].

The completeness of the (propositional fragment of the) system $\mathbf{RE}_{\mathrm{fde}}$ can also be proved by showing that all the axioms of the Hilbert-style formulation

(given on p. 93) are theorems of $\mathbf{RE}_{\mathrm{fde}}$ and $\mathbf{RE}_{\mathrm{fde}}$-derivability is closed under the rules (8)-(11). All these facts are shown below.

In what follows $\vdash$ stands for $\vdash_{\mathbf{RE}_{\mathrm{fde}}}$.

Fact 5.5.2 $A \wedge B \vdash A$

(1)	$\mathrm{t}(A \wedge B)$	Assumption
(2)	$\mathrm{f}(A)$	Assumption
(3)	$\mathrm{t}(A)$	Et$\wedge$ (1)
(4)	$\mathrm{t}(B)$	Et$\wedge$ (1)

Fact 5.5.3 $A \wedge B \vdash B$

Proof as above.

Fact 5.5.4 $A \vdash A \vee B$

(1)	$\mathrm{t}(A)$	Assumption
(2)	$\mathrm{f}(A \vee B)$	Assumption
(3)	$\mathrm{f}(A)$	Ef$\vee$ (2)
(4)	$\mathrm{f}(B)$	Ef$\vee$ (2)

Fact 5.5.5 $B \vdash A \vee B$

Proof as above.

Fact 5.5.6 $A \wedge (B \vee C) \vdash (A \wedge B) \vee C$

(1)	$\mathrm{t}(A \wedge (B \vee C))$	Assumption
(2)	$\mathrm{f}((A \wedge B) \vee C)$	Assumption
(3)	$\mathrm{t}(A)$	Et$\wedge$ (1)
(4)	$\mathrm{t}(B \vee C)$	Et$\wedge$ (1)
(5)	$\mathrm{f}(A \wedge B)$	Ef$\vee$ (2)
(6)	$\mathrm{f}(C)$	Ef$\vee$ (2)
(7)	$\mathrm{t}(B)$	Et $\vee$ 2 (4,6)
(8)	$\mathrm{f}(B)$	Ef $\wedge$ 1 (5,3)

Fact 5.5.7 $A \vdash \neg\neg A$

(1)	$\mathrm{t}(A)$	Assumption
(2)	$\mathrm{f}(\neg\neg A)$	Assumption
(3)	$\mathrm{t}^*(\neg A)$	Ef$\neg$ (2)
(4)	$\mathrm{f}(A)$	Et$^*\neg$ (3)

Fact 5.5.8 $\neg\neg A \vdash A$

(1)	$\mathrm{t}(\neg\neg A)$	Assumption
(2)	$\mathrm{f}(A)$	Assumption
(3)	$\mathrm{f}^*(\neg A)$	Ef$\neg$ (1)
(4)	$\mathrm{t}(A)$	Ef$^*\neg$ (3)

Fact 5.5.9 *If $A \vdash B$ and $B \vdash C$, then $A \vdash C$*

It follows from the hypothesis that there are closed trees $\mathcal{T}_1$ and $\mathcal{T}_2$ for $\{\mathrm{t}(A), \mathrm{f}(B)\}$ and $\{\mathrm{t}(B), \mathrm{f}(C)\}$ respectively. Therefore the following is a closed tree for $\{\mathrm{t}(A), \mathrm{f}(C)\}$:

$$\begin{array}{c} \mathrm{t}(A) \\ \mathrm{f}(C) \\ \hline \mathrm{t}(B) \quad | \quad \mathrm{f}(B) \\ \mathcal{T}_2 \qquad\quad \mathcal{T}_1 \end{array}$$

Fact 5.5.10 *If $A \vdash B$ and $A \vdash C$, then $A \vdash B \wedge C$*

It follows from the hypothesis that there are closed trees $\mathcal{T}_1$ and $\mathcal{T}_2$ for $\{\mathrm{t}(A), \mathrm{f}(B)\}$ and $\{\mathrm{t}(A), \mathrm{f}(C)\}$ respectively. Therefore the following is a closed tree for $\{\mathrm{t}(A), \mathrm{f}(B \wedge C)\}$:

$$\begin{array}{c} \mathrm{t}(A) \\ \mathrm{f}(B \wedge C) \\ \hline \mathrm{t}(B) \quad | \quad \mathrm{f}(B) \\ \mathrm{f}(C) \qquad\quad \mathcal{T}_1 \\ \mathcal{T}_2 \end{array}$$

Fact 5.5.11 *If $A \vdash C$ and $B \vdash C$, then $A \vee B \vdash C$*

It follows from the hypothesis that there are closed trees $\mathcal{T}_1$ and $\mathcal{T}_2$ for $\{\mathrm{t}(A), \mathrm{f}(C)\}$ and $\{\mathrm{t}(B), \mathrm{f}(C)\}$ respectively. Therefore the following is a closed tree for $\{\mathrm{t}(A \vee B), \mathrm{f}(C)\}$:

$$\begin{array}{c} \mathrm{t}(A \vee B) \\ \mathrm{f}(C) \\ \hline \mathrm{t}(A) \quad | \quad \mathrm{f}(A) \\ \mathcal{T}_1 \qquad\quad \mathrm{t}(B) \\ \qquad\quad\ \ \mathcal{T}_2 \end{array}$$

Fact 5.5.12 *If* $A \vdash B$, *then* $\neg B \vdash \neg A$.

