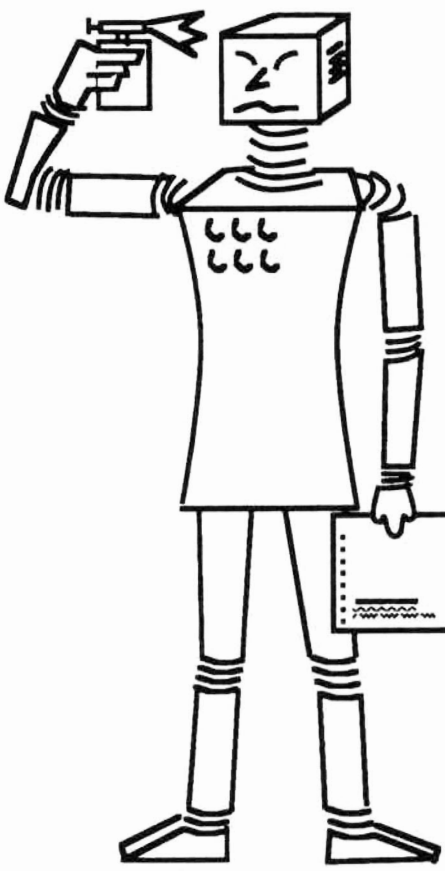
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT



Transformation and Structuring of Computer Generated Proofs

Christoph Lingenfelder

SEKI Report SR-90-26

Transformation and Structuring of Computer Generated Proofs

vom Fachbereich Informatik der Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
genehmigte Dissertation von

Christoph Lingenfelder

Datum der wissenschaftlichen Aussprache:	16. November 1990
Berichterstatter:	Prof. Jörg Siekmann, Ph.D.
	Prof. Dr. Klaus Madlener
Dekan:	Prof. Dr. Gerhard Zimmermann

Abstract

One of the main disadvantages of computer generated proofs of mathematical theorems is often their complexity and incomprehensibility. This is even more of a problem for the presentation of inferences drawn by automated reasoning components in other AI systems. Proof transformation procedures have been designed in order to state these proofs in a formalism that is more familiar to a human mathematician. But usually the essential idea of a proof is still not easily visible.

We describe a procedure to transform proofs represented as abstract refutation graphs into natural deduction proofs with a special emphasis on an "intelligent" selection of inference rules. In particular the frequent use of proofs by contradiction is avoided. During this process topological properties of the refutation graphs can be successfully exploited in order to obtain well-structured proofs. This is accomplished by dividing a large proof into a set of hierarchically arranged subproofs which are more easily comprehensible. This may be achieved by formulating lemmata that are then applied more than once in the subsequent proof, but also by simply inserting subgoals or by breaking up a substantial part of a proof into a case analysis.

Contents

Acknowledgements

First of all I want to express my gratitude to Jörg Siekmann who accepted me as a Ph.D. student in his group in Karlsruhe after my graduation. As a physicist and mathematician I had had no previous knowledge of Artificial Intelligence or even computer science. Jörg did a great job to introduce me to the appealing ideas and interesting research problems of Artificial Intelligence and especially of Automated Deduction.

Of course I was not the only person who had been attracted by this excellent research environment. So in Karlsruhe and later in Kaiserslautern a large group of researchers had gathered, every one of which had some influence, more or less important, on this work. Of all the people in the automated deduction group I can only name a few explicitly.

I am very much indebted to Norbert Eisinger for sharing with me his profound knowledge of clause graphs, which helped to develop my intuitive feeling for this form of proof representation. He was always there to construct a counter-example to any of my favorite conjectures. Talking to Xiaorong Huang helped a lot; he had never worked in the classical parts of theorem proving, and thus — as a mathematically trained scientist — has maintained a natural feeling for the difficulty and naturalness of proofs. We had many discussions on the processing of natural deduction graphs, which made me realize what was needed to make proofs understandable. After several years of handling resolution proofs and refutation graphs, there is an apparent danger of ending up regarding these proof representations as the most obvious available.

Thanks to Christoph Weidenbach for the energy with which he helped to implement all the new ideas in the proof transformation program. He was always fast at pointing out why something couldn't work as easily as planned, which had the effect of an intensified search for better solutions. He is the only one, who really knows all the internals of the program and without his help it would not have been possible to do many of the examples. As is customary for the acknowledgements, I take upon my shoulders all the blame for any errors left in the system, both programming bugs and systematic errors.

Modern desk top publishing has taken away the chance to reproach a secretary or typist with all the errors remaining in the text of this thesis. The only thing I can say is that Dan Nesmith, Jörg Siekmann, and Rolf Socher must have overlooked them when reading earlier draft versions of this thesis. I want to thank them for their exact reading and the many suggestions they made.

Last but not least I want to give credit to the DFG (Deutsche Forschungsgemeinschaft) for financing part of the research environment in Kaiserslautern. I have worked in the SFB 314 (Sonderforschungsbereich „Künstliche Intelligenz – wissensbasierte Systeme) for almost four years and profited from the good climate of research and many internal meetings.

1 Introduction

Artificial Intelligence and especially Knowledge-Based Systems have had an increasing success in recent years. Such systems are now capable of doing more and more tasks for which intelligence of some sort is deemed necessary. To obtain better performance and to cope with their increasing complexity most systems felt the need to incorporate some form of automated inferencing system, which today form a crucial part of almost all Knowledge-Based Systems.

At the same time Automated Deduction Systems have gone through a surprisingly fast enhancement by employing specialized datastructures, better search strategies, and reasoning algorithms highly adapted to the special field they were designed for. In this context one has to name all the different unification algorithms and strategies for searching a proof which no longer "construct" a proof in single and rather simple steps, but reduce the proof to a small number of highly concentrated macro-steps. Modern automated theorem proving has even put away with intermediate proof steps altogether and tries to detect sufficient conditions for a proof by showing certain properties of specialized graph or matrix structures.

With the increasing strength of Automated Deduction Systems the length and complexity of computer generated proofs has also reached a degree where they become almost impossible to understand. To add to their incomprehensibility, almost every research group uses its own format and style of stating a proof. It may be given as a pure Resolution Proof, but when the proof is not actually found in distinct single steps, the result may be as complex as an abstract graph or a matrix with some additional conditions imposed on it such as acyclicity or the "spanning" property. This has led to a state where only specialists, and sometimes only specialists in the very method of automated reasoning, are capable to understand and check a proof found by an automated deduction system.

If this has been an obstacle for mathematicians to accept automatic help, when proving technical lemmata, or trying to find proofs interactively, it has even more hindered the explanation of results in other knowledge based systems. Expert systems for instance depend heavily on the quality of their "explanation component" if they want to be accepted for practical use. For the communication of "cooperating agents" or robots it might be sufficient to exchange information in a low-level language, but whenever contact to a human being must be made, the need of easily understandable and clearly structured arguments becomes apparent.

Therefore it is necessary to be able to represent proofs in a more abstract and better structured way. Ideally one would like the proof to be given in natural language, with a large variety of inference rules. As a preliminary step in this direction it seems to be useful to transform the computer generated proof into a proof in a natural deduction system which, although still a system of formal logic, has been devised to approximate as much as possible an intuitive form of reasoning. With this purpose in mind Stanislaw Jaskowski [Ja34] and Gerhard Gentzen [Ge35] invented calculi of natural deduction whose inference rules, as Dag Prawitz formulates in [Pr65],

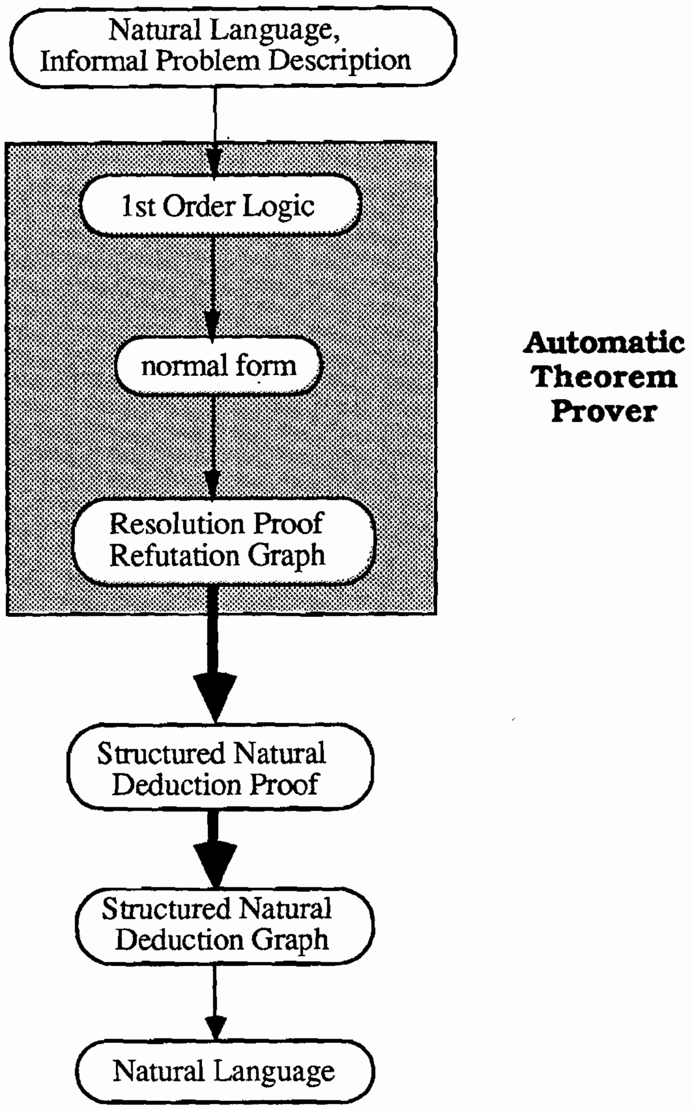
> *correspond closely to procedures common in intuitive reasoning, and when informal proofs — such as are encountered in mathematics for example — are formalized within these systems, the main structure of the informal proof can often be preserved.*

Proof Transformation is an old problem of logic, but it has been neglected in a quest for automatically finding proofs. The problem of how to search for a proof became so prevalent that the original incentive for formal arguments, that is to convince someone else of the correctness of a proposition, became almost forgotten. The main aspects of proof transformation used to be theoretical in nature, but now stylistic aspects should begin to play a more important role. Peter Andrews was the first to take up these issues again, when he proposed a method to transform matrix proofs into natural deduction proofs [An80].

The transformation of proofs into a natural deduction formulation has solved some of the problems, see [An80], [Mi83], or [Li86], but by and large the increasing length and complexity of the transformed proofs adds to their incomprehensibility rather than to reduce it. It is therefore paramount to be able to state the proofs in a hierarchically structured way, as mathematicians do, formulating subgoals and lemmata. It should also be avoided to overload the proofs with a large number of trivial steps that hide the interesting ideas of the proof.

We aim to simplify and transform proofs that are found automatically into that subset of natural language a mathematician might use. This shall be done in several steps.

The figure on the next page shows the different tasks that need to be performed when an automated deduction system tries to find a proof. Starting from an informal description of the problem or the theorem to prove, a formal (first order) formulation of the problem is generated. It is here where classical theorem provers begin their job, most of them transforming the formulae into a normal form before proving them. The output is a proof in a format that is best suited for the computer, a resolution proof for instance or a refutation graph, but not necessarily for a human.


Natural Language, Informal Problem Description
1st Order Logic
normal form
Resolution Proof Refutation Graph
Automatic Theorem Prover
Structured Natural Deduction Proof
Structured Natural Deduction Graph
Natural Language

Now proof transformation comes into the picture. The proof can then be rewritten in a natural deduction calculus. But if one is not careful to lay open the internal structure of the proof during this process, the resulting natural deduction proof is often not easier to understand than the original. Therefore, at the same time, the proof has to be reorganized and presented in a structured form using inference rules of natural deduction. From this intermediate representation where the information has been made explicit of how different parts of the proof are logically connected, one can then start to generate a proof in natural language.

In this thesis we want to examine the transformation step from refutation graphs into structured natural deduction proofs and further on into natural deduction graphs; these are the steps marked with bold arrows in the figure above. The thesis is organized in three main chapters.

In chapter 2 the formal basis for this work is defined, especially the different calculi and proof representations, as resolution proofs, refutation graphs, and as natural deduction proofs. Then some important properties of clause graphs, and more specifically of deduction and refutation graphs are developed. Finally the system of natural deduction, that has been chosen by Gerhard Gentzen for its simplicity and systematic use of the connectives, is extended by some further rules with the objective of allowing shorter proofs by avoiding long series of trivial proof steps.

Chapter 3 contains the basic system of proof transformation with a special emphasis on an "intelligent" selection of inference rules. In particular, we try to avoid proofs by contradiction, and the need to break up the formulae into single literals in the resulting natural deduction derivations.

The task of finding the underlying proof structure is presented in chapter 4. This can be accomplished by the elegant expedient of exploiting topological properties of the refutation graphs in order to come up with a well-organized proof. Structure can be imposed upon the proofs by introducing lemmata, both to avoid duplication of parts of the proof and to arrange a larger proof in a sequence of subgoals easier to understand. Another means of structuring proofs is its division into several disjoint parts by employing a case analysis. This constitutes very often the only possibility to prove an existentially quantified formula without having to fall back on a proof by contradiction.

2 Logical Calculi and Proof Representations: Resolution, Clause Graphs, Natural Deduction Systems

In this chapter we will define the logic and introduce all the basic definitions for the logical calculi used in this thesis. Everything is standard first order predicate logic, and we need resolution and a natural deduction system based on Gerhard Gentzen's calculus NK [Ge35]. Additionally, as our actual starting point of the proof transformation will not be a resolution proof, but rather the result of a graph-based theorem prover, we must introduce the representation of proofs as graphs, i.e. as so-called refutation graphs.

2.1 General Definitions

This section contains the basic definitions of the underlying logic. There are no important differences from the usual way of defining these concepts; similar definitions can for instance be found in [Ga86] or in [Lo78].

2.1-1 Definition: (signature, terms)

We define a *signature* $\mathbb{F}$ as the union of the sets of *constant symbols* $\mathbb{F}_0$, and the sets $\mathbb{F}_n$ of n-ary *function symbols* (n = 1, 2, ...); all the $\mathbb{F}_n$ are finite. Let $\mathbb{V}$ be a countable set of *variable symbols*. Then the set $\mathbb{T}$ of *terms* is the smallest set with

(i) $\mathbb{V}, \mathbb{F}_0 \subseteq \mathbb{T}$

(ii) if $f \in \mathbb{F}_n$ and $t_1, t_2, \ldots, t_n \in \mathbb{T}$, then $ft_1t_2\ldots t_n \in \mathbb{T}$.

V(t) is the set of variables occurring in a term t. A term containing no variables is called a *ground term*. $\mathbb{T}_{gr}$ is the set of all ground terms. V(o) is an abbreviation for the set of variables occurring in an arbitrary object o, and the same convention is similarly used for $\mathbb{F}_n$, $\mathbb{F}$, $\mathbb{T}$, and $\mathbb{T}_{gr}$.

2.1-2 Definition: (substitutions)

A *substitution* is a mapping σ: $\mathbb{V} \to \mathbb{T}$ with finite *domain* $V := \{v \in \mathbb{V} \mid \sigma(v) \neq v\}$; $\sigma(V)$ is called the *codomain* of σ. A substitution σ with domain $\{x_1, x_2, \ldots, x_n\}$ and codomain $\{t_1, t_2, \ldots, t_n\}$ is represented as $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. A substitution is extended to a mapping $\mathbb{T} \to \mathbb{T}$ by the usual homomorphism on terms. The application of a substitution to any other object containing terms is defined analogously.

A substitution σ is *idempotent* if $\sigma \circ \sigma = \sigma$. This is equivalent to the requirement that none of the variables of its domain occurs in any of the terms of its codomain, cf. [He87]. In this thesis all substitutions will be idempotent. If a substitution maps into T_{gr}, it is called a *ground substitution,* if it is a bijection and maps into $\mathbb{V}$ it is called a *renaming*.

Let $s,t \in \mathbb{T}$. A *matcher* from s to t is a substitution μ with $\mu s = t$. A *unifier* of s and t is a substitution s with $\sigma s = \sigma t$. If a unifier for s and t exists, then the two terms are said to be *unifiable.*

2.1-3 Definition: (formulae)

We introduce the set $\mathbb{P} = \bigcup_{0 \leq n} \mathbb{P}_n$ consisting of finite sets of n-ary *predicate symbols* (n = 0, 1, ...). There are two special zero-place predicate symbols, *TRUE* (written $\top$) and *FALSE* (written $\bot$). The objects of the form $Pt_1t_2 \ldots t_n$ with $P \in \mathbb{P}_n$ and $t_1, t_2, \ldots, t_n \in \mathbb{T}$ constitute the set $\mathbb{A}$ of *atoms.*

To construct the formulae of First Order Predicate Logic, we use the following additional signs:

(a) Unary connective	$\neg$	*negation sign*
(b) Binary connectives	$\wedge$	*conjunction sign*
	$\vee$	*disjunction sign*
	$\Rightarrow$	*implication sign*
(c) Quantifiers	$\forall$	*universal quantifier*
	$\exists$	*existential quantifier*
(d) Structuring Signs	(	*opening parenthesis*
	)	*closing parenthesis*

The set Φ of *formulae* of First Order Predicate Logic is now defined as the smallest set with:

(i) $\mathbb{A} \subseteq \Phi$
(ii) If $A, B \in \Phi$, then $(A \wedge B)$, $(A \vee B)$, and $(A \Rightarrow B)$ are all in Φ.
(iii) If $A \in \Phi$ and $x \in \mathbb{V}$, then $\neg A$, $(\forall x\ A)$, and $(\exists x\ A)$ are all in Φ.

$(A \Leftrightarrow B)$ is used as an abbreviation for $(A \Rightarrow B) \wedge (B \Rightarrow A)$. Furthermore we write $(\forall x_1, x_2, \ldots, x_n A)$ as an abbreviation for $(\forall x_1 (\forall x_2 \ldots (\forall x_n A)) \ldots)$ and similarly for the existential quantifier. If $\mathcal{M} = \{M_1, M_2, \ldots, M_n\}$ is a finite set of formulae, we write $(\bigwedge \mathcal{M})$ or $(\bigwedge_{1 \leq i \leq n} M_i)$ instead of $(M_1 \wedge (M_2 \wedge \ldots \wedge (M_n)) \ldots)$ and likewise $(\bigvee \mathcal{M})$ or $(\bigvee_{1 \leq i \leq n} M_i)$ instead of $(M_1 \vee (M_2 \vee \ldots \vee (M_n)) \ldots)$.

Parentheses are only used to indicate the range of the connectives, as in $((\neg A) \wedge (B \vee C))$. The outermost parentheses will be omitted most of the time, and we adopt the usual convention to define a binding order of the connectives. We assume that $\neg$ binds more strongly than $\wedge$ and $\vee$, these in turn bind more strongly than $\Rightarrow$ and $\Leftrightarrow$, and the quantifiers $\forall$ and $\exists$ are the weakest. Parentheses may be omitted according to this binding hierarchy, so that the above formula could be written as $\neg A \wedge (B \vee C)$.

2.1-4 Definition: (rank of formulae)

We can now inductively define the *rank r* of a formula $F \in \mathbb{\Phi}$:

(i) if $A \in \mathbb{A}$, then $r(A) = r(\neg A) = 0$.

(ii) if $F,G \in \mathbb{\Phi}$, then
$$\begin{aligned} r(F \wedge G) &= r(F \vee G) = 1 + \max(r(F), r(G)),\\ r(\forall xF) &= r(\exists xF) = 1 + r(F),\\ r(\neg F) &= 1 + r(F), \text{ if } F \notin \mathbb{A}, \text{ and}\\ r(F \Rightarrow G) &= 3 + \max(r(F), r(G)). \end{aligned}$$

The implication is treated differently from the other connectives in order to make subsequent proofs easier. This definition ensures that the rank of an implication can be decreased by rewriting it as a disjunction, as $r(\neg F \vee G) < r(F \Rightarrow G)$.

2.1-5 Definition: (interpretations)

A *variable assignment* is a mapping $v\colon \mathbb{V} \to \mathbb{T}_{gr}$; v is extended to a mapping $\mathbb{A} \to \mathbb{A}_{gr}$ similar to substitutions, cf. definition 2.1-2. An *interpretation* $\mathfrak{I}$ is a pair $(\mathfrak{I}_{gr}, v)$, where $\mathfrak{I}_{gr} \subseteq \mathbb{A}_{gr}$ is a subset of the set of ground atoms and v is a variable assignment. $\mathfrak{I}$ *satisfies* an atom $A \in \mathbb{A}$ if $v(A) \in \mathfrak{I}_{gr}$ and *falsifies* A otherwise. An interpretation $\mathfrak{I}$ satisfies a compound formula H if

$H = (\neg F)$	and $\mathfrak{I}$ does not satisfy F,
$H = (F \wedge G)$	and $\mathfrak{I}$ satisfies both F and G,
$H = (F \vee G)$	and $\mathfrak{I}$ satisfies at least one of F and G,
$H = (F \Rightarrow G)$	and $\mathfrak{I}$ falsifies F or satisfies G,
$H = (\forall xFx)$	and $\mathfrak{I}$ satisfies Ft for all $t \in \mathbb{T}_{gr}$, or
$H = (\exists xFx)$	and $\mathfrak{I}$ satisfies Ft for at least one $t \in \mathbb{T}_{gr}$;

otherwise $\mathfrak{I}$ falsifies the formula. Any object is *satisfiable* if there exists an interpretation satisfying it and is *unsatisfiable* otherwise. An object satisfied by all interpretations is said to be *valid* and called a *tautology*.

A set of objects is satisfied by an interpretation $\mathfrak{I}$ iff $\mathfrak{I}$ satisfies all of its members. Often $\mathfrak{I}$ is used instead of $\mathfrak{I}_{gr}$, if the variable assignment is of no importance.

2.2 Formula Occurrences

The task of an automated deduction system is normally described as proving that a given formula $\varphi \in \Phi$ is a tautology. Most traditional automatic theorem proving systems take the formula φ, negate it, transform it into conjunctive normal form, and then prove its unsatisfiability. The proof is then stated as a resolution proof or in the form of a graph, [Sh79], or matrix, [An81] and [Bi81], but often starts directly with a normalized set of formulae for instance the clausal normal form. As a human proof usually starts from the formula as it was originally given, it is necessary to be able to relate the parts of the refutation graph to parts of the original formula.

In order to establish a well-defined connection between the original formula to be proved and the literal and clause nodes in the proof (when it is represented as a refutation graph), we need a relation between these literal nodes and the atoms occurring in the original formula. The following definitions are made in order to formalize this correspondence.