By hypothesis there is a closed trees $\mathcal{T}$ for $\{\mathrm{t}(A), \mathrm{f}(B)\}$. It follows from Lemma 5.5.1 that there is a closed tree $\mathcal{T}'$ for $\{\mathrm{t}^*(A), \mathrm{f}^*(B)\}$. Therefore the following is a closed tree for $\{\mathrm{t}(\neg B), \mathrm{f}(\neg A)\}$:

$$\begin{array}{c} \mathrm{t}(\neg B) \\ \mathrm{f}(\neg A) \\ \mathrm{f}^*(B) \\ \mathrm{t}^*(A) \\ \mathcal{T}' \end{array}$$

We conclude by mentioning that a 'naive' implementation in Prolog of the system $\mathbf{RE}_{\mathrm{fde}}$ has been developed by Rajev Gore [Gor90].

Chapter 6

A generalization: cut systems

6.1 Proper derived rules

We want to address the following problem:

WHAT IS A DERIVED RULE ?

As background for our investigation we assume the Gentzen-style formalization which uses *sequents*, i.e. expressions of the form:

$$A_0, A_1, \ldots, A_n \vdash B_0, B_1, \ldots, B_m \tag{6.1}$$

with the usual meaning: if all of the A_i's hold, then at least one of the B_i's holds. It is convenient to treat the antecedent and the succedent as *sets* instead of sequences.

By a *Gentzen rule* R we mean a schematic figure for passing from several sequents to another sequent, a typical example being the rule of *Modus Ponens*:

$$\text{(MP)} \qquad \frac{\Gamma \vdash A \quad \Delta \vdash A \to B}{\Gamma, \Delta \vdash B}$$

A finite set of Gentzen rules defines a *Gentzen system*. Given a Gentzen system **S**, there are at least three natural ways of representing proofs in it: as *trees*, as *directed acyclic graphs* or as *sequences* of sequents generated in accordance with the rules of **S**. Whereas there is no difference in complexity

between a proof in the form of directed acyclic graph (or 'd.a.g.') and a proof in the form of a sequence of seqnents, it is well-known that representing proofs in tree-form may introduce a great deal of redundancy, because identical subtrees may appear many times. However, we shall observe in the sequel that the format in which proofs are represented seems to be significant only in some special cases.

The paradigmatic Gentzen system is the calculus of sequents. There are other systems, like Natural Deduction, which can also be represented as Gentzen systems, though their original formulation is not in terms of sequents. In this chapter we restrict our attention to *classical* systems.

Definition 6.1.1 We say that a sequent is *correct for* a Gentzen system **S** if and only if it is provable in **S**. A *rule* R is *correct* for **S** if and only if the conclusion of R is correct for **S** whenever all the premises are correct for **S**.

The notion of correctness of a sequent (and of a rule) makes sense even for systems which are not formulated in terms of sequents. Given a sequent $\Gamma \vdash \Delta$, with $\Delta = \{A_1, \ldots, A_n\}$, by a *single-conclusion sequent associated with* $\Gamma \vdash \Delta$ we mean any sequent:

$$\Gamma, \neg A_1, \ldots, \neg A_{i-1}, \neg A_{i+1}, \ldots, \neg A_n \vdash A_i$$

for $i = 1, \ldots, n$. In classical logic, every sequent is equivalent to every single-conclusion sequent associated with it. So, given an arbitrary proof system **S** (not necessarily formulated in term of sequents), we take as a proof of a multiple-conclusion sequent any proof of a single-conclusion sequent associated with it, that is a proof of the single formula in the succedent from the formulae in the antecedent.

Let us go back to our initial problem: what is a derived rule? A first (as we shall argue, unsatisfactory) answer is the following:

(1) A rule R is a derived rule of a system **S** if and only if R is correct for **S**.

Incidentally, this answer is assumed in many logic textbooks. It is motivated by the fact that when a rule is correct for a system **S** its addition to **S** does not increase the stock of provable sequents.

If we accept such a definition of derived rule, then saying that R is a derived rule of **S** means only that the relation $\vdash$ is closed under R (*there is* a proof of the sequent below the inference line whenever there are proofs of the sequents above the inference line).

From a proof-theoretical point of view, however, we are not interested in the mere *existence* of a proof of the conclusion of a rule R, but in an *effective procedure* for constructing such a proof from proofs of the premises. This leads us to a different notion of derived rule:

(2) A rule R is a derived rule of **S** if and only if there is an effective procedure for constructing a proof of the conclusion of R from proofs of its premises.

But, if we are interested in the complexity of proofs, we should also require that the procedure be *efficient*. A natural requirement, for instance, is that the complexity of the resulting proof does not exceed the sum of the complexities of the component proofs plus a constant c. To make this further requirement precise we can use the λ-measure used in Chapter 4:

Definition 6.1.2 The *complexity* of a proof π, denoted by $\lambda(\pi)$, is the *number of lines* in π (each 'line' being a sequent, a formula, or any other expression associated with an inference step, depending on the system under consideration).

We are now ready to give a definition of derived rule which takes into account the complexity of the procedure associated with it:

(3) An n-premise rule R is a derived rule of **S** if and only if there is an effective procedure, consisting of applications of the primitive rules of **S**, for constructing a proof π of the conclusion of R from proofs $\pi_1, \ldots, \pi_n$ of its premises, such that $\lambda(\pi) \leq \lambda(\pi_1) + \cdots + \lambda(\pi_n) + c$ for some constant c.

We shall call a derived rule satisfying (3) *proper derived rule*.

We suggest that this notion of proper derived rule can be useful for the analysis of the relative complexity of formal proofs. In the rest of this chapter we shall illustrate this claim in a few simple examples.

Again, we point out that the λ-measure is sufficient to establish negative results, but is not sufficient in general for positive results. It may, however, be adequate also for positive results whenever one can show that the length of lines is not significantly increased by the simulation procedure under consideration. All the procedures that we shall cousider in the sequel will be of this kind. We assume the notions of simulation and p-simulation as defined in Chapter 4.

Definition 6.1.3 Let us say that $\mathbf{S}'$ *linearly simulates* $\mathbf{S}$ if and only if there is a constant c such that for every $\mathbf{S}$-proof π of A from Γ there is an $\mathbf{S}'$-proof π' of A from Γ such that $\lambda(\pi') \leq c \cdot \lambda(\pi)$.

In other words $\lambda(\pi') = O(\lambda(\pi))$.

The following proposition is a straightforward consequence of our definitions:

Proposition 6.1.1 *Let* $\mathbf{S}$ *be a Gentzen system* in tree form *and* $\mathbf{S}'$ *be an arbitrary proof system. If all the rules of* $\mathbf{S}$ *are proper derived rules of* $\mathbf{S}'$, *then* $\mathbf{S}'$ *linearly simulates* $\mathbf{S}$.

Among all the possible rules a special role is played by four rules which are correct for an important class of logics (including, of course, the topic of this chapter: classical logic).
The rule of *reflexivity*

(REFL) $\qquad \Gamma \vdash A$ whenever $A \in \Gamma$

the rule of *monotonicity*

(MONO) $$\frac{\Gamma \vdash \Delta}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

the rule of *substitutivity*

(SUBST) $$\frac{\Gamma \vdash \Delta}{s\Gamma \vdash s\Delta}$$

for any substitution function s ($s\Gamma$ for a set Γ is defined in the obvious way), and the rule of *transitivity* or *cut*

$$\text{(CUT)} \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

These rules will be called *common rules*. They are, of course, correct for all classical systems. Since it is difficult to imagine a sensible proof system in which REFL, MONO and SUBST are *not* proper derived rules, we shall restrict our attention to systems in which they are proper derived rules and shall assume, for simplicity's sake, that the constant c, representing the λ-complexity of the derivation of the rule, is equal to 0. On the other hand, there are well-known proof systems in which CUT, though being obviously a correct rule (that is a 'derived rule' in the weak sense), is not a proper derived rule. Since CUT expresses the basic transitivity property of proofs, these systems can hardly claim to represent a realistic model of the notion of classical proof.

6.2 Cut systems

It is somewhat surprising how far one can go by assuming only that cut is a proper derived rule of a system. The next proposition, for example, states that any such system is as powerful (from the point of view of complexity) as any Frege system (another denomination of Hilbert-style proof systems).

Proposition 6.2.1 *If cut is a proper derived rule of* **S**, **S** *linearly simulates any Frege system.*

Proof. A Frege proof π is a sequence of formulae $A_1, \ldots, A_n$, where A_n is the theorem and A_i, $i = 1, \ldots, n$ is either an axiom or follows from previous lines by means of a rule. We shall consider axioms as rules with 0 premises. Since **S** is complete, then for each rule $R = B_1, \ldots, B_m/C$, $m \geq 0$, there is an **S**-proof π_R of C from $B_1, \ldots, B_m$. Let $c = \max(\lambda(\pi_R))$ for R ranging over the set of rules of the Frege system. Let A_n follow from $A_{j1}, \ldots, A_{jk}$, with $ji \leq n-1$. Then, there is an **S**-proof of A_n from $A_{j1}, \ldots, A_{jk}$ and, by MONO, from $A_1, \ldots, A_{n-1}$ with λ-complexity less than or equal to c. Now, let A_{n-1} follow from $A_{p1}, \ldots, A_{pq}$, with $pi \leq n-2$. Then there is an **S**-proof of A_{n-1}

from $A_{p1}, \ldots, A_{pq}$ and, by MONO, from $A_1, \ldots, A_{n-2}$ with λ-complexity less than or equal to c. Since CUT is a proper derived rule of **S**, there is a proof of A_n from $A_1, \ldots, A_{n-2}$ with complexity less than or equal to $2c + d$, where d is the constant associated with the simulation of CUT in **S**. It is easy to verify that by repeating this procedure we eventually obtain an **S**-proof π' of A_n with $\lambda(\pi') \leq c \cdot \lambda(\pi) + d \cdot (\lambda(\pi) - 1)$. □

Remarks: It follows from results in [CR79] that Frege systems can polynomially simulate all natural deduction systems and all Gentzen systems. More recently Buss [Bus87] and Urquhart [Urq87] have shown that resolution cannot polynomially simulate Frege systems. Urquhart has also shown [Urq89] that the cut-free sequent calculus in d.a.g. form cannot p-simulate the sequent calculus with cut. Thus Proposition 6.2.1 above implies that neither resolution nor the cut-free sequent calculus can p-simulate any system in which cut is a proper derived rule.