2.2-1 Definition: (subformulae, formula trees)

For any formula A, we define the set S(A) of *subformulae* of A as follows:

(α) if $A \in \mathbb{A}$, then $S(A)=\{A\}$.

(β) If A is of the form $B \wedge C$, $B \vee C$, or $B \Rightarrow C$, then $S(A)=\{A\} \cup S(B) \cup S(C)$.
B and C are called *immediate subformulae* of A.

(γ) If A is of the form $\neg B$, $\forall x\ B$, or $\exists x\ B$, then $S(A)=\{A\} \cup S(B)$.
In this case B is the only *immediate subformula* of A.

A formula A can be written as a *formula tree* $\tau(A)$, where the leaf nodes are labeled with an atom and the other nodes are labeled with a connective or quantifier, in the following way.

(i) If $A \in \mathbb{A}$, then $\tau(A)$ is the one node tree labeled with A.

(ii) If A is of the form $B*C$, $* \in \{\wedge, \vee, \Rightarrow\}$, then the root of $\tau(A)$ is labeled with $*$, and its two successors are the roots of $\tau(B)$ and $\tau(C)$, respectively.

(iii) If A is of the form $*B$, $* \in \{\neg, \forall x, \exists x\}$, then the root of $\tau(A)$ is labeled with $*$, and its only successor is the root of $\tau(B)$.

2.2-2 Definition: (formula occurrences)

A finite sequence $\omega = <\varphi_1,\varphi_2,\ldots,\varphi_n>$ of formulae is called a *formula occurrence* of a formula φ_n within a formula $\varphi_1 \in \Phi$, if every element of ω is an immediate subformula of its predecessor. ω is called an *atom occurrence* within φ_1, if in addition the last element of ω is an atom.

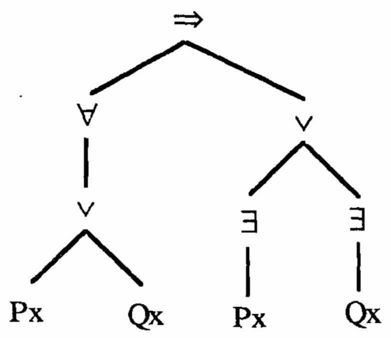
ω_1 is a *specialized formula occurrence* of ω_2 within a formula φ, $\omega_1 \supset \omega_2$, if both are formula occurrences within a common formula φ, and ω_2 is a subsequence of ω_1. $\Omega(\omega)$ and $\Omega_a(\omega)$ denote the sets of formula (atom) occurrences within φ, that are specialized formula occurrences of ω. $\Omega(<\varphi>)$ and $\Omega_a(<\varphi>)$ are sometimes simply written as $\Omega(\varphi)$ and $\Omega_a(\varphi)$.

In most cases there will be no ambiguity as to the first formula of a formula occurrence ω, but when it is important to indicate this formula, it is done using a superscript, as in ω^φ. Unlike term access functions, formula occurrences do not make a distinction between several identical immediate subformulae, as the position in $A\wedge A$, $A\vee A$, or $A\Rightarrow A$ is unimportant.

2.2-3 Example: (formula occurrences)

Let $\varphi = (\forall x\ Px \vee Qx) \Rightarrow (\exists x\ Px) \vee (\exists x\ Qx)$,
$\varphi_1 = \forall x\ Px \vee Qx$,
$\varphi_{11} = Px \vee Qx$,
$\varphi_2 = (\exists x\ Px) \vee (\exists x\ Qx)$.

$\Omega(\varphi)$ can be viewed as the set of partial paths through the formula tree of φ:



$\Omega_a(\varphi) = \{ <\varphi, \varphi_1, \varphi_{11}, Px> <\varphi, \varphi_1, \varphi_{11}, Qx>$
$<\varphi, \varphi_2, \exists xPx, Px> <\varphi, \varphi_2, \exists xQx, Qx>\}$
$\Omega(\varphi) = \Omega_a(\varphi) \cup \{<\varphi, \varphi_1, \varphi_{11}> <\varphi, \varphi_1> <\varphi>$
$<\varphi,\varphi_2, \exists xPx> <\varphi, \varphi_2, \exists xQx> <\varphi, \varphi_2>\}$

Note, that the same atom Px may be the last element of different atom occurrences.

2.3 Resolution

2.3-1 Definition: (literals, clauses)

If A is an atom, then +A and −A are (complementary) *literals*. The set of all literals is $\mathbb{L}$. A finite set of literals is called a *clause*, the number of literals in a clause C is denoted by $|C|$, and $\mathbb{C}$ is the set of all clauses. The clause without any literals is called the *empty clause* and is denoted by $\square$.

Let $\mathfrak{I}=(\mathfrak{I}_{gr}, v)$ with $\mathfrak{I}_{gr} \subseteq \mathbb{A}_{gr}$ be an interpretation. Then $\mathfrak{I}$ satisfies a literal +A iff it satisfies A, it satisfies −A iff it falsifies A. A clause C is satisfied by $\mathfrak{I}$ iff $\mathfrak{I}$ satisfies at least one literal in C.

Two literals are *unifiable* if their signs are equal and their atoms are unifiable. They are called *resolvable* whenever their signs are different and their atoms unifiable.

The *cross-product* $\times$ of two clause sets S and T is the set of clauses consisting of all the clauses combining the literals of a clause in S with the literals of a clause in T, $S \times T = \{ C \cup D \mid C \in S, D \in T \}$. From this definition we can easily derive the following distributive law needed later on:

$$S \times (T_1 \cup T_2) = (S \times T_1) \cup (S \times T_2)$$

2.3-2 Definition: (normal forms)

A formula $F \in \Phi$ is said to be in *Negation Normal Form (NNF)* if it contains no equivalence or implication signs and all its negation signs appear directly before an atom. It is in *Prenex Normal Form (PNF)* if all the quantifiers (together with the variables they bind) are placed at the beginning of the formula, i.e. before any atom or connective; the string of quantifiers (together with the variables they bind) is called the *prefix*, the quantifier-free rest of the formula is called the *matrix*. A formula F in PNF can be transformed into Skolem Normal Form (SNF); all the variables y_i bound by existential quantifiers are replaced by terms $f_i x_1 \ldots x_{n_i}$, where the function symbols f_i are distinct *Skolem functions* and the variables x_j are all the universally quantified variables that are bound before y_i in the prefix of F.

A formula $F \in \Phi$ in Skolem normal form can be transformed into a set of clauses, the so-called *clause form C(F)*, applying the following definition. Note that the construction of the clause form is not a function on interpreted formulae, but on the formulae viewed as strings.

(i) $C(L) = \{\{L\}\}$, if L is a literal.

(ii) $C(F \wedge G) = C(F) \cup C(G)$

(iii) $C(F \vee G) = C(F) \times C(G)$

A formula F is unsatisfiable iff its clause form C(F) is unsatisfiable, i.e. if there exists no interpretation simultaneously satisfying all of the clauses in C(F).

By the construction of the clause form of a formula F, a relation is established between the atom occurrences within F and the literal occurrences in the clause set C(F). The distinction between literals and occurrences of literals must be made, since a literal can be contained in several clauses. This relation will be needed when the notion of a clause graph is introduced in the next section in order to specify how a clause graph represents a formula.

2.3-3 Definition: (resolution method)

Two clauses C and D are *resolvable* if, for a renaming ρ such that C and ρD have no variables in common, there exists a pair of resolvable literals $L \in C$, $K \in \rho D$. If σ unifies the atoms of L and K, then the clause $\sigma(C \setminus L) \cup \sigma(\rho D \setminus K)$ is called a *resolvent* of C and D. In the case where $C = D$ the clause C is said to be *self-resolvable*.

A finite sequence $S_0, S_1, \ldots, S_n$ of clause sets is called a *resolution derivation* of S_n from S_0 if for all i there exist clauses C_i and D_i, such that C_i and D_i are resolvable with resolvent R_i and $S_{i+1} = S_i \cup R_i$. A resolution derivation is called a *resolution refutation* or a *resolution proof* if the final clause set S_n contains the empty clause. ♦

The resolution method for automated theorem proving relies on the fact that a clause set S is unsatisfiable if and only if there exists a resolution refutation starting with S. This means that a formula F is valid if and only if there is a resolution refutation for the clause set $C(\neg F)$.

From the beginning of the "age of resolution" in 1965, [Ro65], a major effort has been to improve on the basic resolution method by developing refinements, restriction strategies, and better datastructures to ease the search and to cut down the search space. Of the older restriction strategies, "Unit-Preference" and "Set-of-Support" [WCR64], [WCR65], "Linear Resolution" [Lo78], "First-Literal-Resolution" [KH69], and "SL-Resolution" [KK71] must be mentioned. Many of these refinements are still widely used, but today the most promising procedures are based on highly adapted datastructures. OTTER, the theorem prover developed at Argonne National Lab, [McC88], uses standard resolution, but employs a fast indexing scheme to find all the possible resolution steps. This system, though probably the most powerful automatic theorem prover to date, is still in line with the tradition of classical logic calculi, in that it derives new formulae to finally prove a theorem or refute its negation.

Modern (automated) theorem proving, however, has altogether put away with the derivation of new formulae, whether they are clauses or not. Instead, the original set of formulae, or its clause form, is arranged in a matrix or a graph structure, and finding a proof amounts to checking certain conditions in this structure. The two competing approaches are the matrix method by Andrews [An76], [An81], and Bibel [Bi81], [Bi82] and the connectiongraph method introduced by Kowalski in [Ko75].

Both methods result in an even more abstract notion of a proof than resolution does, because a proof is no longer viewed as a dynamic process leading from a set of original formulae to a desired goal formula (or a contradiction). Instead, a proof consists of a matrix or a graph containing the given set of formulae, which must be linked in a specific way. In the next section, the notion of a refutation graph will be introduced as a means to formulate proofs found with the connection graph method. These graphs are the final result of the “Markgraf Karl Refutation Procedure”, “MKRP”, a theorem prover developed in Karlsruhe and Kaiserslautern, [MKRP84], [EO88] or [OS89].

2.4 Clause Graphs and Refutation Graphs

2.4-1 Definition: (clause graph)

A *clause graph* is a quadruple $\Gamma = (\mathbb{N}, [\mathbb{N}], \pounds, \Pi)$, where

(a) $\mathbb{N}$ is a finite set. Its members are called the *literal nodes* of Γ.

(b) $[\mathbb{N}] \subset 2^{\mathbb{N}}$ is a partition of the set of literal nodes. The members of $[\mathbb{N}]$ are called the *clause nodes* of Γ. Contrary to the standard definition of a partition, $\emptyset \in [\mathbb{N}]$ is allowed. The (unique) clause node containing $L \in \mathbb{N}$ is denoted by $\lceil L \rceil$, and a clause node consisting of literal nodes L_1 through L_n is denoted by $[L_1 \ldots L_n]$.

(c) $\pounds\colon \mathbb{N} \rightarrow \mathbb{L}$ is a mapping, which labels the literal nodes with literals, such that if $L, K \in \mathbb{N}$ belong to different clause nodes, then $V(\pounds L) \cap V(\pounds K) = \emptyset$.

(d) The set of *polylinks* $\mathbb{\Pi}$ is a partition of a subset of $\mathbb{N}$, such that for all $\Lambda \in \mathbb{\Pi}$ the following polylink condition holds:

(π_1) All the literal nodes in one polylink are labeled with literals whose atoms are unifiable.

(π_2) There must be at least one positive and one negative literal in a polylink.

Literal nodes belonging to no polylink at all are called *pure*; $\mathbb{N}_p$ is the set of all pure literal nodes. Each polylink Λ has two opposite *shores*, a *positive shore* $S^+(\Lambda)$, and a *negative shore* $S^-(\Lambda)$, constituted by the literal nodes with positive and negative literals, respectively. As a literal node belongs to at most one polylink, it is possible to use $\Lambda(N)$ to denote this polylink;.if $N \in \mathbb{N}_p$ $\Lambda(N) = \emptyset$.

These clause graphs, developed by N. Eisinger in [Ei88], are a generalization of Kowalski's connection graphs, [Ko75], and Shostak's refutation graphs, [Sh76]. Unlike Eisinger we have no need for any links different from the polylinks defined above, so that we will often simply use the term link to denote a polylink. Similarly the term "graph" is used as a synonym for clause graph.

2.4-2 Definition: (interpretation of clause graphs)

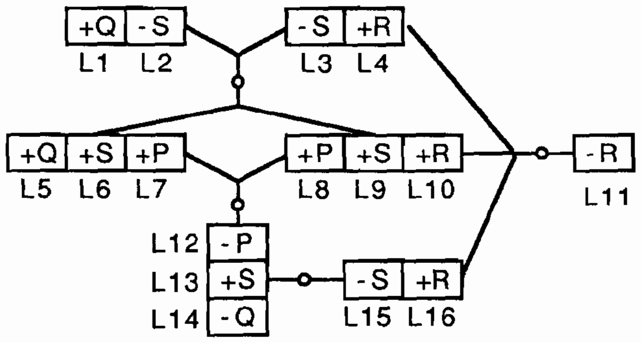
An interpretation $\mathfrak{I}$ satisfies a literal node L if it satisfies the literal £L. $\mathfrak{I}$ satisfies a clause node C if it satisfies at least one of the literal nodes $L \in C$. A clause graph Γ is satisfied by $\mathfrak{I}$ if $\mathfrak{I}$ satisfies all of its clause nodes.

A clause graph $\Gamma = (\mathbb{N}, [\mathbb{N}], £, \mathbb{\Pi})$ is said to *represent* a clause set S if for every clause node $C \in [\mathbb{N}]$ there is a *parent clause* $C' \in S$ and a ground substitution γ such that the restriction of £ to C is a bijection between its literal nodes and the literals of $\gamma C'$. ♦

Note that if a clause graph Γ representing a clause set S is unsatisfiable there is a finite unsatisfiable set of instances of clauses in S, hence S itself is unsatisfiable.

2.4-3 Example: (clause graph)

Here is an example of a clause graph. Literal nodes are drawn as boxes with the appropriate literals inside. It can be seen that the same literal may belong to several literal nodes. Therefore literal nodes cannot be identified by their literals and the labelling outside of the boxes is for their identification. The example contains seven clause nodes, built up by bordering literal nodes. There are four polylinks, {L4, L10, L16, L11}, {L2, L3, L6, L9}, {L7, L8, L12}, and {L13, L15}. Polylinks are drawn as lines with a little dot, which branch on each side to connect the different literal nodes of the opposite shores. The literal nodes L1, L5, and L14 are pure.



It is often necessary to change a given clause graph Γ by adding or removing some of its parts. Since this usually involves several sets of nodes, one has to define carefully what the resulting graph is to be. Adding a set of polylinks Ψ to a clause graph Γ means to change $\mathbb{\Pi}$ by adding a set of links consisting of previously pure literal nodes; the polylink conditions π_1 and π_2, cf. 2.4-1 (d), must of course be obeyed.

Adding a set of literal nodes means adding new pure literal nodes to one of the existing clause nodes. And to add a set of clause nodes is to insert a new set of pure literal nodes to Γ making up the new clause nodes. Since there is normally no ambiguity, all of these operations are written using the same + sign.

Similarly, to remove a set of polylinks Ξ from a clause graph Γ means to make pure all the literal nodes belonging to a link in Ξ, i.e. to add these literal nodes to $\mathbb{N}_p$. Removing a set of literal nodes $M \subseteq \mathbb{N}$ from Γ is to remove them from their respective clause nodes and to change their polylinks accordingly. The literal nodes are simply removed from their shores and, if the shore becomes empty, the whole polylink is removed. Note that if the last literal node in a clause node is removed, then the clause node is deleted altogether, rather than to keep the empty clause in the graph.

A set of clause nodes is removed by removing it from $[\mathbb{N}]$ and all of the literal nodes from $\mathbb{N}$. Removal of any part of a clause graph is written using the – sign. We will now give a rigorous definition.

2.4-4 Definition: (subgraphs)

Let $\Gamma=(\mathbb{N},[\mathbb{N}],\pounds,\amalg)$ be a clause graph, and let $D\in[\mathbb{N}]$ be a clause node in Γ. Let L be a set of literal nodes not in Γ, i.e. $L\cap\mathbb{N}=\emptyset$, and let S be a partition of the literal nodes in L. For $1\leq i\leq n$ let $\Lambda_i\subseteq\mathbb{N}_p$ with $\Lambda_i\cap\Lambda_j=\emptyset$ fulfil the polylink conditions π_1 and π_2. Let $\Psi=\{\Lambda_1,\ldots,\Lambda_n\}$. Then

$\Gamma+L^D$ $:=(\mathbb{N}\cup L,\ [\mathbb{N}]\setminus\{D\}\cup\{D\cup L\},\pounds',\amalg)$,
where $\pounds'$ is an extension of $\pounds$ with $\pounds'L\subseteq\mathbb{L}$.

$\Gamma+S$ $:=(\mathbb{N}\cup L,\ [\mathbb{N}]\cup S,\pounds',\amalg)$,
where $\pounds'$ is an extension of $\pounds$ with $\pounds'L\subseteq\mathbb{L}$.

$\Gamma+\Psi$ $:=(\mathbb{N},[\mathbb{N}],\pounds,\amalg\cup\Psi)$,

Now let $M\subseteq\mathbb{N}$ be a set of literal nodes, and let $T\subseteq[\mathbb{N}]$ be a set of clause nodes in Γ. Furthermore let $\Xi=\{\Theta_1,\ldots,\Theta_n\}\subseteq\amalg$ be a subset of the links of Γ. Then

$\Gamma-M$ $:=(\mathbb{N}\setminus M,[\mathbb{N}\setminus M],\pounds|_{\mathbb{N}\setminus M},\amalg')$ with
$[\mathbb{N}\setminus M]=([\mathbb{N}]\setminus\{\lceil N\rceil\mid N\in M\})\cup(\{\lceil N\rceil\setminus M\mid N\in M\}\setminus\{\square\})$ and
$\amalg'=(\amalg\setminus\{\Lambda(N)\mid N\in M\})$
$\quad\cup\{\Lambda(N)\setminus M\mid N\in M,\ S^+(\Lambda(N)\setminus M)\neq\emptyset,\ S^-(\Lambda(N)\setminus M)\neq\emptyset\}$

$\Gamma-T$ $:=(\mathbb{N}',[\mathbb{N}]\setminus T,\pounds|_{\mathbb{N}'},\amalg'')$ with
$\mathbb{N}'=\mathbb{N}\setminus\bigcup_{C\in T}C$, and where $\amalg''$ is constructed similar to $\amalg'$ by removing all literal nodes of clause nodes in T.

$\Gamma-\Xi$ $:=(\mathbb{N},[\mathbb{N}],\pounds,\amalg\setminus\Xi)$

Γ' is a *subgraph* of a clause graph Γ if it can be obtained from Γ by removing sets of clause nodes and polylinks.

2.4-5 Definition: (separating links)

A *walk* in a clause graph Γ is an alternating sequence $C_0\Pi_1C_1\ldots C_{n-1}\Pi_nC_n$ ($n\geq1$) of clause nodes and polylinks such that for every pair of clause nodes C_j, C_{j+1} one contains a literal node of the positive shore of the connecting polylink Π_j and the other contains a literal node of its negative shore. Regarding clause nodes and polylinks as sets of literal nodes this means for all n either $C_{n-1}\cap S^+(\Pi_n)\neq\emptyset$ and $C_n\cap S^-(\Pi_n)\neq\emptyset$ or $C_{n-1}\cap S^-(\Pi_n)\neq\emptyset$ and $C_n\cap S^+(\Pi_n)\neq\emptyset$.

A set Ψ of links is *separating* Γ if there exist two clause nodes C and D connected by a walk in Γ which are no longer connected in $\Gamma-\Psi$. If Ψ consists of only a single link Λ one can also say that Λ separates Γ.

2.4-6 Definition: (deduction and refutation graphs)

A *trail* in a clause graph Γ is a walk where all the links used are distinct. A trail *joins* its start and end clause nodes C_0 and C_n.

A *cycle* is a trail joining a clause node to itself. If a clause graph Γ contains such a cycle it is called *cyclic*, otherwise *acyclic*. It is called connected if each pair of clause nodes is joined by a trail.

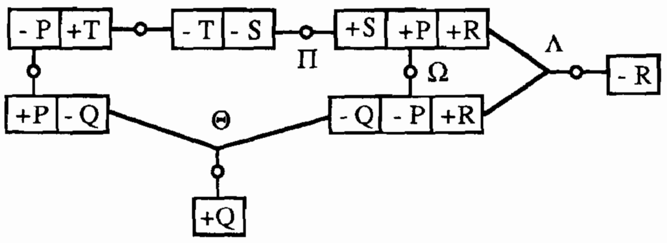
A *component* of a clause graph Γ is a maximal connected subgraph of Γ.

Let Λ and Π be polylinks in a clause graph. Λ is *less nested* than Π, $\Lambda \prec \Pi$, if there exist clause nodes C and D, containing literal nodes of the same shore of Λ, and joined by a trail using Π. $\overset{*}{\prec}$ is the reflexive and transitive closure of $\prec$.

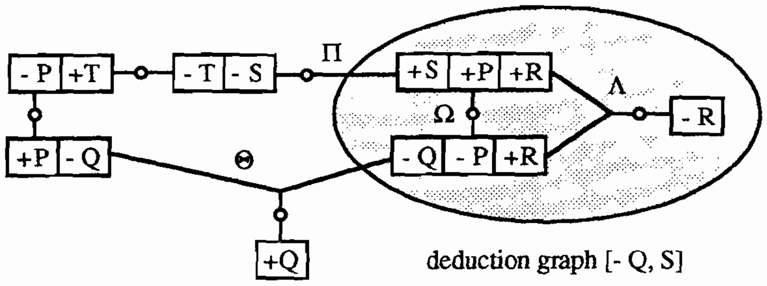
A *deduction graph* is a non-empty, ground, and acyclic clause graph. A *refutation graph* is a deduction graph without pure literal nodes. We sometimes speak of deduction or refutation graphs even if they are not ground, but then the existence of a global substitution is required that transforms them into ground graphs without destroying the polylink conditions for any of its links.

A *minimal deduction (refutation) graph* is one containing no proper subgraph which is itself a deduction (refutation) graph.

2.4-7 Example: (deduction graphs)

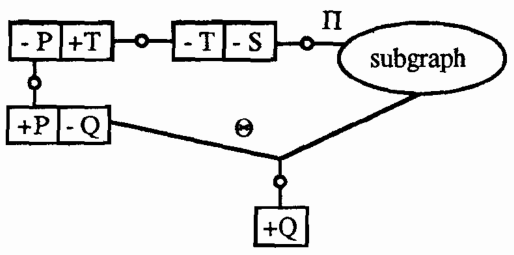


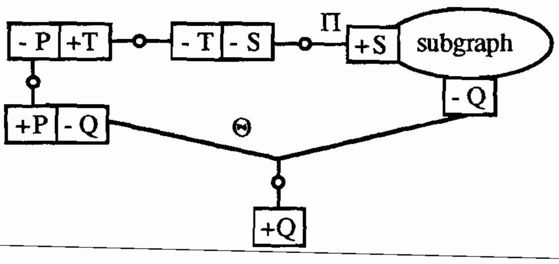
In the above graph, all the literal nodes belong to a link. Only Λ and Θ are separating the graph; they are also maximal links with respect to the nesting order $\prec$. Ω is less nested than Λ ($\Omega \prec \Lambda$), and both Ω and Π are less nested than Λ. There is no cycle, since it is not allowed to use a link more than once in a trail, and when a link is entered, it must be exited via the opposite shore. So the graph constitutes an example for a (minimal) refutation graph.



The subgraph marked in the above figure, consisting of the clause nodes [+S +P +R], [–R], and [–Q –P +R], as well as the links Λ and Ω, is an example of a deduction graph; obviously there can be no cycle, for there are no additional links, but the literal nodes marked –Q and +S are pure. The subgraph is also connected. This property could be destroyed by removing {Ω}, then there would no longer exist a trail between [+S +P +R] and [–Q –P +R].

Below the subgraphs are drawn without specifying their internal clause nodes and links. We will often do so, when the internal structure is unimportant. It is understood, however, that such subgraphs are connected and that all the links having a shore outside are indicated in the drawing. In the second case, there is a special emphasis on the nature of the pure literal nodes of the subgraph.





2.5 Properties of Deduction and Refutation Graphs

In this section we want to examine some of the properties of clause graphs, and especially deduction and refutation graphs. Most of the following lemmata have been proved by Shostak [Sh79] and Eisinger [Ei88], and will be cited here without proof. Care has been taken, however, to change the presentation of the lemmata to meet our slightly different definitions. For all the new theorems, which are needed later on for the proof transformation, proofs will of course be given.

2.5-1 Lemmata: (N. Eisinger)

The following graph theoretic properties of deduction graphs are all proved in N. Eisinger's thesis. The numbers 5_6 through 5_16 are Eisinger's original numbers for the theorems.

5_6: In a deduction graph the relation $\overset{*}{<}$ is a partial ordering.

5_7: In a deduction graph, if a link Π is $\overset{*}{<}$-minimal it is separating.

5_8: If a clause graph is acyclic, then each of its components is acyclic.

5_9: A deduction graph is connected iff it is a minimal deduction graph.

5_10: Each component of a deduction (refutation) graph is a minimal deduction (refutation) graph.

5_11: Let Π be a link in a deduction graph Γ. Then $\Gamma-\{\Pi\}$ is a deduction graph, and if Π is separating, then any two clause nodes intersecting with different shores of Π belong to different components of $\Gamma-\{\Pi\}$.

5_12: Let Π be a $\overset{*}{<}$-minimal link in a deduction graph Γ; let $\Gamma-\{\Pi\}$ have m^-, m^+, and m^0 components containing a negative, a positive, and no shore of Π, respectively, then Γ has $m^0+m^-\circ m^+$ components.

5_13: In a deduction graph with p links, c clause nodes, and m components, the inequality $p<c\leq p+m$ holds.

5_14: A deduction graph is minimal iff it has exactly one more clause node than it has links.

5_15: A deduction graph has exactly two components iff it has exactly two more clause nodes than it has links.

5_16: Let Π be a link in a minimal deduction graph Γ. Then $\Gamma-\{\Pi\}$ has exactly two components, each a minimal deduction graph containing exactly one of the shores of Π. If Π is $\overset{*}{<}$-minimal, then each of the two subgraphs is a component of $\Gamma-\{\Pi\}$. ■

An important operation on clause graphs and especially deduction graphs, which will be needed later on to construct graphs representing subproofs, is to split the graph by "cutting" a clause node into parts.

2.5-2 Definition: (graph splitting)

A clause graph $\Gamma = (\mathbb{N}, [\mathbb{N}], \pounds, \Pi)$ can be *split* by splitting a clause node $C \in [\mathbb{N}]$ into several disjoint parts leaving the link structure unchanged. With $C = C_1 \uplus C_2 \uplus \ldots \uplus C_n$ where $\uplus$ denotes the disjoint union, i.e. $C_i \cap C_j = \emptyset$ for $i \neq j$, the graph changes to $\Gamma' = (\mathbb{N}, ([\mathbb{N}] \setminus \{C\} \cup \{C_1, C_2, \ldots, C_n\}), \pounds, \Pi)$.

2.5-3 Lemma: (graph splitting)

Let $\Gamma = (\mathbb{N}, [\mathbb{N}], \pounds, \Pi)$ be a deduction graph and $C \in [\mathbb{N}]$ a clause node. If the graph is split by $C = C_1 \uplus C_2 \uplus \ldots \uplus C_n$, then

(i) all the new clause nodes C_i lie in different components of the split graph.

(ii) If Γ is minimal, there are exactly n resulting components.

(iii) All the components are themselves deduction graphs.

Proof: (i) If for any pair i, j $(i \neq j)$ C_i and C_j are in the same component of the split graph, then there must be a trail joining C_i and C_j. But since the literal nodes of C_i and C_j are in the same clause node in Γ, this proves that Γ contains a cycle, which is a contradiction to the assumption that Γ is a deduction graph.

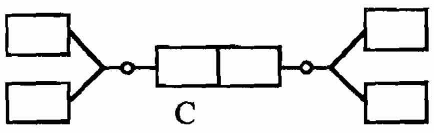
(ii) Let c be the number of clause nodes in Γ, and let p be the number of links. As the deduction graph is minimal, we know that $c=p+1$ (lemma 5_14). The proof is now conducted by induction on the number n of splitparts.

Induction base (n=2): After splitting, the deduction graph Γ' has the same number of links but one additional clause node, therefore $c'=p'+2$. By lemma 5_15 Γ' must have exactly two components.

Induction step $(n \rightarrow n+1)$: Splitting a clause node into n+1 parts can be done by splitting it into n parts, which leads to exactly n components $\Delta_1, \ldots, \Delta_n$ in the resulting deduction graph by induction hypothesis, and then further splitting the clause node C_n into two parts. From (i) we know that C_n does not lie in any of the other components, so splitting it will split Δ_n into two parts without affecting the other components. Hence the total number of splitparts is n+1.

(iii) It is easily seen that no cycle can be introduced by the splitting operation, for no links are added and no literal nodes are joined to new clause nodes. Therefore splitting a deduction graph always leads to another deduction graph. Additionally, as no link is removed, no literal node can become pure, and hence the splitparts of refutation graphs are also refutation graphs. ■

It is not the case that splitting a clause node into n parts always increases the number of components by n-1 as can be seen in the following example, where we start with four components and the graph resulting from a cut through C still has four components.



At this point it is necessary to explain how refutation graphs and deduction graphs represent (refutation) proofs and derivations. Shostak and Eisinger prove that refutation graphs are always unsatisfiable (cf. theorem 4.1 in [Sh79] or lemma 6.1_1 in [Ei88]), and therefore any clause set represented by a refutation graph must itself be unsatisfiable. So the existence of a refutation graph can guarantee the unsatisfiability of a clause set. Furthermore Eisinger shows how a resolution refutation can be constructed from a given refutation graph.

As a corollary of Eisinger's lemma 6.1_6 it is known that every interpretation satisfying a deduction graph must satisfy one of the pure literal nodes. Again a resolution derivation of the clause node consisting of the pure literal nodes from the set of all clause nodes in Γ can be constructed. In this sense a deduction graph with pure literal nodes $L_1, \ldots, L_n$ represents the derivation of $£L_1 \vee \ldots \vee £L_n$ from the underlying clause set.

Here we are not so much interested in proofs of the unsatisfiability of clause sets, but prefer to state the problem in its positive form, i.e. to give a proof of the validity of a formula F (usually not in clause form), when a refutation graph representing $C(\neg F)$ is known. One important thing to know therefore is a relation between the literal nodes in the graph and the atom occurrences of the original formula.

2.5-4 Definition: (clause graph relation)

For a formula F and a clause graph $\Gamma = (\mathbb{N}, [\mathbb{N}], \pounds, \Pi)$ representing C(F) or C(¬F), a relation $\Delta \subseteq \{(\omega_a, L) \mid \omega_a \in \Omega_a(F), L \in \mathbb{N}\}$ is a *clause graph relation* if it is compatible with the relation established by the normalization process (cf. 2.3-2) when the clause form is constructed from the formula. ♦

Δ is in general not a function, which can easily be seen when one envisages the process of constructing the clause form of a formula F. It is often useful, however, to be able to use Δ as a function to denote the set of literal nodes related to a given formula occurrence. In this sense we use Δ as a symbol for one of the two functions defined by the relation Δ,

$\Delta\colon\ \Omega_a(F) \to 2^{\mathbb{N}}$ and $\qquad \Delta\colon \mathbb{N} \to 2^{\Omega_a(F)}$

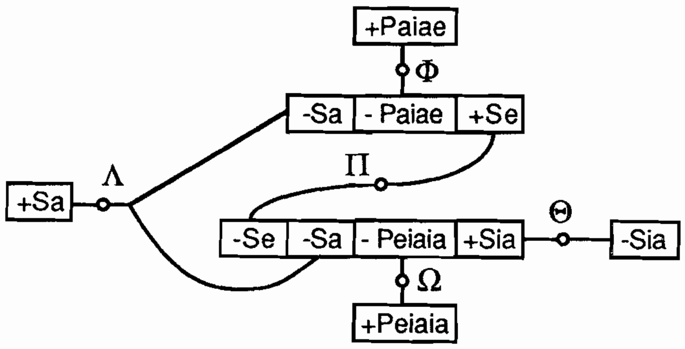
$\Delta(\omega) = \{L \in \mathbb{N} \mid (\omega_a, L) \in \Delta\} \qquad \Delta(L) = \{\omega_a \in \Omega_a(F) \mid (\omega_a, L) \in \Delta\}$

2.5-5 Example: (refutation graph for subgroup criterion)

In this example a refutation graph is given representing a proof of the formula $F = (\forall u\ Puiue) \land (\forall w\ Peww) \land (\forall xyz\ Sx \land Sy \land Pxiyz \Rightarrow Sz) \Rightarrow (\forall v\ Sv \Rightarrow Siv)$. This is a formulation of part of the subgroup criterion, see for instance [De71]:

Let G be a group, S⊆G; if for all x,y in S, $y^{-1} \circ x$ is also in S, then for every v in S its inverse is also in S.

In this formulation e represents a constant (the unit element of the group) and i a unary function (the inverse); Pxyz means that $x \circ y = z$ in the group and Sx stands for $x \in S$. The constant a is introduced during the skolemization of ¬F, which leads to a clause set $S = \{C_1, C_2, C_3, C_4, C_5\}$ with $C_1 = \{+Puiue\}$, $C_2 = \{+Peww\}$, $C_3 = \{-Sx, -Sy, -Pxiyz, +Sz\}$, $C_4 = \{+Sa\}$, and $C_5 = \{-Sia\}$.



The relation Δ is obvious in this small example. It can be seen here that Δ is not a function as both the literal nodes labeled with +Se and +Sia are related to the atom occurrence $<F, Ax_1 \wedge Ax_2 \wedge Ax_3, Ax_3, Sz>$[1]. On the other hand the literal node –Sa in the same clause node as +Se is related to both $<F, Ax_1 \wedge Ax_2 \wedge Ax_3, Ax_3, Sx>$ and $<F, Ax_1 \wedge Ax_2 \wedge Ax_3, Ax_3, Sy>$.

According to Eisinger in [Ei88], a resolution proof can be generated by repeatedly choosing a maximal link (with respect to the nesting order $\overset{*}{<}$) and performing the appropriate resolution step. Starting with link Π the following resolution proof is constructed:

C3,4 & C3,1:	add R1:	{–Sx,–Sy,–Pxiyz,–Sy',–Pziy'z',+Sz'}
R1,1 & C4,1:	add R2:	{–Sy,–Paiyz,–Sy',–Pziy'z',+Sz'}
R2,1 & C4,1:	add R3:	{–Paiaz,–Sy',–Pziy'z',+Sz'}
R3,2 & C4,1:	add R4:	{–Paiaz,–Pziaz',+Sz'}
R4,1 & C1,1:	add R5:	{–Peiaz',+Sz'}
R5,1 & C2,1:	add R6:	{+Sia}
R6,1 & C5,1:	add R7:	□

Only the first resolution step is required by Eisinger's method, all the other steps can be done in any order, so the refutation graph does not only represent a single resolution proof but a whole class of resolution proofs differing in the order of the resolution steps.

At this point we want to prove two technical lemmata regarding clause graphs and their relation to formulae. In the first lemma below the link structure of the graph is of no importance, so it can also be seen as a property of clause sets where the relation Δ is transcribed in the obvious way.

2.5-6 Lemma: (literal removal)

Let F and G be two formulae in Skolem normal form. Let Γ be a clause graph representing F∨G and Δ a clause graph relation for F∨G and Γ.

Then the graph Γ' constructed from Γ by removing all the literal nodes related only to atom occurrences in $\Omega_a(<F \vee G, G>)$ is a clause graph representing F.

$$\Gamma' = \Gamma - (\Delta(\Omega_a(<F \vee G, G>)) \setminus \Delta(\Omega_a(<F \vee G, F>)))$$

[1] We denote by Ax_1 the formula $\forall u\, Puiue$, by Ax_2 the formula $\forall w\, Peww$, and by Ax_3 the formula $\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz$.

Proof: The proof is conducted by induction on the rank of the formula G.

If $r(G) = 0$, G is a literal L and $C(G) = \{\{L\}\}$. Then Γ represents the clause set $S = C(F \vee G) = C(F) \times \{\{L\}\} = \{C \cup \{L\} \mid C \in C(F)\}$. All the clause nodes in Γ must have a parent clause in S. After removing all the literal nodes stemming only from L, all the clause nodes have a parent in C(F), so Γ' represents F.

If $r(G) = n > 0$, we consider first the case where G is a conjunction $G_1 \wedge G_2$. Then by definition $C(F \vee (G_1 \wedge G_2)) = C(F) \times C(G_1 \wedge G_2) = C(F) \times (C(G_1) \cup C(G_2))$. By the law of distributivity this equals $(C(F) \times C(G_1)) \cup (C(F) \times C(G_2)) = C(F \vee G_1) \cup C(F \vee G_2)$. Every clause node in Γ must have a parent clause in this clause set. From the graph we must then remove the literal nodes in

$$\begin{aligned}
&\Delta(\Omega_a(<F \vee G, G>)) \backslash \Delta(\Omega_a(<F \vee G, F>)) \\
= \ &(\Delta(\Omega_a(<F \vee G, G, G_1>)) \cup \Delta(\Omega_a(<F \vee G, G, G_2>))) \backslash \Delta(\Omega_a(<F \vee G, F>)) \\
= \ &(\Delta(\Omega_a(<F \vee G, G, G_1>)) \backslash \Delta(\Omega_a(<F \vee G, F>))) \cup \\
&(\Delta(\Omega_a(<F \vee G, G, G_2>)) \backslash \Delta(\Omega_a(<F \vee G, F>))).
\end{aligned}$$

We can now simply remove $\Delta(\Omega_a(<F \vee G, G, G_1>)) \backslash \Delta(\Omega_a(<F \vee G, F>))$ from the subgraph whose parent clauses are in $C(F \vee G_1)$, as the rest of the graph cannot contain any of these literal nodes. The resulting clause graph represents C(F) by induction hypothesis, for $r(G_1) < r(G)$. The same argument applies to the removal of $\Delta(\Omega_a(<F \vee G, G, G_2>)) \backslash \Delta(\Omega_a(<F \vee G, F>))$ from the subgraph whose parent clauses are in $C(F \vee G_2)$.

If $G = G_1 \vee G_2$ is a disjunction, then $C(F \vee (G_1 \vee G_2)) = C(F) \times C(G_1 \vee G_2) = C(F) \times (C(G_1) \times C(G_2)) = (C(F) \times C(G_1)) \times C(G_2) = C(F \vee G_1) \times C(G_2)$. An argument similar to the previous case allows us to remove $\Delta(\Omega_a(<F \vee G, G, G_1>)) \backslash \Delta(\Omega_a(<F \vee G, F>))$ from the subgraph of Γ representing $C(F \vee G_1)$ before computing the cross-product. The remaining clause graph represents $C(F) \times C(G_2) = C(F \vee G_2)$ by induction hypothesis. From this graph we must remove $\Delta(\Omega_a(<F \vee G, G, G_2>)) \backslash \Delta(\Omega_a(<F \vee G, F>))$, and another application of the induction hypothesis leads to a graph representing C(F). ∎

Previously Shostak [Sh79] has mentioned that there are unsatisfiable ground clause sets, for which every refutation graph contains at least one of its clauses twice. But in this case one can always inhibit the duplication of any specific clause.

2.5-7 Lemma: (choose clause not to duplicate)

For every unsatisfiable ground clause set S containing a clause C, one can construct a refutation graph Γ which contains C only once.

Proof: Let C = $\{L_1\ L_2 \dots L_n\}$, and let $S' = S \setminus \{C\}$. Then $S_i = S' \cup \{[L_i]\}$ is also unsatisfiable. For this clause set there exists a refutation graph using $[L_i]$ only once, for otherwise the copies of $[L_i]$ could be combined using a branching link. This holds for all i. We now construct a refutation graph for S by combining all the refutation graphs for S_i so that all the single graphs are only connected via C, which by construction appears only once. ■

2.6 Natural Deduction Proofs

In 1933, Gerhard Gentzen developed a formal system for mathematical proofs with the intention to describe as closely as possible the actual logical inferences used in mathematical proofs. To quote from [Ge35]:

der möglichst genau das richtige logische Schließen bei mathematischen Beweisen wiedergibt

The main difference between these natural deduction proofs (NDPs) and proofs in the earlier axiomatic systems by Frege, Russell, and Hilbert is that inferences are drawn from assumptions rather than from axioms.

Prawitz describes such systems of natural deduction in [Pr65]:

The inference rules of the systems of natural deduction correspond closely to procedures common in intuitive reasoning, and when informal proofs – such as are encountered in mathematics for example – are formalized within these systems, the main structure of the informal proofs can often be preserved.

We use a linearized form of Gentzen's calculus NK, where the dependencies between formulae are explicitly included as justifications, and where for every formula we give the set of assumption formulae it depends on. The actual form of the proof lines is taken from Andrews [An80], but they differ only in their syntax from Gentzen's rule system for NK in [Ge35].

2.6-1 Definition: (Natural Deduction Proof)

A natural deduction *proof line* consists of

(a) a finite, possibly empty set of formulae, called the *assumptions*

(b) a single formula, called *conclusion*

(c) a *justification.*

A proof line with assumptions $\mathcal{A}$, conclusion F and justification "Rule $\mathfrak{R}$" is written $\{\mathcal{A} \vdash F \text{ Rule } \mathfrak{R}\}$. Sometimes comments are given to make the proof easier to read; these comments are then written as if they were proof lines.

A finite sequence S of proof lines is a *natural deduction derivation* of a formula F from assumptions $\mathcal{A}$, if

(α) F is the conclusion of the last line of S,

(β) $\mathcal{A}$ is the set of assumptions of this last line, and

(γ) every line in S is correctly justified by one of the rules given in 2.6-2.

A proof line $\lambda = \{\mathcal{A} \vdash F \text{ Rule } \mathfrak{R}\}$ within a sequence of proof lines is correctly justified iff $\mathcal{A} \vdash F$ matches the lower part of Rule $\mathfrak{R}$ and there are proof lines before λ in the sequence matching the upper part of Rule $\mathfrak{R}$.

A finite sequence S of proof lines is a *natural deduction proof* of a formula F if it is a natural deduction derivation of F from an empty set of assumptions.

2.6-2 Rules of the Natural Deduction System:

In the following definition of the rules of the natural deduction calculus the letters F, G, and H represent formulae and $\mathcal{A}$ represents a finite set of formulae.

Assumption Rule (Ass): $\dfrac{}{\mathcal{A}, F \vdash F}$

This rule introduces a new assumption. For the intuitionistic calculus NI it replaces the axioms of other calculi. To obtain the full power of classical logic one needs to add an additional axiomatic rule, the law of the excluded middle:

Tertiam non datur (Axiom): $\dfrac{}{\mathcal{A} \vdash F \vee \neg F}$

The following rules are introduction and elimination rules for the various logical connectives. Only the rule of contradiction does not fit into this scheme.

Deduction Rule (⇒I): $$\frac{\mathcal{A}, F \vdash G}{\mathcal{A} \vdash F \Rightarrow G}$$

Modus Ponens (⇒E): $$\frac{\mathcal{A} \vdash F \quad \mathcal{B} \vdash F \Rightarrow G}{\mathcal{A}, \mathcal{B} \vdash G}$$

AND-Introduction (∧I): $$\frac{\mathcal{A} \vdash F \quad \mathcal{B} \vdash G}{\mathcal{A}, \mathcal{B} \vdash F \wedge G}$$

AND-Elimination (∧E): $$\frac{\mathcal{A} \vdash F \wedge G}{\mathcal{A} \vdash F} \quad \text{and} \quad \frac{\mathcal{A} \vdash F \wedge G}{\mathcal{A} \vdash G}$$

OR-Introduction (∨I): $$\frac{\mathcal{A} \vdash F}{\mathcal{A} \vdash F \vee G} \quad \text{and} \quad \frac{\mathcal{A} \vdash G}{\mathcal{A} \vdash F \vee G}$$

NOT-Introduction (¬I): $$\frac{\mathcal{A}, F \vdash \bot}{\mathcal{A} \vdash \neg F}$$

Rule of Double Negation (¬E): $$\frac{\mathcal{A} \vdash \neg\neg F}{\mathcal{A} \vdash F}$$

Rule of Contradiction (Contra): $$\frac{\mathcal{A} \vdash F \quad \mathcal{B} \vdash \neg F}{\mathcal{A}, \mathcal{B} \vdash \bot}$$

Rule of Cases (∨E): $$\frac{\mathcal{A} \vdash F \vee G \quad \mathcal{B}, F \vdash H \quad \mathcal{C}, G \vdash H}{\mathcal{A}, \mathcal{B}, \mathcal{C} \vdash H}$$

Universal Generalization (∀I): $$\frac{\mathcal{A} \vdash Fc}{\mathcal{A} \vdash \forall x \; Fx}$$

provided that c does not occur in any of the assumption formulae in $\mathcal{A}$, and $Fc = \{x \mapsto c\} Fx$.