Notice that our definition of proper derived rule does not make any assumption about the *format* in which proofs are represented. In particular there are many systems in *tree* form (including Natural Deduction and Gentzen's sequent calculus with cut) in which cut *is* a proper derived rule (or can be rendered such by means of minor transformations, see chapter 4). The proposition above then implies that neither resolution nor the cut-free sequent calculus in *d.a.g. form* can p-simulate any system in *tree form* in which cut is a proper derived rule.

Finally notice that the assumption that the system is classical does not play any role in the proof. Therefore it can be extended to any logical system in which the common rules and cut are proper derived rules.

The rule of CUT we have given above is completely general and does not involve any logical constant. Auother form of CUT which is typical of classical systems is the following:

$$\text{(CUT*)} \qquad \frac{\Gamma, A \vdash \Delta \quad \Gamma', \neg A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

This form of cut involves the negation operator and implies that it behaves in a classical way. In classical logic the behaviour of negation is captured by

two basic rules. The first one is also correct in intuitionistic logic:

$$(\mathrm{I}\neg) \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta}$$

The second one is peculiar to classical logic:

$$(\mathrm{E}\neg) \qquad \frac{\Gamma, \neg A \vdash \Delta}{\Gamma \vdash A, \Delta}$$

It is easy to verify that (i) if CUT* and I$\neg$ are proper derived rules of a system **S**, so are CUT and E$\neg$, and (ii) if CUT and E$\neg$ are proper derived rules of **S**, so are CUT* and I$\neg$.

Definition 6.2.1 Say that a system **S** is a *classical cut system* or simply a *cut system* if both CUT* and I$\neg$ are proper derived rules of **S** or, equivalently, if both CUT and E$\neg$ are proper derived rules of **S**.

We have already considered cut systems in relation to **KE** in chapter 4. The definition adopted there was slightly different, but can be easily proved equivalent to the present one under the assumption that the common rules are proper derived rules. Incidentally, this is an illustration of how the notion of cut system is robust under reasonable changes of definition. The following proposition states a property of cut systems:

Proposition 6.2.2 *If* **S** *is a cut system, then all the rules of Gentzen's natural deduction and sequent calculus in tree form are proper derived rules of* **S**. *Moreover, if* **S** *enjoys the subformula property, the derivation of each rule involves only formulae which are (weak) subformulae of formulae occurring in the premises or in the conclusion of the rule.*

The proof is left to the reader.

6.3 Analytic cut systems versus cut-free systems

The appeal of some well-known systems in which CUT is *not* a proper derived rule, like Gentzen's sequent calculus without cut or the tableau method, is

mainly due to the fact that these systems yield proofs with the subformula property (SFP). In contrast, a system in which CUT is a primitive or a proper derived rule allows for proofs without the subformula property. However, it may well be that such 'non-analytic' proofs are not necessary for completeness, i.e. that the existence of a proof with the SFP is always guaranteed. This is trivially true of all systems in which CUT is eliminable. On the other hand, it is obvious that we do not need to *eliminate* CUT in order to obtain proofs with the subformula property: it is sufficient to restrict ourselves to 'analytic cuts', i.e. to cut inferences in which the cut formula is a subformula of the assumptions or of the conclusion. It is well-known that the *elimination* of cuts can greatly increase the complexity of proofs. But we have seen in Chapter 4 that — if we consider proofs in tree-form — proofs including 'analytic cuts' can be essentially shorter than any equivalent cut-free proof while still enjoying the SFP. In particular, we have seen in Section 4.6 that any analytic cut system is essentially more efficient than the tableau method. For the reader's convenience we restate this result here in a different form:

Proposition 6.3.1 *Let* **S** *be any analytic cut system. Then the analytic restriction of* **S** *linearly simulates Gentzen's sequent calculus without cut (in tree form). But Gentzen's sequent calculus without cut (in tree form) cannot p-simulate the analytic restriction of* **S**.

Sketch of the proof. The positive part follows from Proposition 6.1.1 and Proposition 6.2.2. The negative part follows from Theorem 4.5.4 and Theorem 4.6.1.