Existential Generalization (∃I): $$\frac{\mathcal{A} \vdash Ft}{\mathcal{A} \vdash \exists x \; Fx}$$

where $Ft = \{x \mapsto t\} Fx$.

Universal Instantiation (∀E): $\dfrac{\mathcal{A} \vdash \forall x\ Fx}{\mathcal{A} \vdash Ft}$

where $Ft = \{x \mapsto t\}Fx$.

Rule of Choice (∃E): $\dfrac{\mathcal{A} \vdash \exists x\ Fx \qquad \mathcal{B}, Fc \vdash G}{\mathcal{A}, \mathcal{B} \vdash G}$

provided that $Fc = \{x \mapsto c\}Fx$ and c does neither occur in G, nor in ∃x Fx, nor in any of the assumption formulae in $\mathcal{A}$.

2.6-3 Example: (Natural Deduction Proof)

As an example let us prove that (∀x ¬Px) ⇒ ¬(∃y Py). Note that no axiom is introduced, so this formula is also valid in intuitionistic logic.

(1)	1	⊢	∀x ¬Px	Ass
(2)	2	⊢	Pa	Ass
(3)	3	⊢	∃y Py	Ass
(4)	1	⊢	¬Pa	∀E(1)
(5)	1, 2	⊢	⊥	Contra(2,4)
(6)	1, 3	⊢	⊥	∃E(3,5)
(7)	1	⊢	¬∃y Py	¬I(6)
(8)		⊢	(∀x ¬Px) ⇒ ¬(∃y Py)	⇒I(7)

The proof lines have numbers, which are used for two purposes:

(a) in the justification, to indicate which other lines a given line depends on, and

(b) to abbreviate an assumption formula; a number in the place of an assumption formula stands for the formula introduced by the assumption rule in the line with this number.

Note that the reasoning is done exclusively with the conclusion formulae, while the assumptions are only carried along to emphasize the interdependencies between the formulae. This is characteristic of Gentzen's natural deduction system NK, whereas the calculi of sequents, as for example Gentzen's LK, also change the formulae of the antecedent.

2.7 Derived Natural Deduction Rules

Gentzen chose his original set of natural deduction inference rules for its systematic introduction and elimination of the logical connectives. It is very cumbersome, though, to do proofs using only these rules. Therefore, we will introduce a number of derived rules in this section, which will facilitate some often recurring proof processes.

Rule of Propositional Calculus (Tau): $\dfrac{\mathcal{A}_1 \vdash F_1 \ \ldots\ \mathcal{A}_n \vdash F_n}{\mathcal{A}_1,\ldots,\ \mathcal{A}_n \vdash G}$

provided that $F_1 \wedge\ldots\wedge F_n \Rightarrow G$ is a tautology.

This rule substitutes any reasoning that can entirely be done in propositional logic. It is, however, not meant to hide complicated proof sequences, but to shorten many obvious derivations, as for instance in the example below:

2.7-1 Example: (Rule of Propositional Calculus)

In order to prove that (F∨G) follows from (¬F⇒G) one would have to go through the following derivation

(α)	$\mathcal{A}$	$\vdash$	$\neg F \Rightarrow G$	Rule $\mathfrak{R}$
(β)	$\mathcal{A}$	$\vdash$	$F \vee \neg F$	Axiom
Case 1:				
(γ)	$\mathcal{A}$, F	$\vdash$	F	Ass
(δ)	$\mathcal{A}$, F	$\vdash$	$F \vee G$	∨I(γ)
Case 2:				
(ε)	$\mathcal{A}$,¬F	$\vdash$	$\neg F$	Ass
(ζ)	$\mathcal{A}$,¬F	$\vdash$	G	⇒E(α,ε)
(η)	$\mathcal{A}$,¬F	$\vdash$	$F \vee G$	∨I(ζ)
End of Cases(1,2)				
(ϑ)	$\mathcal{A}$	$\vdash$	$F \vee G$	∨E(β,δ,η)

♦

The rule below handles negation in connection with quantifiers. Similar to the rule of propositional calculus it leads to considerably shorter proofs. A natural deduction proof of $\neg(\exists x\ H)$ from $(\forall x\ \neg H)$ is shown in example 3.2-2. The other pairs can be shown to be valid by similar proofs.

Negation Rule (Neg): $$\frac{\mathcal{A} \vdash F}{\mathcal{A} \vdash G}$$

where F equals $\neg(\forall x\ H)$, $\neg(\exists x\ H)$, $\forall x\ \neg H$, or $\exists x\ \neg H$,
and G equals $\exists x\ \neg H$, $\forall x\ \neg H$, $\neg(\exists x\ H)$, or $\neg(\forall x\ H)$,
respectively.

Another often recurring proof scheme is the isolation of a universally quantified formula from a conjunction by means of the $\wedge$-Elimination followed by a subsequent instantiation, especially if the axioms of a problem are formulated as the conjunction of a set of single axiom formulae. Similarly an instantiation of a universally quantified formula can be coupled to an $\wedge$-Elimination, mainly if only one of the subterms is needed in this instance. Therefore we state the rules

AND-Instantiation ($\wedge\forall$E): $$\frac{\mathcal{A} \vdash F \wedge \forall x\, Gx \wedge H}{\mathcal{A} \vdash Gt}$$

where $Gt = \{x \mapsto t\} Gx$.

Instantiation-AND ($\forall\wedge$E): $$\frac{\mathcal{A} \vdash \forall x\ (Fx \wedge Gx)}{\mathcal{A} \vdash Ft} \quad \text{and} \quad \frac{\mathcal{A} \vdash \forall x\ (Fx \wedge Gx)}{\mathcal{A} \vdash Gt}$$

where $Ft = \{x \mapsto t\} Fx$ and $Gt = \{x \mapsto t\} Gx$.

If a universally quantified formula is part of a disjunction it cannot be instantiated with the current set of natural deduction rules. Instead, a proof by case analysis has to be performed even if the desired result would follow from the instantiated disjunction by simple propositional reasoning. This can be remedied by allowing rule

OR-Instantiation ($\vee\forall$E): $$\frac{\mathcal{A} \vdash F \vee \forall x\, Gx \vee H}{\mathcal{A} \vdash F \vee Gt \vee H}$$

where $Gt = \{x \mapsto t\} Gx$.

3 The Transformation of Refutation Graphs into Natural Deduction Proofs

The construction of natural deduction proofs (NDPs), by humans and computers alike, is conducted in single steps. To prove any valid formula F one always starts with a line $\{\vdash F\}$. Such a line is obviously no proof, because it is not correctly justified. Now the proof is constructed by deriving subgoals until the proof is completed. In the intermediate states, called proof outlines by Andrews in [An80], one may find completed subproofs, but also others that are not yet done. To formalize the procedure of the search for such a natural deduction proof, we introduce the notion of Generalized Natural Deduction Proofs.

3.1 Definitions

3.1-1 Definition: (Generalized Natural Deduction Proof)

A finite sequence S of proof lines is called a *Generalized Natural Deduction Proof (GNDP)* of a formula F, if

(i) F is the conclusion of the last line of S,

(ii) the last line of S has no assumptions, and

(iii) every line is either justified by a rule of the calculus (see 2.4-1), or it is justified by a proof (possibly in a different calculus) of its conclusion from its assumptions.

This allows lines not correctly justified within the calculus, but it is assumed that these lines are "sound", in the sense that the formula ($\bigwedge$assumptions $\Rightarrow$ conclusion) is valid. Such lines are called *external* lines, lines justified within the calculus are called *internal.* When no external lines are present in a GNDP, it is a normal NDP.

A GNDP consisting of just one line, which is an external line without assumptions and with conclusion F, is called the *trivial GNDP* for F.

3.1-2 Example: (Generalized NDP)

In this example we give one possible generalized NDP for the formula $F = (\forall u\, Puiue) \wedge (\forall w\, Peww) \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz) \Rightarrow (\forall x\, S\, x \Rightarrow Six)$, with a constant symbol e and a unary function symbol i. This is a formulation of part of the subgroup criterion:

Let G be a group, $S \subseteq G$; if for all x,y in S, $y^{-1} \circ x$ is also in S, then for every x in S its inverse is also in S.

(1)	1	$\vdash$	$(\forall u Puiue) \land (\forall w Peww) \land (\forall xyz\ Sx \land Sy \land Pxiyz \Rightarrow Sz)$	Ass

Let a be an arbitrary constant

(2)	2	$\vdash$	Sa	Ass
(3)	1, 2	$\vdash$	Sia	π
(4)	1	$\vdash$	$Sa \Rightarrow Sia$	$\Rightarrow$I(3)
(5)	1	$\vdash$	$\forall x\ Sx \Rightarrow Six$	$\forall$I(4)
(6)		$\vdash$	$(\forall u Puiue) \land (\forall w Peww) \land (\forall xyz\ Sx \land Sy \land Pxiyz \Rightarrow Sz)$ $\Rightarrow (\forall x\ Sx \Rightarrow Six)$	$\Rightarrow$I(5)

For a proof of π see the refutation graph in example 2.5-5.

In the following we always assume that for a refutation graph Γ proving a formula φ we know a clause graph relation $\Delta \subseteq \Omega_a(\varphi) \times \mathbb{N}$ establishing a clear connection between the formula φ and the literal nodes of the refutation graph. When the proof is automatically generated by a computer, this relation has to be computed during the process of transforming φ into clause form and must be maintained throughout the search for the proof.

Whenever new formulae are introduced during the process of proof transformation, that are not subformulae of the original formula, one must make sure that its relation to the literal nodes of Γ can be computed. To achieve this, the new formulae must be anchored in φ.

3.1-3 Definition: (anchored formulae)

Two formula occurrences ω^{φ} and ω^{ψ} within formulae φ and ψ *share a tail*, if the two sequences coincide in their last m elements, $m \geq 1$.

$$\omega^{\varphi} = \langle \varphi, \varphi_1, \ldots, \varphi_n, \phi_1, \ldots, \phi_m \rangle$$

$$\omega^{\psi} = \langle \psi, \psi_1, \ldots, \psi_k, \phi_1, \ldots, \phi_m \rangle$$

A formula ψ is *anchored* in a formula φ, if all the atom occurrences within ψ share a tail with an atom occurrence within φ, i.e. for every $\omega^{\psi} \in \Omega_a(\psi)$ there is a formula occurrence $\omega^{\varphi} \in \Omega_a(\varphi)$ such that ω^{ψ} and ω^{φ} share a tail. An *anchorage* is described as a function $\nu\colon \Omega_a(\psi) \rightarrow \Omega_a(\varphi)$ that maps atom occurrences to atom occurrences.

3.1-4 Definition: (basis of a formula)

The set $\alpha(\omega) = \{\omega_a \in \Omega_a(\varphi) \mid \omega_a \supset \omega\}$ of atom occurrences under a common formula occurrence $\omega \in \Omega(\varphi)$ is called the *atomic closure* of ω.

A set $\{\omega_1,\ldots,\omega_n\}$ is said to *span* φ if $\Omega_a(\varphi) = \bigcup_{i=1}^{n} \alpha(\omega_i)$. A spanning set is called a *basis* of φ if it contains no pair of formula occurrences ω_i, ω_j with $\omega_i \supset \omega_j$.

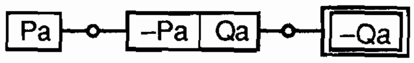
3.1-5 Lemma: (anchored if tails are shared for a basis)

Let φ and ψ be formulae and $\{\omega_1^\psi, \ldots, \omega_n^\psi\} \subseteq \Omega(\psi)$ a basis of ψ. Then ψ is anchored in φ if for all ω_i^ψ there is a formula occurrence $\omega_i^\varphi \in \Omega(\varphi)$ which shares a tail with ω_i^ψ. ♦

3.1-6 Definition: (Polarization of Clause Nodes)

Given a refutation graph Γ justifying an external line α of a GNDP with assumptions A_i, conclusion F, and a clause graph relation Δ, relating all the literal nodes of Γ to atom occurrences within $\varphi := A_1 \wedge A_2 \wedge \ldots \wedge A_n \Rightarrow F$. Then a clause node is *positively polarized* if all of its literal nodes are related to atom occurrences which are specializations of $<\varphi, A_1 \wedge A_2 \wedge \ldots \wedge A_n>$. Otherwise the clause node is said to be *negatively polarized.*

If a refutation graph is drawn and the polarization of its clause nodes must be emphasized, then a negatively polarized clause node is drawn with a double box as in



3.2 Basic Set of Transformation Rules

In order to find a natural deduction proof of a formula F, a finite sequence of generalized NDPs can be constructed whose first element is the trivial GNDP for F, and whose last element is a natural deduction proof of F. To be able to generate such a sequence of GNDPs it is necessary to describe the rules by which a GNDP is constructed from its predecessor in the sequence. In the following example such a sequence is shown for the NDP of the valid formula $(\forall x\ Px \Rightarrow Qx) \wedge Pa \Rightarrow \exists y\ Qy$. This example should shed some light on the nature of the transition rules.

3.2-1 Example: (Proof Transformation)

We start with the trivial GNDP for the formula to be shown. During this example it is assumed that the proofs π_i are always known.

GNDP 1:

(7)		$\vdash$	$(\forall x\ Px \Rightarrow Qx) \wedge Pa \Rightarrow \exists y\ Qy$	π_1

To prove this implication, we are entitled to assume its left hand side. The proof of the right hand side together with the deduction rule will then complete the proof..

GNDP 2:

(1)	1	$\vdash$	$(\forall x\ Px \Rightarrow Qx) \wedge Pa$	Ass
(6)	1	$\vdash$	$\exists y\ Qy$	π_2
(7)		$\vdash$	$(\forall x\ Px \Rightarrow Qx) \wedge Pa \Rightarrow \exists y\ Qy$	$\Rightarrow$I(6)

To show the existence of Qy, a witness must be found.

GNDP 3:

(1)	1	$\vdash$	$(\forall x\ Px \Rightarrow Qx) \wedge Pa$	Ass
(5)	1	$\vdash$	Qa	π_3
(6)	1	$\vdash$	$\exists y\ Qy$	$\exists$I(5)
(7)		$\vdash$	$(\forall x\ Px \Rightarrow Qx) \wedge Pa \Rightarrow \exists y\ Qy$	$\Rightarrow$I(6)

Now a subformula containing Q is isolated from an internal formula by applying propositional rules.

GNDP 4:

(1)	1	$\vdash$	$(\forall x\ Px \Rightarrow Qx) \wedge Pa$	Ass
(2)	1	$\vdash$	$(\forall x\ Px \Rightarrow Qx)$	$\wedge$E(1)
(5)	1	$\vdash$	Qa	π_4
(6)	1	$\vdash$	$\exists y\ Qy$	$\exists$I(5)
(7)		$\vdash$	$(\forall x\ Px \vee Qx) \wedge Pa \Rightarrow \exists y\ Qy$	$\Rightarrow$I(6)

The next step is to instantiate the universally quantified formula, such that Qa appears in it.

<u>GNDP 5:</u>

(1)	1	$\vdash$	$(\forall x\ Px \Rightarrow Qx) \land Pa$	Ass
(2)	1	$\vdash$	$(\forall x\ Px \Rightarrow Qx)$	$\land$E(1)
(3)	1	$\vdash$	$Pa \Rightarrow Qa$	$\forall$E(2)
(5)	1	$\vdash$	Qa	π_5
(6)	1	$\vdash$	$\exists y\ Qy$	$\exists$I(5)
(7)		$\vdash$	$(\forall x\ Px \Rightarrow Qx) \land Pa \Rightarrow \exists y\ Qy$	$\Rightarrow$I(6)

Now Qa holds, if Pa can be shown.

<u>GNDP 6:</u>

(1)	1	$\vdash$	$(\forall x\ Px \Rightarrow Qx) \land Pa$	Ass
(2)	1	$\vdash$	$(\forall x\ Px \Rightarrow Qx)$	$\land$E(1)
(3)	1	$\vdash$	$Pa \Rightarrow Qa$	$\forall$E(2)
(4)	1	$\vdash$	Pa	π_6
(5)	1	$\vdash$	Qa	$\Rightarrow$E(3,4)
(6)	1	$\vdash$	$\exists y\ Qy$	$\exists$I(5)
(7)		$\vdash$	$(\forall x\ Px \Rightarrow Qx) \land Pa \Rightarrow \exists y\ Qy$	$\Rightarrow$I(6)

And Pa follows directly from (1) by $\land$-Elimination, which leads to the desired natural deduction proof.

<u>GNDP 7 (NDP):</u>

(1)	1	$\vdash$	$(\forall x\ Px \Rightarrow Qx) \land Pa$	Ass
(2)	1	$\vdash$	$(\forall x\ Px \Rightarrow Qx)$	$\land$E(1)
(3)	1	$\vdash$	$Pa \Rightarrow Qa$	$\forall$E(2)
(4)	1	$\vdash$	Pa	$\land$E(1)
(5)	1	$\vdash$	Qa	$\Rightarrow$E(3,4)
(6)	1	$\vdash$	$\exists y\ Qy$	$\exists$I(5)
(7)		$\vdash$	$(\forall x\ Px \Rightarrow Qx) \land Pa \Rightarrow \exists y\ Qy$	$\Rightarrow$I(6)

There are three largely different groups of transformation rules. Rules of the first group, *external rules,* insert a justification within the calculus for a previously external line, and insert one or more external lines with different conclusion formulae. It is in fact a form of backward reasoning, and internal lines are not affected. Examples for this type are the transitions between $GNDP_1$ and $GNDP_2$ or between $GNDP_2$ and $GNDP_3$.

For the second type, *mixed rules,* both internal and external lines are used to change the GNDP. This type has been used in the construction of $GNDP_6$ from $GNDP_5$. The conclusion formula of the external line(s) remains unaltered, but the set of assumption formulae is changed.

Rules of the third group, *internal rules,* only apply rules of the calculus to internal lines, deducing further internal lines, but do not affect any external line. This last type of rules uses axioms or previously proved formulae to derive new ones by means of forward reasoning. This type of rules induces forward reasoning, either simply in propositional logic or for instance by computing instances of universally quantified formulae.

In the following sections we shall give a formal account of these transition rules. In their description, $\mathcal{A}$ is a list of assumption formulae, capital letters indicate single formulae, small Greek letters are used as labels for the lines, the justification Rule $\mathfrak{R}$ stands for an arbitrary rule of the natural deduction calculus, and the justifications π, π', π_1, and π_2 represent proofs of the respective lines. For all these rules one must make sure that the proofs π', π_1, or π_2 can be constructed from π or are otherwise known. How this can be done if the proof is given in form of a refutation graph will be shown later, when the automatic proof transformation procedure is described.

3.2-2 External Rules

The transformation rules are to be read as follows: the lines on the left hand side of the arrow ($\rightarrowtriangle$) are replaced by those on the right hand side in the next generalized NDP of the sequence.

<u>E∧:</u>

$$(\gamma)\ \mathcal{A}\ \vdash\ F\wedge G \qquad \pi \qquad \rightarrowtriangle \qquad \left\{\begin{array}{llllr} (\alpha) & \mathcal{A} & \vdash & F & \pi_1 \\ (\beta) & \mathcal{A} & \vdash & G & \pi_2 \\ (\gamma) & \mathcal{A} & \vdash & F\wedge G & \wedge I(\alpha,\beta) \end{array}\right.$$

<u>E∨1:</u>

$$(\delta)\ \mathcal{A}\ \vdash\ F\vee G \qquad \pi \qquad \rightarrowtriangle \qquad \left\{\begin{array}{llllr} (\alpha) & \mathcal{A}, \neg F & \vdash & \neg F & \text{Ass} \\ (\beta) & \mathcal{A}, \neg F & \vdash & G & \pi' \\ (\gamma) & \mathcal{A} & \vdash & \neg F\Rightarrow G & \Rightarrow I(\beta) \\ (\delta) & \mathcal{A} & \vdash & F\vee G & \text{Tau}(\gamma) \end{array}\right.$$

<u>E⇒:</u>

$$(\gamma)\ \mathcal{A}\ \vdash\ F\Rightarrow G \qquad \pi \qquad \rightarrowtriangle \qquad \left\{\begin{array}{llllr} (\alpha) & \mathcal{A}, F & \vdash & F & \text{Ass} \\ (\beta) & \mathcal{A}, F & \vdash & G & \pi' \\ (\gamma) & \mathcal{A} & \vdash & F\Rightarrow G & \Rightarrow I(\beta) \end{array}\right.$$

E∀:

(β) $\mathcal{A} \vdash \forall x Fx$ π $\rightarrow$

Let c be an arbitrary constant				
(α)	$\mathcal{A}$	$\vdash$	Fc	π'
(β)	$\mathcal{A}$	$\vdash$	$\forall x Fx$	$\forall I(\alpha)$

c must be a "new" constant, not occurring in $\mathcal{A}$ or Fx.

E∃:

(β) $\mathcal{A} \vdash \exists x Fx$ π $\rightarrow$

(α)	$\mathcal{A}$	$\vdash$	$\neg\forall x\neg Fx$	π'
(β)	$\mathcal{A}$	$\vdash$	$\exists x Fx$	$\exists G(\alpha)$

E¬:

(γ) $\mathcal{A} \vdash \neg F$ π $\rightarrow$

(α)	$\mathcal{A}, F$	$\vdash$	F	Ass
(β)	$\mathcal{A}, F$	$\vdash$	$\bot$	π'
(γ)	$\mathcal{A}$	$\vdash$	$\neg F$	$\neg I(\beta)$

All these external rules are *sound* in the sense that a GNDP is always transformed into a GNDP, i.e. if π exists, then the resulting proofs π', π_1, and π_2 (in any correct calculus for first order logic) are guaranteed to exist as well.

3.2-3 Mixed Rules

M-Cases:

(α)	$\mathcal{A}$	$\vdash$	$F \vee G$	Rule $\mathfrak{R}$
(ζ)	$\mathcal{A}$	$\vdash$	H	π

$\rightarrow$

(α)	$\mathcal{A}$	$\vdash$	$F \vee G$	Rule $\mathfrak{R}$
We consider separately the cases of (α)				
Case 1:				
(β)	$\mathcal{A}, F$	$\vdash$	F	Ass
(γ)	$\mathcal{A}, F$	$\vdash$	H	π_1
Case 2:				
(δ)	$\mathcal{A}, G$	$\vdash$	G	Ass
(ε)	$\mathcal{A}, G$	$\vdash$	H	π_2
End of cases (1, 2) of (α)				
(ζ)	$\mathcal{A}$	$\vdash$	H	$\vee E(\alpha,\gamma,\varepsilon)$

M-Choose:

(α)	$\mathcal{A}$	$\vdash$	$\exists x Fx$	Rule $\mathfrak{R}$
(δ)	$\mathcal{A}$	$\vdash$	G	π

$\rightarrow$

(α)	$\mathcal{A}$	$\vdash$	$\exists x Fx$	Rule $\mathfrak{R}$
(β)	$\mathcal{A}, Fc$	$\vdash$	Fc	Ass
(γ)	$\mathcal{A}, Fc$	$\vdash$	G	π'
(δ)	$\mathcal{A}$	$\vdash$	G	$\exists E(\alpha,\gamma)$

c may not occur in $\mathcal{A}$, Fx, or G.

3.2-4 Internal Rules

All of these deducing rules use internal lines and add a new internal line to the generalized proof. Here the new lines are marked with an arrow, "$\rightarrow$", and written below their parent lines. Rule $\mathfrak{R}$ can be any rule of the natural deduction calculus.

<u>I$\perp$:</u>

	(α)	$\mathcal{A}$	$\vdash$	F	Rule $\mathfrak{R}$
	(β)	$\mathcal{A}$	$\vdash$	$\neg F$	Rule $\mathfrak{R}'$
$\rightarrow$	(γ)	$\mathcal{A}$	$\vdash$	$\perp$	Contra(α,β)

<u>I$\wedge$left:</u>

	(α)	$\mathcal{A}$	$\vdash$	$F \wedge G$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	F	$\wedge$E(α)

<u>I$\wedge$right:</u>

	(α)	$\mathcal{A}$	$\vdash$	$F \wedge G$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	G	$\wedge$E(α)

<u>I$\Rightarrow$:</u>

	(α)	$\mathcal{A}$	$\vdash$	$F \Rightarrow G$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	$\neg F \vee G$	Tau(α)

<u>I$\neg\wedge$:</u>

	(α)	$\mathcal{A}$	$\vdash$	$\neg(F \wedge G)$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	$\neg F \vee \neg G$	Tau(α)

<u>I$\neg\vee$:</u>

	(α)	$\mathcal{A}$	$\vdash$	$\neg(F \vee G)$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	$\neg F \wedge \neg G$	Tau(α)

<u>I$\neg\Rightarrow$:</u>

	(α)	$\mathcal{A}$	$\vdash$	$\neg(F \Rightarrow G)$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	$F \wedge \neg G$	Tau(α)

<u>I$\neg\neg$:</u>

	(α)	$\mathcal{A}$	$\vdash$	$\neg(\neg F)$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	F	$\neg$E(α)

<u>I$\neg\forall$:</u>

	(α)	$\mathcal{A}$	$\vdash$	$\neg(\forall x Fx)$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	$\exists x(\neg Fx)$	Neg(α)

<u>I$\neg\exists$:</u>

	(α)	$\mathcal{A}$	$\vdash$	$\neg(\exists x Fx)$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	$\forall x(\neg Fx)$	Neg(α)

<u>I$\forall$:</u>

	(α)	$\mathcal{A}$	$\vdash$	$\forall x Fx$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	Ft	$\forall$E(α)

for an arbitrary term t.

3.2-5 Lemma: (Completeness of the Rule System)

If there exists an ND-derivation of a formula $F \in \Phi$ from assumptions $A_1, A_2, \ldots, A_n$, then there is a finite sequence of GNDPs starting with

(α_1)	A_1	$\vdash$	A_1	Ass
(α_2)	A_2	$\vdash$	A_2	Ass
	$\ldots$			
(α_n)	A_n	$\vdash$	A_n	Ass
(ω)	$A_1, A_2, \ldots, A_n$	$\vdash$	F	π

and ending with an ND-derivation for F, such that every element in the sequence is derived from its predecessor by application of one of the transformation rules above.

Proof: The proof is by induction on the rank of the formula to be derived.

We will first consider the case where $r(F) = 0$; this is the case if F is an atom or the negation of an atom. In this case we have to use induction on the maximum rank r_{max} of previously derived (or assumed) formulae that can still be needed in the proof. If $r_{max} = 0$, then F must be one of the previously derived (or assumed) formulae, in which case the proposition holds trivially. Let $r_{max} > 0$ be the rank of the maximal derived formula G_{max}. We must then show that a rule can be applied such that r_{max} decreases. If G_{max} is a conjunction, I$\wedge$left and I$\wedge$right can be applied and the proof can be conducted using only the two resulting subformulae with strictly smaller rank. If G_{max} is a disjunction, M-Cases can be applied, and in both of the resulting subproofs the disjunction is no longer needed, as one of its parts (with smaller rank) is an assumption. If the top symbol of G_{max} is a universal quantifier, I$\forall$ can be applied as often as needed (finitely many times), and the proof can be conducted using only the results obtained. All the results have strictly smaller rank. If the top symbol of G_{max} is an existential quantifier, M-Choose can be applied, thus adding an extra assumption, but lowering the maximal rank. If G_{max} is the negation of a negation then I$\neg\neg$ removes two negations leading to an equivalent formula of smaller rank; if G_{max} is the negation of a composed formula the negation can be moved inside by the appropriate variant of I$\neg$ leading to an equivalent formula of equal rank that is not an implication. Finally, if G_{max} is an implication $A \Rightarrow B$, it can be transformed into $\neg A \vee B$; now $r(\neg A \vee B) = 1+\max(r(\neg A), r(B)) \leq 2+\max(r(A), r(B)) < 3+\max(r(A), r(B)) = r(A \Rightarrow B)$. This completes the base case of the induction.

If $r(F) > 0$, Then E∧, E∨, E⇒, E∀, and E¬ easily handle the cases where F is a conjunction, a disjunction, an implication, a universal quantification, or a negation, respectively. If the top symbol of F is an existential quantifier, E∃ followed by E¬ does the job, after which ⊥ (FALSE) with rank 0 (a contradiction) has to be shown.

■

The completeness of the rule system for natural deduction proofs is now a simple special case of lemma 3.2-5, where the set of assumptions is empty. Therefore:

3.2-6 Corollary: (Completeness of the Rule System)

For any valid formula $F \in \Phi$ there is a finite sequence $<g_1, g_2, \dots g_n>$ of GNDPs starting with the trivial GNDP for F and ending with a natural deduction proof of F, such that every element in the sequence is derived from its predecessor by application of one of the transformation rules above.

3.3 Additional Transformation Rules

The procedure to construct natural deduction proofs that is suggested by the completeness proof in the previous section is actually very similar to the construction of a tableau proof, see [Sm68]. The resulting proofs suffer from two major stylistic shortcomings, though. For once, almost all the composed formulae have to be broken up until the literal level is reached. This problem can be partly solved by avoiding the division into subgoals whenever it seems possible to derive a composed formula as a whole. A method to do just that is proposed in section 3.4.

Secondly, the relatively rich choice of inference rules inherent in the original natural deduction system is much reduced when the search strategy imposed by the rule system is applied. This is very annoying, because it takes away one of the most important features of natural deduction and tends to produce too many proofs by contradiction. The only remedy for that problem seems to be the introduction of additional transformation rules, allowing a broader choice of rules during the construction of a proof.

Whenever a negation (of a composed formula) has to be proved, the basic system of transformation rule forces a proof by contradiction. This is not necessary in most cases, and can be dealt with by introducing the following transformation rules:

E¬¬:

$$(\beta)\ \mathcal{A} \vdash \neg\neg F \quad \pi \quad \longrightarrow \quad \left\{\begin{array}{lllr} (\alpha) & \mathcal{A} & \vdash F & \pi' \\ (\beta) & \mathcal{A} & \vdash \neg\neg F & \mathrm{Tau}(\alpha) \end{array}\right.$$

E¬∧:

$$(\beta)\ \mathcal{A} \vdash \neg(F\wedge G) \quad \pi \quad \longrightarrow \quad \left\{\begin{array}{lllr} (\alpha) & \mathcal{A} & \vdash \neg F\vee\neg G & \pi' \\ (\beta) & \mathcal{A} & \vdash \neg(F\wedge G) & \mathrm{Tau}(\alpha) \end{array}\right.$$

E¬∨:

$$(\beta)\ \mathcal{A} \vdash \neg(F\vee G) \quad \pi \quad \longrightarrow \quad \left\{\begin{array}{lllr} (\alpha) & \mathcal{A} & \vdash \neg F\wedge\neg G & \pi' \\ (\beta) & \mathcal{A} & \vdash \neg(F\vee G) & \mathrm{Tau}(\alpha) \end{array}\right.$$

E¬⇒:

$$(\beta)\ \mathcal{A} \vdash \neg(F\Rightarrow G) \quad \pi \quad \longrightarrow \quad \left\{\begin{array}{lllr} (\alpha) & \mathcal{A} & \vdash F\wedge\neg G & \pi' \\ (\beta) & \mathcal{A} & \vdash \neg(F\Rightarrow G) & \mathrm{Tau}(\alpha) \end{array}\right.$$

E¬∃:

$$(\beta)\ \mathcal{A} \vdash \neg\exists xFx \quad \pi \quad \longrightarrow \quad \left\{\begin{array}{lllr} (\alpha) & \mathcal{A} & \vdash \forall x\neg Fx & \pi' \\ (\beta) & \mathcal{A} & \vdash \neg\exists xFx & \mathrm{Neg}(\alpha) \end{array}\right.$$

All of these external rules handling negation are sound and can always be used if applicable.

There are three further rules which can be applied when a disjunction has to be shown. The first one, E∨2, is a symmetric variant of E∨1. There is nothing to choose between E∨1 and E∨2, so that the choice can be done entirely for stylistic reasons. The other two, E∨left and E∨right, are obviously not sound, as they proceed to a stronger proposition. But if the stronger proposition actually holds, the ensuing proof is much more natural. The same is the case for E∃-constructive.

E∨2:

$$(\delta)\ \mathcal{A} \vdash F\vee G \quad \pi \quad \longrightarrow \quad \left\{\begin{array}{lllr} (\alpha) & \mathcal{A},\neg G & \vdash \neg G & \mathrm{Ass} \\ (\beta) & \mathcal{A},\neg G & \vdash F & \pi' \\ & & & \\ (\gamma) & \mathcal{A} & \vdash \neg G\Rightarrow F & \Rightarrow\!\mathrm{I}(\beta) \\ (\delta) & \mathcal{A} & \vdash F\vee G & \mathrm{Tau}(\gamma) \end{array}\right.$$

E∨left:

$$(\beta)\ \mathcal{A} \vdash F\vee G \quad \pi \quad \longrightarrow \quad \left\{\begin{array}{lllr} (\alpha) & \mathcal{A} & \vdash F & \pi' \\ (\beta) & \mathcal{A} & \vdash F\vee G & \vee\mathrm{E}(\alpha) \end{array}\right.$$

E∨right:

$$(\beta)\ \mathcal{A} \vdash F\vee G \quad \pi \quad \longrightarrow \quad \left\{ \begin{array}{llll} (\alpha) & \mathcal{A} & \vdash G & \pi' \\ (\beta) & \mathcal{A} & \vdash F\vee G & \vee E(\alpha) \end{array} \right.$$

E∃-constructive:

$$(\beta)\ \mathcal{A} \vdash \exists x Fx \quad \pi \quad \longrightarrow \quad \left\{ \begin{array}{llll} (\alpha) & \mathcal{A} & \vdash Ft & \pi' \\ (\beta) & \mathcal{A} & \vdash \exists x Fx & \exists G(\alpha) \end{array} \right.$$

If the formula to be shown is the negation of a universal quantification and a term can be constructed constituting a counterexample, the negation can be moved inside in combination with rule E∃-constructive; this leads to

E¬∀:

$$(\gamma)\ \mathcal{A} \vdash \neg\forall x Fx \quad \pi \quad \longrightarrow \quad \left\{ \begin{array}{llll} (\alpha) & \mathcal{A} & \vdash \neg Ft & \pi' \\ (\beta) & \mathcal{A} & \vdash \exists x \neg Fx & \exists I(\alpha) \\ (\gamma) & \mathcal{A} & \vdash \neg\forall x Fx & Neg(\beta) \end{array} \right.$$

The external rules can be divided into two categories. E∧, E⇒, E∀, E¬¬, E¬∧, E¬∨, E¬⇒, and E¬∃ are called *automatic,* because no further information is needed for their application. The four variations of the rule E∨, as well as E∃, E¬∀, E∃-constructive, and E¬ may only be applied with the permission of the user or with additional information, for example if the term t in the case of E∃-constructive is explicitly given. These external rules are therefore called *user-guided,* or *proof-driven,* if the transformation is governed by a proof such as a refutation graph previously established.

M-Unless is useful as a variation of M-Cases if one of the cases is trivial. It is sound and simply adds an additional assumption.

M-Unless:

$$\left. \begin{array}{llll} (\alpha) & \mathcal{A} & \vdash F\vee G & \text{Rule } \mathfrak{R} \\ (\zeta) & \mathcal{A} & \vdash G & \pi \end{array} \right\} \longrightarrow \quad \left\{ \begin{array}{llll} (\alpha) & \mathcal{A} & \vdash F\vee G & \text{Rule } \mathfrak{R} \\ (\beta) & \mathcal{A}, F & \vdash F & \text{Ass} \\ (\gamma) & \mathcal{A}, F & \vdash G & \pi' \\ (\varepsilon) & \mathcal{A} & \vdash F\Rightarrow G & \Rightarrow I(\gamma) \\ (\zeta) & \mathcal{A} & \vdash G & Tau(\alpha,\varepsilon) \end{array} \right.$$

Often it is necessary to prove a theorem using case analysis, assuming a formula G in one case and its negation ¬G in the other. This is formalized by

<u>M-Divide:</u>

$$(\zeta)\ \mathcal{A}\ \vdash F \qquad \pi \qquad \longrightarrow \qquad \left\{\begin{array}{llll}
(\alpha) & \mathcal{A} & \vdash G \vee \neg G & \text{Axiom} \\
\multicolumn{4}{l}{\text{We consider separately the cases of } (\alpha)} \\
\multicolumn{4}{l}{\underline{\text{Case 1:}}} \\
(\beta) & \mathcal{A}, G & \vdash G & \text{Ass} \\
(\gamma) & \mathcal{A}, G & \vdash F & \pi_1 \\
\multicolumn{4}{l}{\underline{\text{Case 2:}}} \\
(\delta) & \mathcal{A}, \neg G & \vdash \neg G & \text{Ass} \\
(\varepsilon) & \mathcal{A}, \neg G & \vdash F & \pi_2 \\
\multicolumn{4}{l}{\text{End of cases (1, 2) of } (\alpha)} \\
(\zeta) & \mathcal{A} & \vdash F & \vee\text{E}(\alpha,\gamma,\varepsilon)
\end{array}\right.$$

M-Divide is obviously sound, and could therefore always be applied, but its automatic application is hindered by the need to find a suitable formula G. In many cases, M-Divide can be used instead of M-Cases, when there is no suitable disjunction that can be used for a case analysis.

The following rule, M-Inf, formalizes a common type of inference. If it is possible to prove the left hand side of a (known) implication, then the right hand side also holds. This rule is, of course, not sound, as there is no reason to believe that F can always be shown. Therefore it can only be applied when the validity of F (with assumptions $\mathcal{A}$) is known before.

<u>M-Inf:</u>

$$\left.\begin{array}{llll}
(\alpha) & \mathcal{A} & \vdash F \Rightarrow G & \text{Rule } \mathfrak{R} \\
(\gamma) & \mathcal{A} & \vdash G & \pi
\end{array}\right\} \longrightarrow \left\{\begin{array}{llll}
(\alpha) & \mathcal{A} & \vdash F \Rightarrow G & \text{Rule } \mathfrak{R} \\
(\beta) & \mathcal{A} & \vdash F & \pi' \\
(\gamma) & \mathcal{A} & \vdash G & \Rightarrow\text{E}(\alpha,\beta)
\end{array}\right.$$

Similar to rule Tau in the Natural Deduction Calculus, one can define a transformation rule I-Tau combining all possible derivations in propositional logic.

<u>I-Tau:</u>

$$\begin{array}{lllll}
 & (\alpha_1) & \mathcal{A}_1 & \vdash F_1 & \text{Rule } \mathfrak{R}_1 \\
 & (\alpha_2) & \mathcal{A}_2 & \vdash F_2 & \text{Rule } \mathfrak{R}_2 \\
 & (\alpha_n) & \mathcal{A}_n & \vdash F_n & \text{Rule } \mathfrak{R}_n \\
\longrightarrow & (\beta) & \cup\mathcal{A}_i & \vdash F & \text{Tau}(\alpha_1,\ldots,\alpha_n)
\end{array}$$

provided that F is a consequence of F_1 through F_n in propositional logic.

For practical purposes rule I-Tau must be further divided into smaller rules. Three types are possible, namely *analytic rules,* breaking up formulae, *synthetic rules,* constructing formulae from others, and finally *converting rules,* which transform a formula into an equivalent form. A complete list of these rules can be found in appendix B. Here is just one example of each of the different types.

<u>I∧left:</u>		(α)	$\mathcal{A}$	$\vdash$	$F \wedge G$	Rule $\mathfrak{R}$
	$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	F	$\wedge E(\alpha)$
<u>Is∧:</u>		(α)	$\mathcal{A}_1$	$\vdash$	F	Rule $\mathfrak{R}_1$
		(β)	$\mathcal{A}_2$	$\vdash$	G	Rule $\mathfrak{R}_2$
	$\rightarrow$	(γ)	$\mathcal{A}_1,\mathcal{A}_2$	$\vdash$	$F \wedge G$	$\wedge I(\alpha, \beta)$
<u>Ic∨∧:</u>		(α)	$\mathcal{A}$	$\vdash$	$E \vee (F \wedge G)$	Rule $\mathfrak{R}$
	$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	$(E \vee F) \wedge (E \vee G)$	Tau(α)

3.4 A Semiautomatic Proof System

The set of transformation rules defined in the previous section constitutes a proof system for natural deduction proofs. This means that for any valid formula F, there is a finite sequence of GNDPs starting with $\vdash F$ and ending with an NDP for F. Every element in this sequence follows from its predecessor by application of one of the transition rules (Completeness of the set of transition rules).

This system could be used as a proof checker, the user choosing from a menu of applicable rules, and the system correctly applying them. However the system can actually do much more by preselecting a subset of the transformation rules and giving the user a much smaller choice of rules.

In the previous section, we mentioned the problem that almost all composed formulae had to be broken up until the literal level is reached. This happens if external rules are always applied first as suggested by the completeness proof in 3.2-5. To inhibit this behaviour we introduce the notion of "integral formulae" which should not normally be separated into subformulae. Then, starting from the trivial GNDP, external rules are applied – if necessary with the help of the user – until all of the conclusion formulae of external lines are either literals or suitable subformulae of internal conclusions. In this way we avoid having to break down the formula to prove into its literals and to build it up again later. If a complex formula is contained as a whole in some axiom, then there is hope that a shorter, more comprehensive proof is available.

3.4-1 Definition: (integral formulae)

In the context of (generalized) natural deduction proofs, *integral formulae of an external line* are defined as follows. We distinguish between *strongly integral* and *weakly integral formulae:*

(a) Any conclusion formula F of an internal line is a strongly integral formula for an external line ε, if it depends only on assumptions which ε also depends on.

(b) Any strongly integral formula is also weakly integral for the same external line.

(c) If F is a strongly (weakly) integral formula, then with

$F = A \land B$,	A and B are strongly (weakly) integral,
$F = A \lor B$,	A and B are weakly integral,
$F = A \Rightarrow B$,	B is weakly integral, strongly , if both F and A are strongly integral,
$F = \neg(\neg A)$,	A is strongly (weakly) integral,
$F = \forall x\ Ax$,	At becomes strongly (weakly) integral, if it is known, that the term t must be inserted for x during the proof,
$F = \exists x Fx$	Fc becomes strongly (weakly) integral, if it is known, that c is inserted for x during the proof.

If $F = \neg(A\land B)$, $\neg(A\lor B)$, $\neg(A\Rightarrow B)$, or $\neg\exists xAx$, then $\neg A\lor\neg B$, $\neg A\land\neg B$, $A\land\neg B$, or $\forall x\neg Ax$ are strongly (weakly) integral, respectively.

In the following, for each of the external lines in a GNDP there are sets of strongly and weakly integral formulae. External rules must not be applied if the conclusion of the external line is among its strongly integral formulae, for in this case it is possible to derive the goal without further breaking up the formula. If the formula to prove is only weakly integral, then there is no guarantee that a direct proof exists, but in this case there is a good chance that the proof can be conducted by case analysis if the goal formula is a subformula of a disjunction, or by an application of M-Inf if it appears on the right hand side of an implication. Then mixed rules may be used, and only when no more external or mixed rules can be applied does the system start using internal rules.

Now the strategy for a semiautomatic proof system can be described by the following algorithm:

3.4-2 Algorithm: (basic proof transformation)

1. Start with GNDP $= \emptyset \vdash F$. Initialize strongly and weakly integral formulae for this external line as empty sets.
2. Apply automatic external rules as long as possible, until all of the external lines conclude in weakly integral formulae. Whenever new assumptions are added, they become strongly integral formulae for the external line in case. As always, the complete sets of integral formulae are computed.

   If the external line is now integral, go to 4.
3. If possible, ask the user whether any of the user-guided external rules should be applied. If so, do it and update the sets of integral formulae, then go to 2.
4. Now apply mixed rules at the user's discretion until no longer possible. After every application the sets of integral formulae have to be updated. If M-Inf was applied, go to 2.
5. Check whether the proof is already completed, in which case the GNDP is returned as final proof.
6. Let the user choose which of the internal rules shall be applied. Then go to 4.

3.5 The Automatic Transformation Procedure

Of course, there is no reason why "the user" could not be the computer itself, which selects the transformation rule according to appropriate heuristics, making use of the information in a given proof, for instance a previously computed refutation graph.

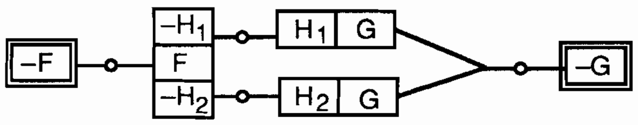
The selection between different rules that might be applicable is guided by the refutation graph representing the proof of the external lines in a GNDP. The assumptions of such external lines may then be treated as axioms for this particular proof. In a refutation graph there is a priori no distinction between clause nodes representing axioms and others representing (negated) theorem parts. In order to formalize such a distinction we use the notion of polarization of clause nodes defined in 3.1-6, so that clause nodes are positively polarized, if they stem from an axiom (or an assumption), and negatively polarized, if they represent a part of the (negated) theorem.