The interest of these results lies mainly in the fact that the cut-free sequent calculus in tree-form (or the tableau method) seems to lend itself well to computer search. The tree-form, in this case, is more natural. Moreover, as far as the cut-free sequent calculus is concerned, it seems that one can find relatively short proofs in d.a.g. or linear form only by employing the Thinning rule (for some results see [Urq90b]). However, as argued in Section 2.1, this rule does not suit the usual 'bottom-up' search procedures used in automated deduction, essentially because it is not *invertible*. On the other hand there are analytic cut systems, like our **KE**, which are as suitable for automation as cut-free Gentzen systems but are essentially more efficient.

Chapter 7

Conclusions

Our discussion shows that **KE** constitutes a refutation system which lends itself well to computer search. We propose it as an alternative both to the tableau method and resolution which combines features of both systems and remedies some of their drawbacks. It is sufficiently similar to the tableau method to share all its 'desirable' features: it does not require reduction to any normal form, obeys the subformula principle, works in the 'analytic' (or 'bottom-up') direction, is 'natural', easy to implement and can be adapted to a wide variety of non-classical logics. On the other hand it avoids the basic redundancy of cut-free methods and establishes a closer connection with classical semantics as a result of the crucial role played by the rule PB in the analysis of the formulae of the 'disjunctive' (β) type. We have explicitly stressed the connection between this kind of analysis and that the Davis-Putnam procedure which is, in turn, closely related to resolution. So, from this point of view, **KE** can be seen as an extension of a resolution-style analysis to the domain of full first-order logic. Our semantic-oriented argument, presented in Chapter 2, is sufficient to ensure that this kind of analysis represents a *uniform* improvement over the traditional cut-free analysis, at least as far as the complexity of proofs is concerned. The results contained in Chapter 4 confirm and strengthen this argument in the more conventional framework of complexity theory: the **KE** analysis stands with the cut-free in a relation of *dominance* with respect to p-simulation (**KE** linearly simulates the tableau method but the tableau method cannot p-simulate **KE**). Whether or not such considerations of complexity be relevant to the choice of a formal representation for purposes other than the practical ones, is a

matter which is still to be explored. Our discnssion in Chapter 2, however, strongly suggests that the tableau method is not a 'natnral' formalization of the notion of a classical refutation and goes near to characterizing **KE** as *the* natural one.

From the vantage-point of **KE** cut-free systems are seen as *too restricted*, in a way which is not justified by their own purposes (subformula property and easy proof-search). Similarly, resolution is too restricted from the syntactical point of view: efficient classical refutations can be obtained without confining ourselves to the domain of formnlae in clausal form. There are, in fact, connections between **KE** and non-clausal resolution [Mur82, MW80]. However **KE** represents refutations in a completely different way: its rules are genuine *inference rules*, in the sense that they formalize traditional inference principles. We have also emphasized (in Section 3.8) that **KE** can be represented as a kind of 'natnral deduction' system. Moreover, the presence of a rule like PB, which can be used in a non-analytic way, makes it possible to use **KE** to formalize non-analytic argnments as well as analytic ones. This should be considered as an advantage from many points of view. First, as Dana Scott pointed out [Sco73], analytic logics, i.e. logics which admit of formal representations that obey the subformula principle, are more an exception than the rule. Therefore **KE** provides a logical framework which can be extended in a natural way to non-classical logics for which a cut-free representation is impossible. Second, even within the domain of classical logic, the possibility of representing non-analytic proofs can be an advantage. For example, it has been known since the earliest work on automated theorem proving (see for instance [Wan60]) that biconditionals tend to cause combinatorial explosion in proof search procednres based on a cut-free Gentzen system. Recent resnlts by Urqnhart [Urq89, Urq90b] have not only proved that this phenomenon is unavoidable, but also that the use of analytic cut cannot help as far as pure biconditional expressions are concerned (see Theorem 6.1 in [Urq90b]). In this case, non-analytic methods are definitely more efficient. This is one example of the well-known fact that using non-analytic, 'external', lemmata can considerably shorten proofs. Boolos [Boo84] has illustrated this fact strikingly by exhibiting a natural class of first-order schemata which have no feasible tablean proofs but do have short natural deduction proofs. He traced the fact that natural deduction methods are essentially more efficient than the tableau method to the fact that '*modus ponens*, or *cut*, is obviously a valid derived rule of standard natural deduc-

tion systems, but *not obviously* a valid derived rule of the method of trees'[1], and adds that 'the most significant feature possessed by natural deduction methods but not by the tree method, a feature that can easily seem like a virtue, is [...] that it permits the development and the utilization within derivations of *subsidiary* conclusions'[2]. Our notions of 'proper derived rule' and 'cut-system', defined and used in Chapter 6, were also meant to formalize some of the intuitions underlying Boolos' remark.

So, there are many contexts in which purely analytic methods are hopelessly inefficient. As Boolos himself pointed out, there is a trivial solution to this problem: simply add cut to the cut-free rules. But this move would not eliminate the redundancy of the cut-free rules that we have pointed out in Chapter 2. Moreover, the cut rule would be, so to speak, 'thrown into' the cut-free system in a purely external way, and we would be left with no obvious criteria concerning its application. We claim that **KE** represeuts a natural and elegant solution to this problem: it does not employ any rule which generates redundancy in the sense of Chapter 2; the cut rule comes into play only when the operational rules cannot be further applied; although its applications can be restricted to analytic ones, the unrestricted system can be used to simulate efficiently all the conventional non-analytic proof systems and therefore any algorithm based on them.