In this section we show how a proof represented as a refutation graph can guide the "search" for a natural deduction proof. In this context, search means to transform the given, graph-represented proof into the natural deduction calculus, rather than to find an original proof.

To this end, we use the above rule system and gradually change generalized NDPs in order to complete the natural deduction proof. To achieve this, all the tasks formerly done by the user have to be taken over. This includes the choice between several applicable transformation rules and the update of the deduction graph during the whole process. The information about the automatically generated proof consists of a refutation graph, a ground substitution for all the formulae needed, which contains the information about skolemization and duplication of clauses, and a relation Δ between the literal nodes of the refutation graph and the atom occurrences of the input formulae, indicating where the literal nodes stem from.

In this section, external rules will only be considered applicable if their conclusions are not integral. The first point of choice in algorithm 3.4-2 comes up, when rule E$\vee$ is applicable. In this case one of its four versions can always be used, so in order to decide which one, we use information from the graph. Whenever E$\vee$-left or E$\vee$-right can be applied, that is when one part of the disjunction is directly provable, then the other part does not appear in the refutation graph, which is a property easy to check. The choice between E$\vee$1 and E$\vee$2 is only a stylistic one; in this case there are always two clause nodes in the graph representing the two parts of the disjunction, so the problem appears to be symmetric. In fact, both E$\vee$1 and E$\vee$2 always work. But the structure of the refutation graph may help to make a good choice as can be seen in the following example where F$\vee$G must be proved:

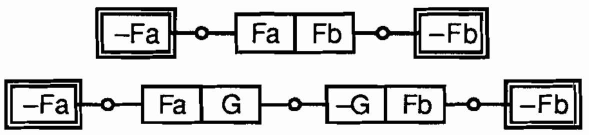
3.5-1 Example Graph: (disjunctive theorem)



The above refutation graph represents a proof of $F \vee G$ from axiom formulae $H_1 \wedge H_2 \Rightarrow F$, $H_1 \vee G$, and $H_2 \vee G$. E$\vee$1 and E$\vee$2 allow to choose either $\neg G$ or $\neg F$ as an additional assumption. A good heuristic is to choose the one with a larger number of literal nodes in its shore, $\neg G$ in this example, as an additional assumption. In this way the rest of the proof can be divided into two independent parts, while choosing $\neg F$ would lead to a subsequent application of the Rule of Cases. This heuristic can also be generalized to cases where the formulae F and G are not literals. Another heuristic is to use the formula as an assumption that appears in the graph in several copies.

When an existentially quantified formula is to be proven, one has to consult the total unifier of the refutation graph. If the variable has been instantiated only once, that is, no copy of the formula has been needed, then E-∃constructive can be set to work. If copies have been made, one way to continue is always to construct a proof by contradiction starting with an application of E-∃. But there are certain cases, when this is not necessary.

3.5-2 Example Graphs: (existentially quantified theorem)



In the upper graph the proof can be done in two cases with Fa and Fb as assumptions, while in the lower one M-Divide can be applied for G and ¬G, and so the refutation graph is cut into two parts. Another possibility in the second case is to derive Fa ∨ Fb first, and continue as in case one.

More complex situations arise, when a decision about mixed rules has to be made. After all, it is certainly not useful to divide the proof into cases every time a disjunction has been derived. And we have seen earlier, that some of these rules may lead into dead ends when applied incautiously. This is not desired, however, and large classes of graphs have to be found where an application of such rules is safe and leads to nice proofs.

In the following section we use the rule M-Inf as an example to show how certain structural properties of the refutation graph can be sufficient to guarantee the safety of the application of such rules. The rule M-Inf is repeated below as a reminder:

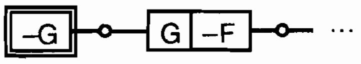
<u>M-Inf:</u>

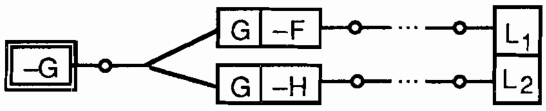
$$\left.\begin{array}{llll} (\alpha) & \mathcal{A} & \vdash F\Rightarrow G & \text{Rule } \mathfrak{R} \\ (\gamma) & \mathcal{A} & \vdash G & \pi \end{array}\right\} \longrightarrow \left\{\begin{array}{llll} (\alpha) & \mathcal{A} & \vdash F\Rightarrow G & \text{Rule } \mathfrak{R} \\ (\beta) & \mathcal{A} & \vdash F & \pi' \\ (\gamma) & \mathcal{A} & \vdash G & \Rightarrow E(\alpha,\beta) \end{array}\right.$$

Let as assume that we have to decide about the application of M-Inf. The danger is to apply it, when a proof of F is not known or when F is not provable at all. So we have to make sure that F is valid and its proof can be constructed from the graph.

This can be seen from the graph, provided the automatically found proof has used the formula $F \Rightarrow G$. The simplest case is when the formulae F and G involved are both literals.

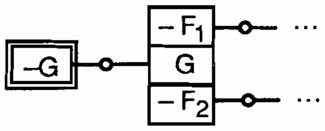
3.5-3 Example Graphs: (graph condition for M-Inf)

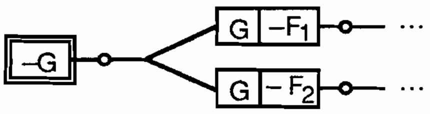




Whenever the graph has the first structure, the rule may be applied. Should the clause [–F G] be missing in the graph, the rule can obviously not be applied, but there are also situations, where [–F G] is part of the graph, but F is still not derivable from axioms, as in the second example graph. In this case one could, for instance, break up the proof into cases with assumptions L_1 and L_2, but only in the first case does the proof use the implication $F \Rightarrow G$.

3.5-4 Example Graphs: (graph condition for M-Inf)

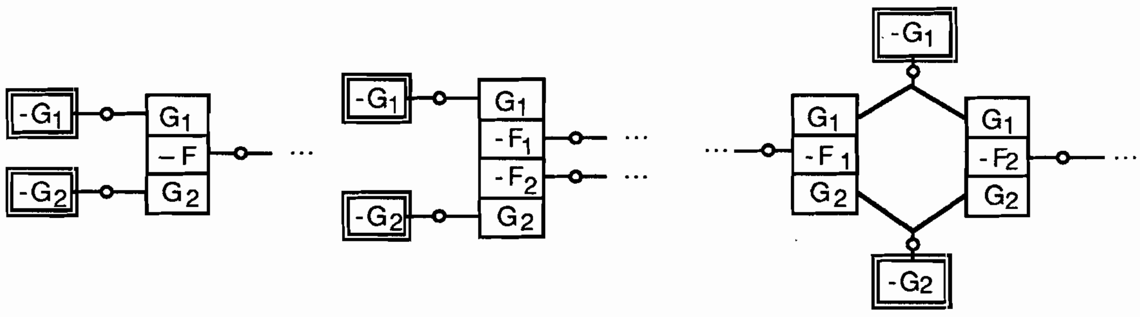




The two refutation graphs above cover the cases, where F is a conjunction or disjunction. In the first case ($F=F_1 \wedge F_2$) there must not be a trail between $-F_1$ and $-F_2$, since then the graph would be cyclic, and therefore not a refutation graph. If in the second case ($F=F_1 \vee F_2$) there were no such trail, then the refutation graph would not be minimal, since any one of the branches could be omitted. This leads to a situation where either F_1 or F_2 can be derived independently as in E∨left or E∨right. If, on the other hand, there is such a trail, then E∨1 or E∨2 can be applied after M-Inf.

When $G=G_1 \vee G_2$ is a disjunction and F either literal, conjunction, or disjunction, then the following refutation graphs allow an application of M-Inf:

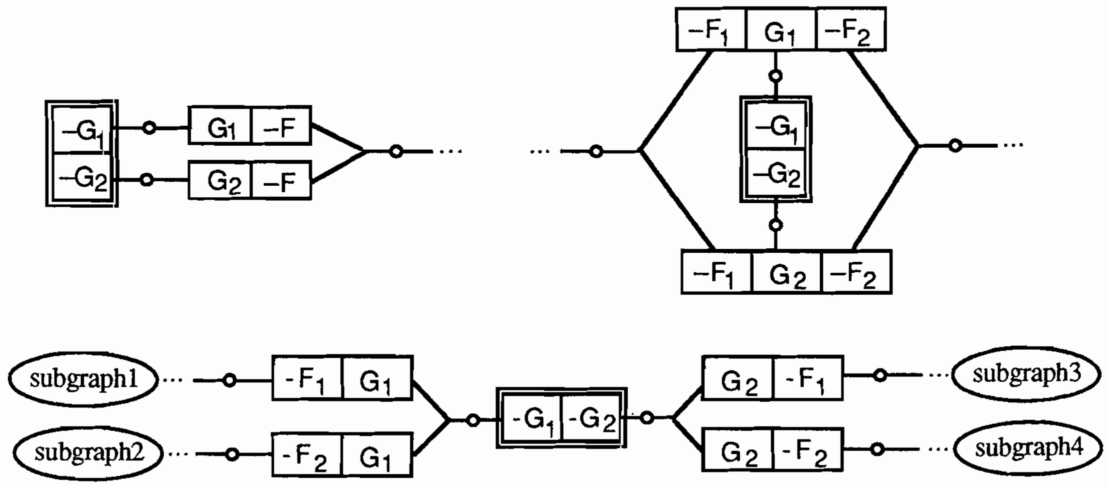
3.5-5 Example Graphs: (graph condition for M-Inf)



As in the first case of 3.5-4 there must not be a trail between $-F_1$ and $-F_2$ in the refutation graph in the second of the example graphs; and the last graph is only minimal if there is such a trail.

The set of examples is completed with the case where G is a conjunction. Again F may be either literal, conjunction, or disjunction.

3.5-6 Example Graphs: (graph condition for M-Inf)



When F is a disjunction, there are two essentially different possible graphs. They correspond to the cases of the subsequent applications of different versions of rule E∨. The first one – subsequent application of E∨-left or E∨-right – is of course isomorphic to the case, where F is a literal. The second case is the last one illustrated above; for this graph to become a refutation graph, trails must exist both between subgraph 1 and subgraph 2 as well as between subgraph 3 and subgraph 4. Otherwise the graph would not be minimal.

The cases where F or G are implications can always be seen as one of the disjunction cases above . And also when F or G are quantified formulae, there is no essential difference. We will now prove a general condition which guarantees a successful application of rule M-Inf.

3.5-7 Lemma: (Graph Condition for M-Inf)

Given a GNDP with an internal line $\{(\lambda)\ \mathcal{A} \vdash F \Rightarrow G \text{ Rule } \mathfrak{R}\}$ and an external line $\{(\xi)\ \mathcal{A} \vdash G\ \Gamma\}$, where Γ is a minimal refutation graph representing a proof of $\bigwedge\mathcal{A} \Rightarrow G$ with a clause graph relation Δ between the atom occurrences of $\bigwedge\mathcal{A} \Rightarrow G$ and the literal nodes of Γ.

If all the literal nodes in $\Delta(<\bigwedge\mathcal{A} \Rightarrow G, G>)$ are directly adjacent exclusively to literal nodes of $\Delta(<\bigwedge\mathcal{A} \Rightarrow G, \bigwedge\mathcal{A}, \ldots, F \Rightarrow G>)$[1], then a refutation graph Γ' representing a proof of $\bigwedge\mathcal{A} \Rightarrow F$ can be constructed by

$$\Gamma' = \Gamma - \bigcup\nolimits_{N \in \Delta(\Omega_a(<\bigwedge\mathcal{A} \Rightarrow G,\, G>))} \Lambda(N).$$

Proof: In order to prove the lemma we must show that the resulting clause graph is a refutation graph, and that all of its clause nodes represent assumption formulae or have parent clauses in the clause form of the negation of F.

Let us first assume that $(F \Rightarrow G) \in \mathcal{A}$, then (a subset of) the clause form of $F \Rightarrow G$ is directly present in the refutation graph Γ. No links are added, so no cycle can be introduced. Γ' cannot be empty, because it still contains at least the literal nodes $\Delta(\Omega_a(<\bigwedge\mathcal{A} \Rightarrow G, \bigwedge\mathcal{A}, \ldots, F \Rightarrow G, F>))$, and as any links are deleted (automatically) only if all of the literal nodes of one shore have been removed, no literal node can have become pure. Hence the graph Γ' is still a refutation graph.

All the clause nodes stemming from the negation of G have been completely removed from Γ, so all the clause nodes in Γ' have parents in the clause form of $\mathcal{A}$ or have been constructed by deleting $\Delta(\Omega_a(<F \Rightarrow G, G>))$ from the subgraph $\Gamma_{\neg F \vee G}$ representing $F \Rightarrow G$. Now by lemma 2.5-6, $\Gamma_{\neg F \vee G} - \Delta(\Omega_a(<\neg F \vee G, G>))$ is a clause graph representing $\neg F$. So Γ' contains only clause nodes describing the negation of F or assumption formulae.

If $(F \Rightarrow G) \notin \mathcal{A}$, it must have been previously derived from the assumptions $\mathcal{A}$. Therefore the refutation graph Γ must contain a deduction graph proving $F \Rightarrow G$ as a

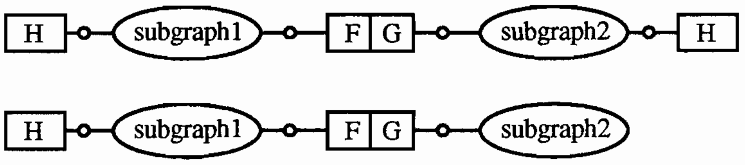
[1] If $\Delta(F \Rightarrow G)$ is only contained as a deduction graph in Γ, then $\Delta(<\bigwedge\mathcal{A} \Rightarrow G, \bigwedge\mathcal{A}, \ldots, F \Rightarrow G>))$ means the pure literal nodes of this subgraph.

subgraph. In this case F⇒G can be derived first from this deduction graph (as a subproblem)[1], the clause form of F⇒G will then actually appear in the graph and the problem is reduced to the first case above. Q.e.d. ■

For the other proof-driven transformation rules, E∃-constructive, E¬∀, E∨left, and E∨right, the conditions for applicability are easier to check. If a line {𝒜 ⊢ F∨G Γ} is in the GNDP, E∨left can be applied, if Δ(<F∨G, G>) is empty, and the case of E∨right is symmetrical. If a line {𝒜⊢∃xF Γ} is in the GNDP, then E∃-constructive can be applied, if in all the literal nodes in Δ(∃xF) the variable x has been substituted by the same ground term t. The case of E¬∀ behaves similar.

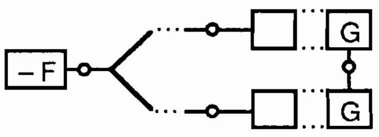
No graph properties need to be checked before an application of the rules M-Cases, M-Divide, or M-Unless, for these rules are sound and the refutation graph required will always exist. But one can give "necessary conditions" for their application to be useful. The example graphs below cover only the case, where F, G, and H are literals, but they can be generalized just as those for rule M-Inf.

M-Cases:



For rule M-Cases the idea is to "cut" through a disjunction F∨G to obtain refutation graphs for a proof assuming F and G respectively. This is only useful, however, when the resulting graphs both contain (parts of) the original theorem H, as in the upper example graph. If one of the resulting graphs does no longer contain (parts of) H then the subsequent proof can only be completed as a proof by contradiction. It is not always necessary that all of Δ(H) should appear in both resulting graphs; if for instance H = A ∨ B, then one part might prove A and the other B.

E-Divide:

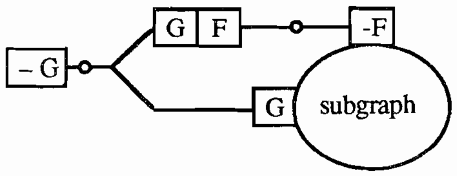


This rule M-Divide is only a variation of the rule M-Cases, which can be applied when no suitable disjunction for M-Cases is available. A removal of the link between G and ¬G must separate the graph into two components, both of which contain (parts of) F. This effect could also be reached by introducing the axiom G∨¬G and then use

[1] Cf. the chapter on proof structuring, especially the transformation rule E-Lemma.

it for case analysis. Such a rule is also suggested by Frank Pfenning and Daniel Nesmith in [PN90].

M-Unless:



M-Unless is also a special case of M-Cases, where one of the cases is trivial. Here a cut through F and G in [F G] leads to the desired linearization of the proof.

The problem of selecting the correct internal rules remains to be dealt with. When all the external formulae are integral and none of the mixed rules can be applied, then the task is either to derive the external formulae by forward reasoning, using internal rules, or to derive new internal lines in order to apply mixed rules later on.

In case of strongly integral formulae one only has to apply I–, I∀, or analytic propositional rules in order to derive the desired formula. When a weakly integral formula is to be proved, then one starts just as for strongly integral formulae, but stops when the subformula is reached, where further subformulae become only weakly integral. This must be a disjunction, an implication, or an existentially quantified formula, containing the original formula as a subformula. Now the appropriate mixed rule must be applied, and then the process has to be repeated.

3.6 Updating the Refutation Graph

In the rest of this chapter, the process of updating the refutation graph after each application of a rule is described. There are essentially five tasks to be done:

1. Update the total substitution.
2. Ensure that the conclusion formula of every proof line is anchored in the formula F originally to be proved, so that the corresponding literal nodes can be obtained using Δ.
3. Update the sets of integral formulae for the external line currently worked with.
4. Add parts of the refutation graph to the set of positively polarized clause nodes, when new assumptions are introduced.

5. Construct refutation graphs for newly created external lines, this means to remove parts from the graph or to divide it.

For these tasks we will give examples and prove that the graphs resulting from task five above have all the necessary properties to prove the new external lines.

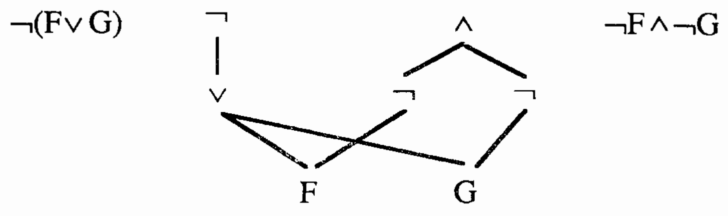
The substitution has to be altered whenever one of its components has been used. This is the case for rule I$\forall$, when the component $x \mapsto t$ can be removed from the substitution. Similarly, when E$\exists$-constructive or E$\neg\forall$ are applied, a component $x \mapsto t$ can be removed. Finally, when E$\forall$ or M-Choose are used, the component $x \mapsto c$ can be removed, where c is the Skolem constant having been introduced when the Skolem normal form of the formula to prove was constructed.

The handling of Skolem terms, i.e. terms with a Skolem function depending on one or more variables, may seem to be a problem, as both transformation rules E$\forall$ and M-Choose always introduce Skolem constants. This is not the case, however, as can be seen for the (axiom) formula $\forall x \exists y Pxy$, which is skolemized to $Pxfx$ with a Skolem function f. Now rule M-Choose can only be applied after the variable x has been instantiated by a ground term t_g and the formula became $\exists y P t_g y$. Now the Skolem constant introduced in the next step, when M-Choose is applied, corresponds to the Skolem term ft_g. So for every ground instantiation of the Skolem term fx there will be a different Skolem constant in the natural deduction proof.

It is necessary to map the conclusion formula of every proof line to a formula anchored in the formula F to prove, so that the corresponding literal nodes in the refutation graph can be localized using Δ. The anchorage is trivial if only analytic transformation rules are applied, where the new conclusion formulae are always immediate subformulae of their predecessors. For synthetic and converting rules, however, and also for some of the external rules, this is not the case. But an anchorage can always be obtained; with the help of lemma 3.1-5 it suffices to find a basis of the new formula that shares a tail with atom occurrences in F.

3.6-1 Example: (anchorage)

As an example we assume that $\neg(F \vee G)$ is transformed into $\neg F \wedge \neg G$. Now the two formula occurrences $\omega_F = \langle \neg F \wedge \neg G, \neg F, F \rangle$ and $\omega_G = \langle \neg F \wedge \neg G, \neg G, G \rangle$ form a basis of $\neg F \wedge \neg G$. They obviously share a tail with $\langle \neg(F \vee G), (F \vee G), F \rangle$ and $\langle \neg(F \vee G), (F \vee G), G \rangle$ which were anchored before in the formula to prove, so that an anchorage can be constructed. The two formula trees are shown below



♦

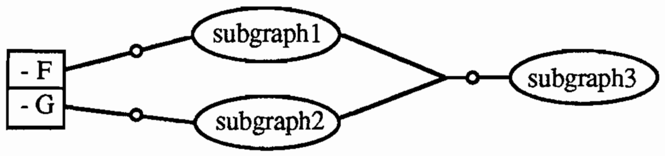
New sets of integral formulae must be computed, whenever new assumptions are introduced or new internal lines are derived with conclusion formulae, which were not integral before. This is the case for the external rules E⇒, E¬, E∨1, E∨2, and M-Divide, for the mixed rules M-Cases, M-Unless, and M-Choose, and for rule I∀. The rule E⇒ is an example, where a new assumption is introduced. In this case this formula and its integral closure is added to the set of integral formulae.

For the other external rules and M-Inf, the set of integral formulae of the parent line is passed on to its successors. For rule E∧ for example, the lines $\{\mathcal{A} \vdash F\ \pi_1\}$ and $\{\mathcal{A} \vdash F\ \pi_2\}$ inherit the set of integral formulae from $\{\mathcal{A} \vdash F \wedge G\ \pi\}$. None of the internal rules computing derivations in propositional logic change the sets of integral formulae, as these were integral before.

The set of positively polarized literal nodes in the graph must be augmented, if the rule applied makes an additional assumption, which has previously been a subformula of the theorem to prove. The most common example is rule E⇒; originally all the literal nodes in Δ(<F⇒G>) were negatively polarized, but when F is assumed as an axiom in the proof of G, all of Δ(<F⇒G, F>) becomes positively polarized. The same has to be done analogously for rules E¬, E∨1, and E∨2. One has to understand, that positively polarized parts represent "axioms". This neatly reflects the general idea of natural deduction proofs, where new assumptions are introduced during the proof process.