Given our previous considerations, some future directions of reasearch suggest themselves. One concerns the development of efficient proof search procedures for classical logic based on **KE**. We expect that, for analytic procedures, some variant of Bibel's connectiou method ([Bib82]; see also the exposition in [Wal90]), should prove useful. Similar proof-search procedures for non-classical logics could be developed by adapting Wallen's metbods [Wal90]. Moreover, as suggested above, we expect **KE** to be useful in formulating non-analytic proof-search algorithms both for classical and non-classical logics, especially when non-analytic methods are the only efficient ones, or even, in the case of some non-classical logics, the only possible ones.

[1] [Boo84], p.373.
[2] [Boo84], p.377.

Bibliography

[AB75] A. R. Anderson and N.D. Belnap Jr. *Entailment: the Logic of Relevance and Necessity*, volume 1. Princeton University Press, Princeton, 1975.

[Ajt88] M. Ajtai. The complexity of the pigeonhole principle. In *Proceedings of the 29th Annual Symposium on the Foundations of Computer Science*, 1988. Preliminary version.

[Avr88] A. Avron. The semantics and proof theory of linear logic. *Theoretical Computer Science*, 57:161–184, 1988.

[Bel76] N. D. Belnap Jr. How a computer should think. In G. Ryle, editor, *Contemporary Aspects of Philosophy*, pages 30–55. Oriel Press, 1976.

[Bel77] N. D. Belnap Jr. A useful four-valued logic. In J. M. Dunn and G. Epstein, editors, *Modern uses of multiple-valued logics*, pages 8–37. Reidel, Dordrecht, 1977.

[Bet55] E. W. Beth. Semantic entailment and formal derivability. *Mededelingen der Koninklijke Nederlandse Akademie van Wetenschappen*, 18:309–342, 1955.

[Bet58] E. W. Beth. On machines which prove theorems. *Simon Stevin Wissen Natur-kundig Tijdschrift*, 32:49–60, 1958. Reprinted in [SW83], vol. 1, pages 79–90.

[BG88] S. R. Buss and G. Turán. Resolution proofs of generalized pigeonhole principles. *Theoretical Computer Science*, 62:311–17, 1988.

[Bib82] W. Bibel. *Automated Theorem Proving.* Vieweg, Braunschweig, 1982.

[Bla70] R. Blanché. *La logique et son histoire.* Armand Colin, Paris, 1970.

[Boc61] I. M. Bochensky. *A History of Formal Logic.* University of Notre Dame, Notre Dame (Indiana), 1961.

[Boo84] G. Boolos. Don't eliminate cut. *Journal of Philosophical Logic*, 7:373–378, 1984.

[BS89] H. A. Blair and V. S. Subrahmanian. Paraconsistent logic programming. *Theoretical Computer Science*, 68:135–154, 1989.

[Bus87] S. R. Buss. Polynomial size proofs of the pigeon-hole principle. *The Journal of Symbolic Logic*, 52:916–927, 1987.

[Car87] W. A. Carnielli. Systematization of finite many-valued logics through the method of tableaux. *The Journal of Symbolic Logic*, 52:473–493, 1987.

[Cel87] C. Cellucci. Using full first order logic as a programming language. In *Rendiconti del Seminario Matematico Università e Politecnico di Torino. Fascicolo Speciale 1987*, pages 115–152, 1987. Proceedings of the conference on 'Logic and Computer Science: New Trends and Applications'.

[Cel88] C. Cellucci. Efficient natural deduction. In *Temi e prospettive della logica e della filosofia della scienza contemporanee*, volume I, pages 29–57,CLUEB Bologna, 1988.

[Cha70] C.L. Chang. The unit proof and the input proof in theorem proving. *Journal of the Association for Computing Machinery*, 17:698–707, 1970.

[CL73] C.L Chang and R.C.T. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, Boston, 1973.

[Coo71] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual Symposium on the Theory of Computing*, 1971.

[Coo78] S. A. Cook. Letter to Richard Statman. Unpublished, 4, April 1978.

[CR74] S. A. Cook and R. Rechow. On the length of proofs in the propositional calculus. In *Proceedings of the 6th Annual Symposium on the Theory of Computing*, pages 135–148, 1974.

[CR79] S. A. Cook and R. Rechow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44:36–50, 1979.

[CS88] V. Chvátal and E. Szemerédi. Many hard examples for resolution. *Journal of the Association for Computing Machinery*, 35:759–768, 1988.

[Dav83] M. Davis. The prehistory and early history of automated deduction. In [SW83], pages 1–28. 1983.

[DCHLS90] N.C.A. Da Costa, L.J. Henschen, J.J. Lu, and V.S. Subrahmanian. Automatic theorem proving in paraconsistent logics: Theory and implementation. In M.E. Stickel, editor, *Lecture Notes in Artificial Intelligence*, volume 449, pages 72–86, 1990. 10th International Conference on Automated Deduction.