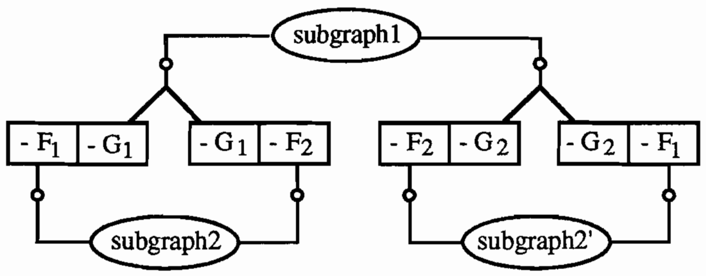
The most difficult part of the updating process is the division of the graph, when the proof can be split into two independent parts. This is the case after an application of E∧, M-Divide, or M-Cases. With literals F and G, a graph might have the following form before application of E∧:

3.6-2 Example Graph:



The refutation graph must now be split into two parts. This is done by dividing the clause nodes [–F –G] into two clause nodes [–F] and [–G]. The remaining graph is obviously still a refutation graph, since no literal node becomes pure and no cycle can be introduced, but the graph is no longer minimal. It can be seen, that its two components are independent refutation graphs for F and G. If these components have a non-empty intersection (subgraph 3), then this subgraph must be duplicated. This is not desirable, however, when it is too large. In this case one should work with the subgraph first, in order to generate a lemma, which can later be used in both the proofs of F and G. Lemma generation is dealt with in the next chapter.

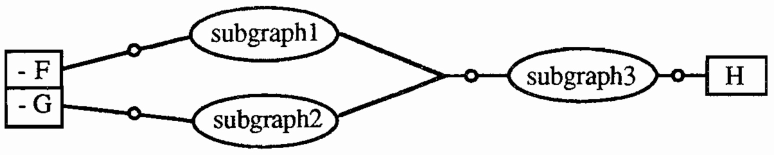
3.6-3 Example Graph:

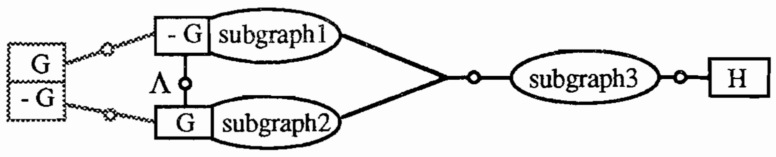


The structure of the graph becomes more complex, when F and G are disjunctions. The cut must now be fourfold, separating –F from –G in four different clause nodes. Now any of the (four) isomorphic components containing parts of G together with subgraph1 represents a proof of G. To prove F one has a choice of adopting the component containing subgraph2 or the one containing subgraph2'. To add to the complexity, any two subgraphs may intersect.

The structure of the refutation graphs resulting from rule M-Cases or M-Divide is "dual" to those from E$\wedge$. This can be seen in the graphs below, where F, G and H are assumed to be literals.

3.6-4 Example Graphs: (M-Cases, M-Divide)





This time one must cut through the positively polarized clause node [F G], and the resulting two graph components are both proofs for H, assuming F or G as an additional assumption. In case of M-Divide one can insert a clause node [G ¬G] for the link Λ between ¬G and G and properly connect it to the rest of the graph; then it is possible to continue by cutting through [G ¬G] and continue with cases G and ¬G.

We will now rigorously prove that the graphs resulting from the splitting procedure really are refutation graphs for the formulae in question. To this end we prove the following lemma:

3.6-5 Lemma: (Splitting the graph)

Let Γ be a minimal deduction graph representing a formula $H = (F \vee G) \wedge H_{rest}$ with clause graph relation Δ. Then after splitting all the clause nodes corresponding to $F \vee G$, the resulting graph has a component Γ_F representing $F \wedge H_{rest}$ and a component Γ_G representing $G \wedge H_{rest}$.

Proof: Because of the symmetry it is sufficient to prove that one of the resulting components represents $G \wedge H_{rest}$. The proof can be done by induction on the number n of clause nodes related to $F \vee G$.

If $n=1$, the graph is split in exactly two components, one of which contains no F-parts according to lemma 2.5-3. Now lemma 2.5-6 guarantees that this component represents $G \wedge H_{rest}$.

If $F \vee G$ is represented by $n+1$ clause nodes $C_1, \dots, C_{n+1}$ in Γ, we can split the graph by splitting $C_1, \dots, C_n$ first. Then by induction hypothesis, there is a component Γ_G containing no F-literal-nodes of the clause nodes $C_1, \dots, C_n$; Γ_G could still contain C_{n+1}, of course. If this is the case we split C_{n+1} in Γ_G into two components, one of which contains no F-literal-nodes as in the induction base, which completes the proof. ■

3.6-6 Corollary: (Splitting theorem)

Let Γ be a minimal refutation graph proving $F \wedge G$ from assumptions $\mathcal{A}$. Then one of the components resulting from splitting the clause nodes corresponding to $F \wedge G$ is a minimal refutation graph for $\mathcal{A} \Rightarrow F$ and one is a minimal refutation graph for $\mathcal{A} \Rightarrow G$.

Proof: Γ proves $\mathcal{A} \Rightarrow F \wedge G$, therefore Γ represents $\mathcal{A} \wedge (\neg F \vee \neg G)$. The previous lemma tells us that, after splitting all the clause nodes corresponding to $\neg F \vee \neg G$, there is a component containing no G-parts and a component without any F-parts. Both these components are minimal (by definition) and refutation graphs (by Lemma 2.5-1), therefore they prove $\mathcal{A} \Rightarrow F$ and $\mathcal{A} \Rightarrow G$, respectively. ■

As suggested by example 3.6-4, splitting is not confined to breaking up the formula to be proved into several parts. The mechanism is exactly the same, when the proof is done by case analysis. In this case the splitting has to be done using a disjunctive assumption (or derived) formula.

3.6-7 Definition: (Distributed Formulae)

Let Γ be a clause graph representing a formula $H = H_1 \wedge (F \vee G) \wedge H_{rest}$. Then lemma 3.6-5 guarantees the existence of clause graphs Γ_F and Γ_G if Γ is split between F and G, such that Γ_F contains no G-parts and Γ_G no F-parts. The formula H_1 is said to be *distributed* in Γ with respect to the disjunction $F \vee G$ if both Γ_F and Γ_G contain literal nodes corresponding to H_1.

3.6-8 Lemma: (Graph Condition for M-Cases)

Given a GNDP with an internal line $\{(\lambda)\ \mathcal{A} \vdash F \vee G\ \text{Rule}\mathfrak{R}\}$ and an external line $\{(\xi)\ \mathcal{A} \vdash H\ \Gamma\}$, where Γ is a minimal refutation graph proving $\mathcal{A} \Rightarrow H$ with a clause graph relation Δ.

Then, if H is distributed in Γ with respect to $F \vee G$, splitting the graph by splitting all the clause nodes in $\Delta(\Omega_a(F \vee G))$ between their F- and G-parts, leads to a component proving $\mathcal{A}, F \Rightarrow H$ and a component $\mathcal{A}, G \Rightarrow H$, so that M-Cases can be applied.

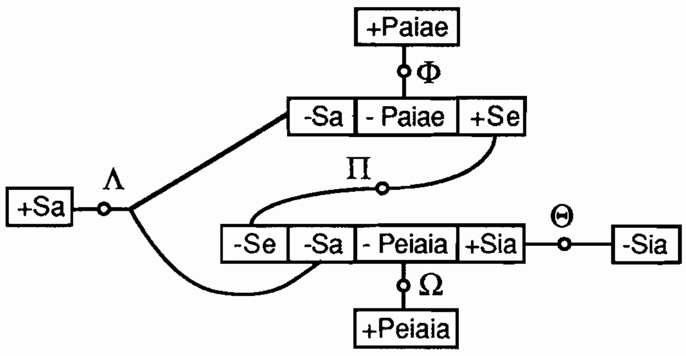
Proof: As in the proof of lemma 3.5-7 we assume that $(F \vee G) \in \mathcal{A}$, otherwise $F \vee G$ can be derived as a subgoal first. Hence Γ represents the formula $\mathcal{A}_{rest} \wedge (F \vee G) \wedge \neg H$. From lemma 3.6-5 we know that splitting leads to components representing $\mathcal{A}_{rest} \wedge F \wedge \neg H$ and $\mathcal{A}_{rest} \wedge G \wedge \neg H$, which are refutation graphs proving $\mathcal{A}, F \Rightarrow H$ and $\mathcal{A}, G \Rightarrow H$, respectively. ■

3.7 Example

As an example of the automatic proof transformation procedure, we use again a part of the subgroup criterion (cf. example 2.5-5). The formula to prove is $F = \forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz) \Rightarrow (\forall v\, Sv \Rightarrow Siv)$. In order to develop a natural deduction proof of this formula we start with the following information:

- a refutation graph Γ proving F (repeated below)
- a clause graph relation $\Delta \subseteq \Omega_a(F) \times \mathbb{N}$
- a ground substitution γ mapping the variables in $C(\neg F)$ to the ground terms occurring in corresponding clause nodes of the refutation graph. The variables of the formula $\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz$, which is instantiated twice, are renamed to x_1, y_1, z_1, and x_2, y_2, z_2.

All of this information, the refutation graph, the clause graph relation, and the ground substitution has been automatically generated by the theorem prover MKRP, see [MKRP84] and [Le88]. Below the refutation graph Γ is repeated:



$\gamma = \{u \mapsto a,\ w \mapsto ia,\ v \mapsto a,\ x_1 \mapsto a,\ y_1 \mapsto a,\ z_1 \mapsto e,\ x_2 \mapsto e,\ y_2 \mapsto a,\ z_2 \mapsto ia\}$

Δ is the relation explained in example 2.5-5.

The transformation process is now started with the trivial GNDP for F

(16) $\vdash$ $(\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz))$
$\Rightarrow (\forall v\, Sv \Rightarrow Siv)$ Γ

The first rule to be applied is E⇒, which leads to the next GNDP:

(1)	1	⊢	$\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)$	Ass
(15)	1	⊢	$\forall v\, Sv \Rightarrow Siv$	Γ
(16)		⊢	$(\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)) \Rightarrow (\forall v\, Sv \Rightarrow Siv)$	⇒I(15)

The new external line is justified by the same refutation graph Γ, Δ and γ have also not changed, but now we can compute the set I of integral formulae for the external line (15) according to definition 3.4-1. With the ground terms for the instantiations from γ, $I = \{\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)$, $\forall u\, Puiue$, $Paiae$, $\forall w\, Peww$, $Peiaia$, $\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz$, $Sa \wedge Paiae \Rightarrow Se$, $Se \wedge Sa \wedge Peiaia \Rightarrow Sia\}$.

The conclusion of line 15 is not in I, so the next external rule is applied (E∀), the variable v being replaced by the constant a.

(1)	1	⊢	$\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)$	Ass
Let a be an arbitrary constant				
(14)	1	⊢	$Sa \Rightarrow Sia$	Γ
(15)	1	⊢	$\forall v\, Sv \Rightarrow Siv$	∀I(14)
(16)		⊢	$(\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)) \Rightarrow (\forall v\, Sv \Rightarrow Siv)$	⇒I(15)

Again Γ and Δ remain unchanged, but now the component $v \mapsto a$ is removed from γ. The next step is another application of E⇒, and the following GNDP is constructed:

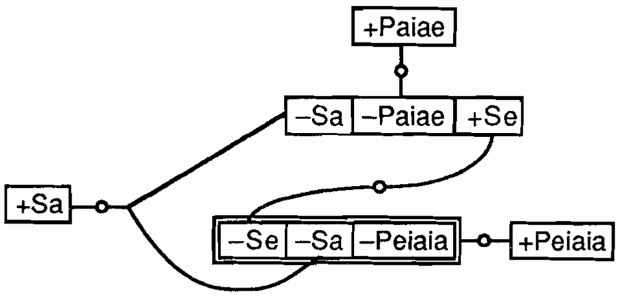
(1)	1	⊢	$\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)$	Ass
Let a be an arbitrary constant				
(2)	2	⊢	Sa	Ass
(13)	1, 2	⊢	Sia	Γ
(14)	1	⊢	$Sa \Rightarrow Sia$	⇒I(13)
(15)	1	⊢	$\forall v\, Sv \Rightarrow Siv$	∀I(14)
(16)		⊢	$(\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)) \Rightarrow (\forall v\, Sv \Rightarrow Siv)$	⇒I(15)

With the introduction of a new assumption the set of integral formulae is augmented by Sa; but with $Sa \wedge Paiae \Rightarrow Se$ and $Se \wedge Sa \wedge Peiaia \Rightarrow Sia$ being integral, the formulae Se and Sia become integral as well.

Up to here only automatic transformation rules were applied. As this is no longer possible at this stage and the current goal formula is integral, we try to apply one of the mixed rules. With the implication $Se \wedge Sa \wedge Peiaia \Rightarrow Sia$ being an instance of an assumption formula, M-Inf can be applied after checking the graph condition (3.5-7). Here the only link from [Sia] connects to the clause node representing the implication, so M-Inf may be applied and the GNDP changes to

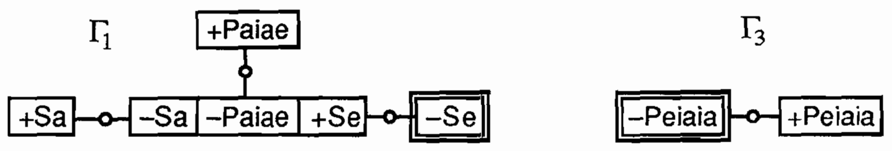
(1)	1	$\vdash$	$\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)$	Ass
Let a be an arbitrary constant				
(2)	2	$\vdash$	Sa	Ass
(3)	1	$\vdash$	$\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz$	$\wedge$E(1)
(4)	1	$\vdash$	$Se \wedge Sa \wedge Peiaia \Rightarrow Sia$	$\forall$E(3)
(12)	1	$\vdash$	$Se \wedge Sa \wedge Peiaia$	Γ'
(13)	1, 2	$\vdash$	Sia	$\Rightarrow$E(4, 12)
(14)	1	$\vdash$	$Sa \Rightarrow Sia$	$\Rightarrow$I(13)
(15)	1	$\vdash$	$\forall v\ Sv \Rightarrow Siv$	$\forall$I(14)
(16)		$\vdash$	$(\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)) \Rightarrow (\forall v\ Sv \Rightarrow Siv)$	$\Rightarrow$I(15)

Actually, it took three rules to derive this GNDP, I$\wedge$right and I$\forall$ were applied in order to isolate the implication needed for M-Inf. This could be done by using the fact that (and the reason why) Sia was integral. After this the graph has changed for the first time, it is no longer a graph proving Sia, but a graph proving $Se \wedge Sa \wedge Peiaia$:



Δ is now the restriction of the previous relation to the remaining literal nodes, and γ has shrunk to $\{u \mapsto a,\ w \mapsto ia,\ x_1 \mapsto a,\ y_1 \mapsto a,\ z_1 \mapsto e\}$.

Continuing to construct the sequence of GNDPs, rule E$\wedge$ can be applied, after which the GNDP has three external lines and the refutation part has to be broken up in order to construct three graphs proving these three lines. One of these graphs is not needed, however, as the formula Sa already appears as the conclusion of line 2.



(1)	1	$\vdash$	$\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)$	Ass

Let a be an arbitrary constant

(2)	2	$\vdash$	Sa	Ass
(3)	1	$\vdash$	$\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz$	$\wedge E(1)$
(4)	1	$\vdash$	$Se \wedge Sa \wedge Peiaia \Rightarrow Sia$	$\forall E(3)$
(9)	1, 2	$\vdash$	Se	Γ_1
(11)	1, 2	$\vdash$	$Peiaia$	Γ_3
(12)	1, 2	$\vdash$	$Se \wedge Sa \wedge Peiaia$	$\wedge I(9, 2, 11)$
(13)	1, 2	$\vdash$	Sia	$\Rightarrow E(4, 12)$
(14)	1	$\vdash$	$Sa \Rightarrow Sia$	$\Rightarrow I(13)$
(15)	1	$\vdash$	$\forall v\ Sv \Rightarrow Siv$	$\forall I(14)$
(16)		$\vdash$	$(\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz))$	
			$\Rightarrow (\forall v\ Sv \Rightarrow Siv)$	$\Rightarrow I(15)$

This procedure finally ends with a natural deduction proof of F:

(1)	1	$\vdash$	$\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)$	Ass

Let a be an arbitrary constant

(2)	2	$\vdash$	Sa	Ass
(3)	1	$\vdash$	$\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz$	$\wedge E(1)$
(4)	1	$\vdash$	$Se \wedge Sa \wedge Peiaia \Rightarrow Sia$	$\forall E(3)$
(5)	1	$\vdash$	$\forall u\, Puiue$	$\wedge E(1)$
(6)	1	$\vdash$	$Paiae$	$\forall E(5)$
(7)	1, 2	$\vdash$	$Sa \wedge Paiae$	$\wedge I(2, 6)$
(8)	1	$\vdash$	$Sa \wedge Paiae \Rightarrow Se$	$\forall E(3)$
(9)	1, 2	$\vdash$	Se	$\Rightarrow E(7, 8)$
(10)	1	$\vdash$	$\forall w\, Peww$	$\wedge E(1)$
(11)	1	$\vdash$	$Peiaia$	$\forall E(10)$
(12)	1, 2	$\vdash$	$Se \wedge Sa \wedge Peiaia$	$\wedge I(9, 2, 11)$
(13)	1, 2	$\vdash$	Sia	$\Rightarrow E(4, 12)$
(14)	1	$\vdash$	$Sa \Rightarrow Sia$	$\Rightarrow I(13)$
(15)	1	$\vdash$	$\forall v\ Sv \Rightarrow Siv$	$\forall I(14)$
(16)		$\vdash$	$(\forall u\, Puiue \wedge \forall w\, Peww \wedge (\forall xyz\, Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz))$	
			$\Rightarrow (\forall v\ Sv \Rightarrow Siv)$	$\Rightarrow I(15)$

4 Discovering the Internal Structure of a Proof

4.1 Ordering Natural Deduction Proofs

After the transformation process from a refutation graph into a natural deduction formalism the proof may now be represented as a directed acyclic graph (dag) in the following way: The nodes are labeled with proof lines of the NDP, incoming edges represent the proof lines from which the proof line of a node was derived, and outgoing edges represent steps, for which this line was needed itself. Thus axioms (or lines justified by the Assumption Rule) have no incoming edges, and the only node with no outgoing edge represents the theorem. This idea, first used by Chester in [Ch75] is defined more precisely below.

4.1-1 Definition: (NDPs represented as dags)

A directed acyclic graph (dag) with a set of nodes N, a set of edges $E \subseteq N \times N$ and a reachability relation $E^* \subseteq N \times N$, defined as the transitive closure of E, is called a *Natural Deduction Graph (NDG)*, if there is

(α) a bijection between the set of nodes N and the set of proof lines of an NDP. Therefore the nodes may be labelled with proof lines, and one may speak of the node instead of the proof line.

(β) $(n_1, n_2) \in E^*$ whenever the proof line of n_1 appears in the justification of that in n_2. If $(n_1, n_2) \in E$, then the edge is labelled with the respective rule.

(γ) $(n_1, n_2) \in E^*$ when

(γ_1) both n_1 and n_2 introduce new assumptions (by rule Ass), which are removed (by Rules $\Rightarrow$I, $\neg$I, $\vee$E, or $\exists$E) in nodes n_4 and n_3, respectively.

(γ_2) n_3 is used in the derivation of n_4, i.e. there is a chain of edges from n_3 to n_4.

(δ) $(n_1, n_2) \in E^*$, in the case of rule $\exists$E, when

$n_1 =$	$\{ \mathcal{A}$	$\vdash \exists x\, Fx$	Rule $\mathfrak{R}\}$
$n_2 =$	$\{ \mathcal{A}, Fc$	$\vdash Fc$	Ass$\}$
$n_3 =$	$\{ \mathcal{A}, Fc$	$\vdash G$	Rule $\mathfrak{R}'\}$
$n_4 =$	$\{ \mathcal{A}$	$\vdash G$	$\exists E(n_1, n_3)\}$

The reachability relation E^* defines a partial order on the proof lines, which must always be obeyed when the proof is written down in a linear form. In the next step the proof lines will be totally ordered, so that the proof can be stated in sequential form. This total order may be different from the rather accidental order of the original NDP.

Both conditions (γ) and (δ) ensure that new assumptions are not made before they are actually needed, or that subproofs are completely nested in the superior proof. This is sometimes enforced by natural deduction calculi such as Jaskowski's box formalism [Ja33].

4.1-2 Definition: (Generalized NDGs)

A dag with nodes N' and edges $E' \subseteq N' \times N'$ is called a *Generalized NDG (GNDG)*, if it is an NDG or if it can be derived from an NDG with nodes N and edges E as follows:

(α) N' must be a partition of N.

(β) There must be an ordering E_i on the internal nodes $N_i \in N'$ compatible with E^*, i.e. $(n_1, n_2) \in E^*$ implies $(n_1, n_2) \in E_i$ for all $n_1, n_2 \in N_i$.

(γ) $E' = \{(n'_1, n'_2) \mid \exists n_1 \in n'_1, n_2 \in n'_2 \text{ with } (n_1, n_2) \in E\}$.

The *size* of a node n' is the number of original proof lines appearing in it, its *rank* is the number of its immediate predecessors $\{n \mid (n,n') \in E\}$. ♦

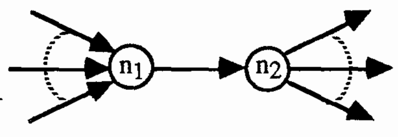
In order to linearize natural deduction proofs, Chester [Ch75] starts with an NDG and constructs a sequence of GNDGs until the graph is (almost) linear. A new element in this sequence is obtained by application of one of the following rules. If both rules can be applied, rule 1 is always preferred.

Rule 1: Combine nodes n_1 and n_2, if $(n_1, n_2) \in E$, and if for all $n \in N$ $(n_1, n) \in E$ implies $n = n_2$, and $(n, n_2) \in E$ implies $n = n_1$. The nodes from n_1 will be smaller in the internal order than those from n_2.

Rule 2: Combine nodes n_1 and n_2, if $(n_1, n_2) \in E$, n_1 contains no line justified by $\Rightarrow$I, $\text{size}(n_1) \leq \text{maxsize}$, and for all $n \in N$ $(n_1, n) \in E$ implies $n = n_2$. In doing so, use nodes with least rank first. Again the nodes from n_1 will be smaller in the internal order than those from n_2.

The following two drawings illustrate the situations of the two rules above. In rule 1 the first node is the only direct reason for the second node, which itself is the only immediate successor of the first node. This is the case, when the first line of n_2 follows directly from the last line of n_1. The second rule applies, when all the reasons for n_2 are only used in this instance, no assumption is removed (by Rules ⇒I, ¬I, ∨E, or ∃E), and n_1 is not too large. Otherwise the last line of n_1 is considered a lemma.