[DH76] B. Dunham and H. Wang. Towards feasible solutions of the tautology problem. *Annals of Mathematical Logic*, 10:117–154, 1976.

[DLL62] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the Association for Computing Machinery*, 5:394–397, 1962. Reprinted in [SW83], pp. 267–270.

[DP60] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7:201–215, 1960. Reprinted in [SW83], pp. 125–139.

[Dum78] M. Dummett. *Truth and other Enigmas.* Duckworth, London, 1978.

[Dun76] J. M. Dunn. Intuitive semantics for first-degree entailment and coupled trees. *Philosophical Studies*, 29:149–168, 1976.

[FE89] L. Farinas del Cerro and P. Enjalbert. Modal resolution in clausal form. *Theoretical Computer Science*, 65:1–33, 1989.

[Fit83] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* Reidel, Dordrecht, 1983.

[Fit87] M. Fitting. Resolution for intuitionistic logic. In Z.W. Ras and M. Zemankova, editors, *Methodologies for Intelligent Systems*, pages 400–407. North-Holland, Amsterdam, 1987.

[Fit88] M. Fitting. First-order modal tableaux. *Journal of Automated Reasoning*, 4:191–213, 1988.

[Fit90] M. Fitting. *First-Order Logic and Automated Theorem Proving.* Springer-Verlag, Berlin, 1990.

[Gab90a] D. M. Gabbay. Lds-labelled deductive systems. Preliminary draft of a book intended for Oxford University Press, May 1990.

[Gab90b] D. M. Gabbay. Theory of algorithmic proof. In S. Abrfamsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Theoretical Computer Science*, volume 1. Oxford University Press, 1990. To appear.

[Gal77] Z. Galil. On the complexity of regular resolution and the Davis-Putnam procedure. *Theoretical Computer Science*, 4:23–46, 1977.

[Gal86] J. H. Gallier. *Logic for Computer Science.* Harper & Row, New York, 1986.

[Gen35] G. Gentzen. Unstersuchungen über das logische Schliessen. *Math. Zeitschrift*, 39:176–210, 1935. English translation in [Sza69].

[Gir87a] J. Y. Girard. *Proof Theory and Logical Complexity.* Bibliopolis, Napoli, 1987.

[Gir87b] J. Y. Girard. Lineal logic. *Theoretical Computer Science*, 50:1–101, 1987.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the theory of NP-Completeness.* W.H. Freeman & Co., San Francisco, 1979.

[Gor90] R. Gore. Naive implementations of semantic tableaux with and without cut for classical logic, asserted 3-valued logic and belnap's 4-valued logic. Technical report, Cambridge University Computer Laboratory, 1990. Forthcoming.

[Hac85] A. Hacken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[Hin55] J. K. J. Hintikka. Form and content in quantification theory. *Acta Philosophica Fennica*, VIII, 1955.

[Jef81] R. C. Jeffrey. *Formal Logic: its Scope and Limits.* McGraw-Hill Book Company, New York, second edition, 1981. First edition 1967.

[Kal34] L. Kalmar. Über die Axiomatisierbarkeit des Aussagenkalkulus. *Acta Scient. Math. Szeged*, 7:222–243, 1934.

[Kan57] S. Kanger. Provability in logic. *Acta Universitatis Stockolmiensis, Stockolm studies in Philosophy*, 1, 1957.

[Kan63] S. Kanger. A simplified proof method for elementary logic. In *Computer Programming and Formal Systems*, pages 87–94. North-Holland, Amsterdam, 1963. Reprinted in [SW83].

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. 1972.

[Kle67] S. C. Kleene. *Mathematical Logic.* John Wiley & Sons, Inc., New York, 1967.

[Mat62] V. A. Matulis. Two versions of classical computation of predicates witbout structural rules. *Soviet Mathematics*, 3:1770–1773, 1962.

[MMO83] S. Yu. Maslov, G. E. Mints, and V.P Orevkov. Mechanical proof-search and the theory of logical deduction in the USSR. In [SW83], pages 29–38. 1983.

[Mon88a] M. Mondadori. Classical analytical deduction. Annali dell'Università di Ferrara; Sez. III; Discussion paper series 1, Università di Ferrara, 1988.

[Mon88b] M. Mondadori. Classical analytical deduction, part II. Annali dell'Università di Ferrara; Sez. III; Discussion paper series 5, Università di Ferrara, 1988.

[Mon88c] M. Mondadori. On the notion of a classical proof. In *Temi e prospettive della logica e della filosofia della scienza contemporanee*, volume I, pages 211–214, CLUEB Bologna, 1988.

[Mon88d] M. Mondadori. Sulla nozione di dimostrazione classica. Annali dell'Università di Ferrara; Sez. III; Discussion paper series 3, Università di Ferrara, 1988.

[Mon89] M. Mondadori. An improvement of Jeffrey's deductive trees. Annali dell'Università di Ferrara; Sez. III; Discussion paper series 7, Università di Ferrara, 1989.

[Mur82] N. V. Murray. Completely non-clausal theorem proving. *Artificial Intelligence*, 18:67–85, 1982.

[MW80] Z. Manna and R. Waldinger. A deductive approach for program synthesis. *ACM Transactions on Programming Languages and Systems*, 2:90–121, 1980.

[OS88] F. Oppacher and E. Suen. HARP: A tableau-based theorem prover. *Journal of Automated Reasoning*, 4:69–100, 1988.

[Par66] G. H. R. Parkinson, editor. *Leibniz. Logical Papers.* Oxford University Press, 1966.

[Pra65] D. Prawitz. *Natural Deduction. A Proof-Theoretical Study.* Almqvist & Wilksell, Uppsala, 1965.

[Pra74] D. Prawitz. Comments on Gentzen-type procedures and the classical notion of truth. In A. Dold and B. Eckman, editors, *ISILC Proof Theory Symposium. Lecture Notes in Mathematics, 500*, pages 290–319, Berlin, 1974. Springer.

[Pra78] D. Prawitz. Proofs and the meaning and completeness of the logical constants. In J. Hintikka, I. Niinduoto, and E. Saarinen, editors, *Essays on Mathematical and Philosphical Logic*, pages 25–40. Reidel, Dordrecht, 1978.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12:23–41, 1965.

[Sch56] K. Schütte. Ein System des verknüpfenden Schliessens. *Archiv für matematische Logik und Grundlagenforschung*, 2:34–67, 1956.

[Sch77] K. Schütte. *Proof Theory.* Springer-Verlag, Berlin, 1977.

[Sco70] D. Scott. Outline of a mathematical theory of computation. PRG Technical Monograph 2, Oxford University Computing Laboratory, Programming Research Group, 1970. Revised and expanded version of a paper under the same title in the Proceedings of the Fourth Annual Princeton Conference on Information Sciences and Systems (1970).

[Sco73] D. Scott. Completness and axiomatizability in many-valued logic. In L. Henkin, editor, *Tarsky Symposium.* American Mathematical Society, 1973.

[Smu65] R. M. Smullyan. Analytic natural deduction. *The Journal of Symbolic Logic*, 30:549–559, 1965.

[Smu68a] R. M. Smullyan. *First-Order Logic.* Springer, Berlin, 1968.

[Smu68b] R. M. Smullyan. Analytic cut. *The Journal of Symbolic Logic*, 33:560–564, 1968.

[Smu68c] R. M. Smullyan. Uniform gentzen systems. *The Journal of Symbolic Logic*, 33:549–559, 1968.

[Sta77] R. Statman. Herbrand's theorem and Gentzen's notion of a direct proof. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 897–912. North-Holland, Amsterdam, 1977.

[Sta78] R. Statman. Bounds for proof-search and speed-up in the predicate calculus. *Annals of Mathematical Logic*, 15:225–287, 1978.

[Sti86] M. E. Stickel. An introduction to automated deduction. In W. Bibel and P. Jorrand, editors, *Fundamentals of Artificial Intelligence*, pages 75–132. Springer-Verlag, 1986.

[Sto87] L. Stockmeyer. Classifying the computational complexity of problems. *The Journal of Symbolic Logic*, 52:1–43, 1987.

[Sun83] G. Sundholm. Systems of deduction. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume I, chapter I.2, pages 133–188. Reidel, Dordrecht, 1983.

[SW83] J. Siekman and G. Wrightson, editors. *Automation of Reasoning.* Springer-Verlag, New York, 1983.

[Sza69] M. Szabo, editor. *The Collected Papers of Gerhard Gentzen.* North-Holland, Amsterdam, 1969.

[Ten78] N. Tennant. *Natural Logic.* Edimburgh University Press, Edinburgh, 1978.

[Tho41] I. Thomas, editor. *Greek Mathematics*, volume 2. William Heinemann and Harvard University Press, London and Cambridge, Mass., 1941.

[TMM88] P. B. Thistlewaite, M. A. McRobbie, and B. K. Meyer. *Automated Theorem Proving in Non Classical Logics.* Pitman, 1988.

[Tse68] G. S. Tseitin. On the complexity of derivations in propositional calculus. *Studies in constructive mathematics and mathematical logic. Part 2*, pages 115–125, 1968. Seminars in mathematics, V. A. Stkelov Mathematical Institute.

[Urq87] A. Urquhart. Hard examples for resolution. *Journal of the Association for Computing Machinery*, 34:209–219, 1987.

[Urq89] A. Urquhart. The complexity of Gentzen systems for propositional logic. *Theoretical Computer Science*, 66:87–97, 1989.

[Urq90a] A. Urquhart. Complexity of proofs in classical propositional logic. In Y. Moschovakis, editor, *Logic from Computer Science*. Springer-Verlag, Berkeley. Forthcoming.

[Urq90b] A. Urquhart. The relative complexity of resolution and cut-free Gentzen systems. *Discrete Applied Mathematics*. Forthcoming.

[Wal90] L.A. Wallen. *Automated Deduction in Non-Classical Logics*. The MIT Press, Cambridge, Mass., 1990.

[Wan60] H. Wang. Towards mechanical mathematics. *IBM Journal for Research Development*, 4:2–22, 1960. Reprinted in [SW83], pp. 244-264.