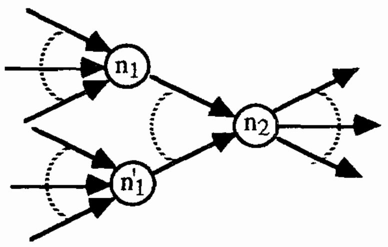
Rule 1:



Rule 2:

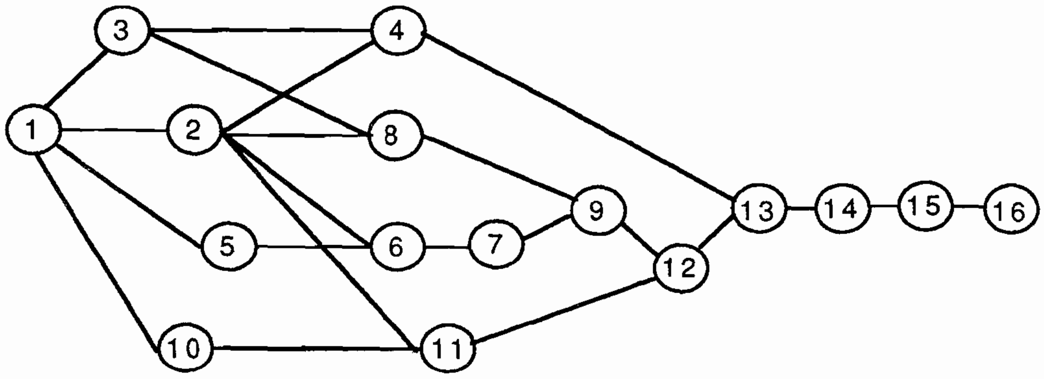
$\text{rank}(n_1) \le \text{rank}(n'_1)$

$\text{size}(n_1) \le \text{maxsize}$

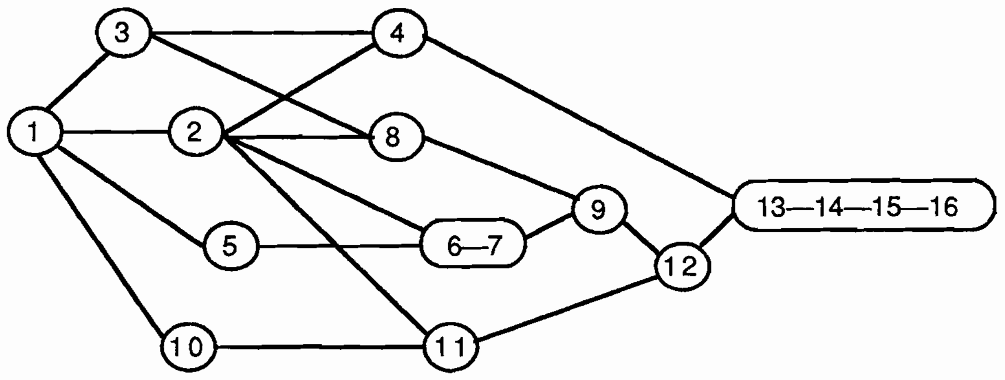


4.1-3 Example: (Natural Deduction Graph)

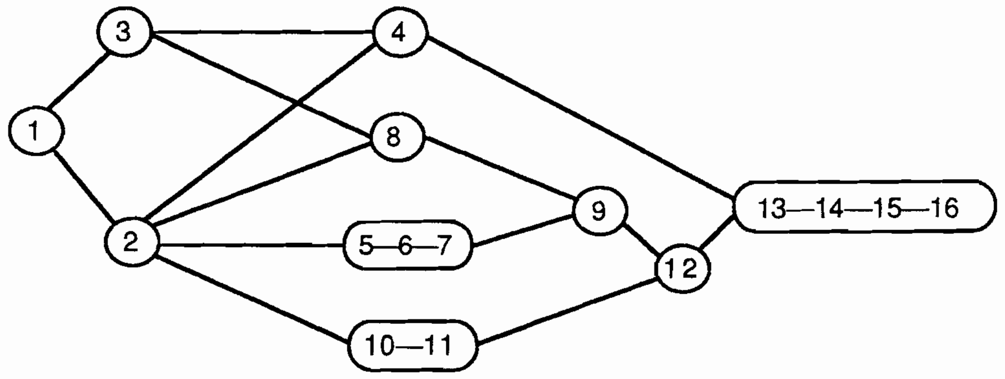
As an example we start with the natural deduction proof automatically constructed in example 3.7 by the basic proof transformation procedure (see algorithm 3.4-2). Below, the natural deduction graph is shown representing the dependency relation between the lines of the natural deduction proof. In line 2 a new (arbitrary) constant 'a' is introduced, so all the lines using this constant actually depend on line 2. The link between (1) and (2) has been introduced in order to fulfill property (γ) of definition 4.1-1.



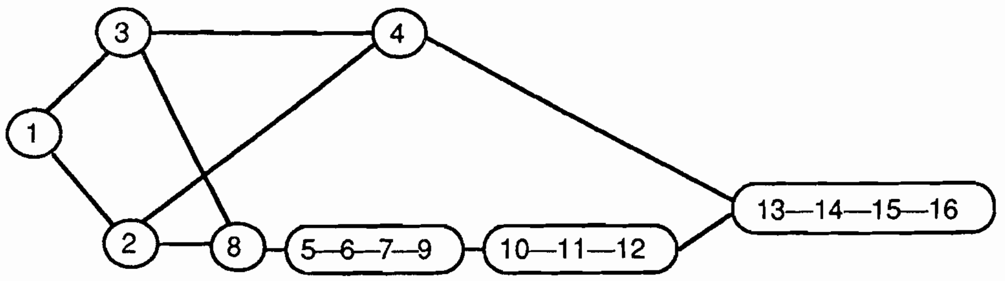
Now rule 1 can be applied several times, combining nodes 6 with 7 and 13 through 16, which leads to the following generalized natural deduction graph.



No further applications of rule 1 being possible, rule 2 must be set to work. There are several nodes with only one successor, but only (5) and (10) have a minimal number of predecessors (one in this case). Therefore we combine (5) and (10) with their respective successors. After this the direct links from one to (5) and (10) are no longer needed, as both of the new nodes also depend on (2), viz.



Rule 1 can still not be applied, and, with only one predecessor, (5-6-7) and (10-11) are the minimal nodes for rule 2, so that the graph becomes



Next rule 1 is used to combine (8), (5-6-7-9), and (10-11-12), after which some further applications of rule 2 result in the following linearized version of the proof:

1—2—3—8—5—6—7—9—10—11—12—4—13—14—15—16

The method of structuring natural deduction proofs described above does not always lead to nicely structured proofs. Even in this small example there are choice points where the rules do not suggest a unique continuation, and although the result is a totally ordered sequence no internal structure has become visible at all. The proof may be too short for the introduction of a lemma, but it is still sensible to regard line 9, the fact that the unit element e is a member of the subset S, as a natural subgoal.

With Chester's method, [Ch75], this is difficult to arrive at, as he starts his transformation process from a given natural deduction proof having no information of how the proof was found. However, if the information is exploited of how the original proof was constructed, subgoals and proof structure can be achieved in a more natural way. Lemmata can either be found during the process of finding the proof, or later by analyzing the topological structure of the refutation graph.

The new method proposed to find the underlying structure of the proofs by detecting lemmata and dividing the proof into several cases is developed in the following sections 4.3 through 4.5, but before we must insert a section on "trivial" and "obvious" proofs in order to be able to decide whether a subgoal is complicated enough to make it worth while to be mentioned explicitly in the full proof.

4.2 Trivial Subproofs

In the following sections 4.3 through 4.5, subgraphs of the original refutation graph will be viewed as deduction graphs representing lemmata in a larger proof. Obviously, this only makes sense, when the deduction graph in question is complex enough to warrant the introduction of a lemma. Otherwise it may be better to repeat a trivial argument instead of using a lemma. It is of course not straightforward to decide which deduction graph (or lemma) is non-trivial. To make a decision we use a heuristic approach taking into account several properties of the refutation graph and its subgraph, the deduction graph proving the lemma.

It is indeed not easy to find objective criteria to decide when a proof is trivial. In [Da81] Martin Davis proposes that

> *"an inference is obvious, precisely when a Herbrand proof of its correctness can be given involving no more than one substitution instance of each clause"*.

Jeffrey Pelletier and Piotr Rudnicki argue along the same line in [PR86], but point out that in general it may be difficult to decide if any proof of a given fact is non-obvious because this requires to check a property of all possible proofs. This doesn't pertain to our case, however, since we are only concerned with the question if a given proof is trivial as opposed to the question whether an obvious proof can be found for a given theorem.

So Davis' approach seems to be a good starting point, however there is an additional complication. We have to figure out whether a given proof (deduction graph) is a substantial part of a larger proof. When this is the case, it is normally desirable to use the subgraph as a lemma or as an intermediate step in the overall proof. Therefore we must check if the rest of the proof – after removing the proof of the lemma – has become "easier". According to Davis, this will be the case when the subgraph contains an instance of a clause, of which a different instance appears somewhere else in the rest of the proof. It may even be the case that both resulting proofs are obvious although the total proof wasn't. But that's what dividing large proofs into steps is all about.

Finally, when it comes to make someone understand a proof, other non-logical properties must also be taken into consideration. For example its absolute length and the length in relation to the total proof must be taken into account. When the subproof is relatively long, it will always pay to prove it separately as a lemma. If this lemma is already known to the reader one may later dispense with its proof altogether. Doing this intelligently requires a database of known lemmata and a model of the reader's knowledge about the field of mathematics under consideration. When a freshman uses the system as an explanation for a proof one may not omit arguments which a graduate student might consider trivial. Conversely, it may obscure the idea of a complex proof to mention all the applications of lemmata that have been thoroughly understood long before.

As one never knows, however, who will read the proof later, it is useful to postpone this decision as long as possible. At this stage it is not yet necessary to take a user model into account, this will only be done when the natural deduction proof is finally linearized by applying the method described in section 4.1 to arrange the lemmata in a linear order and, of course, to linearize their internal structure if they are considered complex enough for the prospective reader.

4.3 Shared Subgraphs as Lemmata

Now we assume that a proof of a formula φ has already been found by an automated deduction system. We will further assume that this proof is represented as a refutation graph Γ; this representation can easily be constructed from a given resolution proof, see [Po85] or [Le88]. In addition to the refutation graph, in order to establish a correspondence between the literal nodes of Γ and the atom occurrences within φ, we need a clause graph relation $\Delta \subseteq \Omega_a(\varphi) \times \mathbb{N}$, which must of course have been maintained throughout the search for a proof, especially during the process of normalization of the original formula.

An initial "trivial" generalized natural deduction proof (GNDP) can then be constructed to start a transformation process as described in chapter 3. Now some of the transformation rules, E$\wedge$ for instance, lead to additional external lines, and as a consequence to a division of the refutation graph according to the splitting theorem 3.6-6. In the simplest case the refutation graph proving $F \wedge G$ is "cut" through the clause $[-F -G]$, such that the two resulting components are refutation graphs for F and G, respectively. In general, however, the two components may have a non-empty intersection, and this is similarly the case for the other rules leading to a division of the refutation graph. The splitting theorem does not take this into account, so that these shared subgraphs are always duplicated and therefore processed more than once.

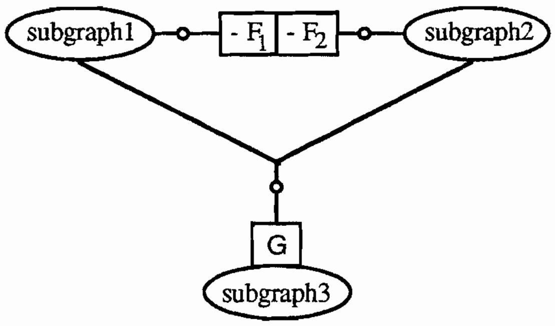
This does not matter if the intersection is comparatively small, when it may easily be copied and later used several times in the resulting subproofs. If it is relatively large and complex, however, it may be sensible to prove a lemma first and then use it in all the proof parts. In order to formalize such a procedure, a new transformation rule E-Lemma is introduced.

<u>E-Lemma:</u>

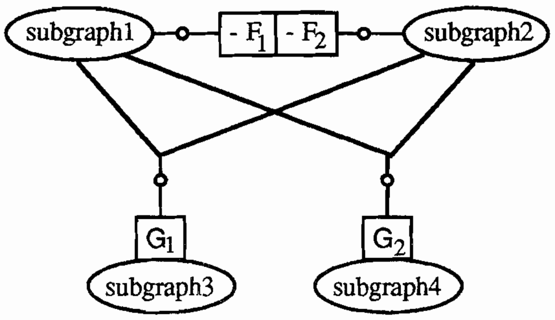
$$\begin{array}{llll} (\beta_1) & \mathcal{A}_1 & \vdash F_1 & \pi_1 \\ & & \dots & \\ (\beta_n) & \mathcal{A}_n & \vdash F_n & \pi_n \end{array} \quad \longrightarrow \quad \left\{ \begin{array}{llll} (\alpha) & \bigcap \mathcal{A}_i & \vdash G & \pi_0 \\ (\beta_1) & \mathcal{A}_1 & \vdash F_1 & \pi_1' \\ & & \dots & \\ (\beta_n) & \mathcal{A}_n & \vdash F_n & \pi_n' \end{array} \right.$$

This rule must of course be used with discretion, i.e. only when specifically called for by a heuristic. In particular it may only be applied when all the literal nodes in the refutation graph π_0 are positively polarized, so that it is possible to prove G from axioms and current assumptions only. It goes without saying that π_0 must be a common subgraph of all the graphs π_i. In constructing the graphs π_i' one is entitled to use the formula G as an additional axiom. The case $n = 1$ may also be meaningful, when a lemma is introduced as a subgoal, see section 4.3.

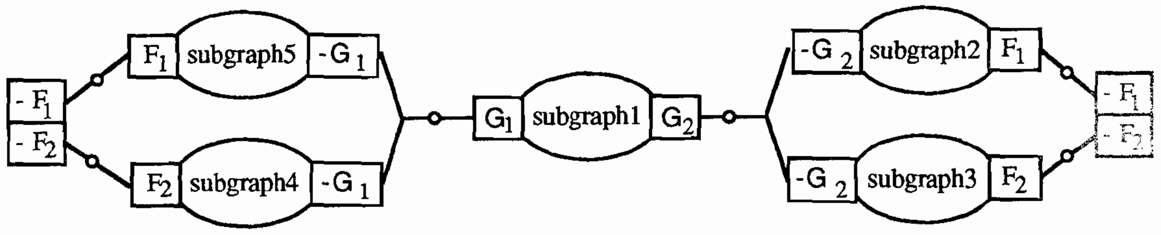
Let us consider now what these shared subgraphs may look like. We always assume that a cut is being made in order to apply E$\wedge$ to an external proof line with conclusion $F_1 \wedge F_2$. In the simplest case the lemma consists of just one atom G. Then the graph has the form



The case where G is a conjunction $G_1 \wedge G_2$ is almost as simple. It only means that there are now two independently shared parts, viz.

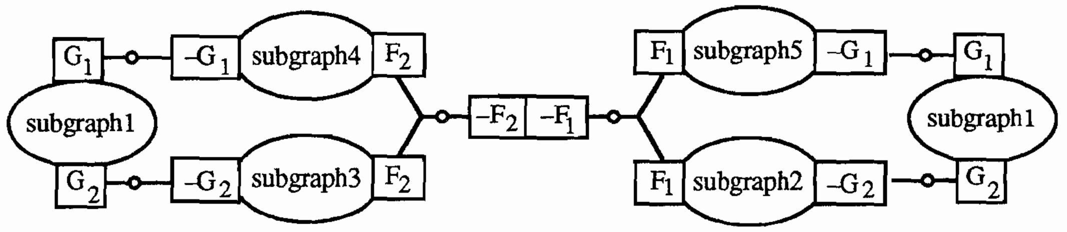


When G is a disjunction $G_1 \vee G_2$, however, things are no longer as easy. At this point one should recall, that a deduction graph constitutes a derivation of the disjunction of its pure literal nodes from the set of clauses it contains, cf. chapter 2. Therefore one might think that it suffices to introduce a link between the subgraphs 3 and 4 of the previous case, combining them to a new deduction graph. It is true that we would then know a derivation for the disjunction $G_1 \vee G_2$, but we have also introduced a cycle into the graph, which therefore ceases to be a refutation graph. As a matter of fact, a shared subgraph representing a disjunction can only happen, when the theorem $F_1 \wedge F_2$ appears more than once in the graph, as in the following example:



After cutting through both clause nodes $[-F_1 -F_2]$ the graph divides into four components, only two of which are needed. The "mixed" components containing both clause nodes $[-F_1]$ and $[-F_2]$ may be discarded. The other two components represent proofs for F_1 and F_2, respectively. Both proofs can then for instance be done by case analysis after the lemma $G_1 \vee G_2$ has been introduced.

According to theorem 2.5-7 every refutation graph can be rewritten such that a given (ground) clause corresponds to just one clause node in the graph. If one chooses the theorem clause $[-F_1 -F_2]$ to appear only once, the refutation graph of the last example takes a form as shown below. Now the subgraph proving $G_1 \vee G_2$ is no longer really shared, but two copies of it exist in the refutation graph.



So in general one has to search for isomorphic subgraphs that are complex enough to warrant the introduction of a lemma. In addition to an isomorphic graph structure all the literal nodes must represent identical literals and they must be related to the same atom occurrences.

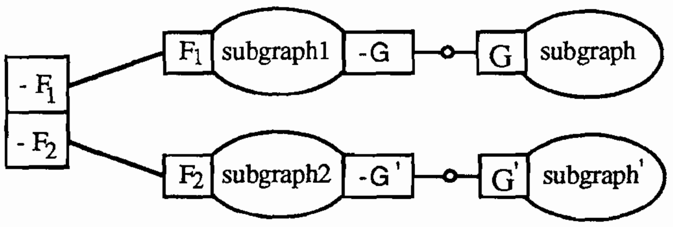
We have now seen how a deduction graph Δ (as a subgraph of a refutation graph Γ) must look like in order to represent a conjunctive or a disjunctive lemma. If a quantified formula is to be represented it is useful to leave the variables in the graph, i.e. all the clause nodes are (renamed) copies of their parent clauses, and a ground substitution γ transforms Γ into a (ground) refutation graph.

If an existentially quantified formula is represented the pure literal nodes of the corresponding deduction graph will contain a constant symbol that does not occur anywhere in the rest of the refutation graph (before applying γ). This means that the proof is independent of the actual constant symbol.

More interesting are deduction graphs representing a universally quantified formula. In this case it must be possible to replace a subterm s of a pure literal node by any ground term and change γ accordingly without violating the polylink condition. Then the lemma is the formula where s is replaced by a universally quantified variable.

If such a lemma is used in different instantiations during the proof the corresponding deduction graph will have to be duplicated as in the case of a disjunctive lemma above. Now all these subgraphs are isomorphic in structure, but they represent literals that may differ in the terms that were inserted in the place of s.

Such a lemma corresponds to a resolvent used more than once (and in different instantiations) during the resolution proof. Thus, if the refutation graph was originally constructed from a resolution proof one should keep this information in order to obviate the search for that kind of lemma. An example can easily be constructed by slightly altering the above graph where the atoms G and G' may differ in some of their arguments.



4.4 Subgoals Defined by Separating Links

In the previous section, the main incentive for the introduction of a lemma was to avoid an unnecessary duplication of parts of the proof. But this is not the only reason why mathematicians use lemmata. In many cases they are used purely to structure the proof, so that the idea behind a proof becomes better visible.

In an automatic proof transformation the difficulty is obviously to find meaningful lemmata. And it is here again that the topological structure of the refutation graph may successfully be exploited. The task is to find parts of the refutation graph that are sufficiently complex in order to justify the introduction of a lemma, while they should at the same time be easily separable from the rest of the graph. Besides, all the parts belonging to the proposed lemma must of course have been positively polarized before.

If it were possible to find a link or a small set of links separating the refutation graph, and fulfilling the above requirements, one might use the positively polarized part as a lemma. So, during the process of proof transformation, when no more automatic external rules nor E∃-constructive can be applied, the search for such links is done using the following algorithm:

4.4-1 Algorithm: (find separating links to define lemmata)

1. Starting with a non-trivial refutation graph $\Gamma=(\mathbb{N},\mathbb{C},\pounds,\Pi)$, we want to compute a set Ψ of candidates for a lemma. The process is initialized by setting $\Psi:=\{\}$.

2. Compute the set $\Psi_{min}:=\{\Lambda\in\Pi \mid \Lambda$ is minimal with respect to the nesting order $\overset{*}{\prec}$ on $\Pi\}$. These are the links separating the graph. The removal of a given $\Lambda\in\Psi_{min}$ from Γ results in two deduction graphs Γ_Λ^+ and Γ_Λ^-.

3. Compute $\Psi_0:=\{\Lambda\in\Psi_{min} \mid$ exactly one of Γ_Λ^+ or Γ_Λ^- is trivial$\}$. These are useless for lemma purposes, since they can only lead to trivial lemmata. Therefore let $\Psi_{min}:=\Psi_{min}\setminus\Psi_0$.

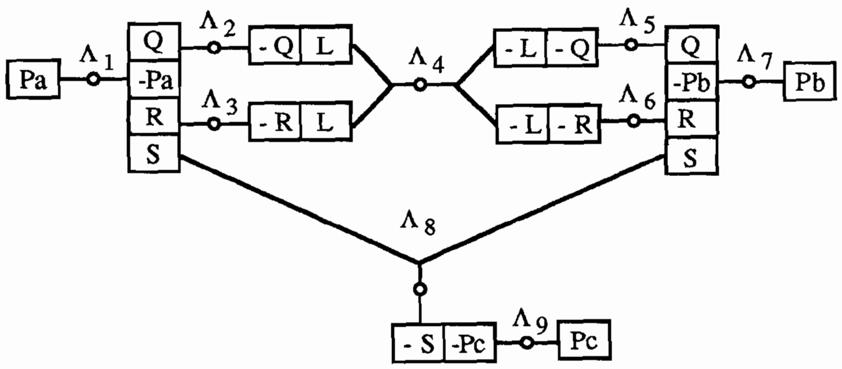
4. Let $\Psi:=\Psi\cup\Psi_{min}$.

5. For each $\Lambda\in\Psi_0$ duplicate the trivial subgraph Γ_Λ^{tr}, if Λ is branching on the non-trivial side. Note that Γ is changed by this operation. Let $\Psi_{min}:=\{\Theta\in\Pi_\Lambda^{ntr}\setminus\Psi_{min} \mid \Theta$ is minimal with respect to the nesting order $\overset{*}{\prec}$ on $\Pi_\Lambda^{ntr}\}$. These are the links that additionally separate the graph after the duplication of a subgraph.

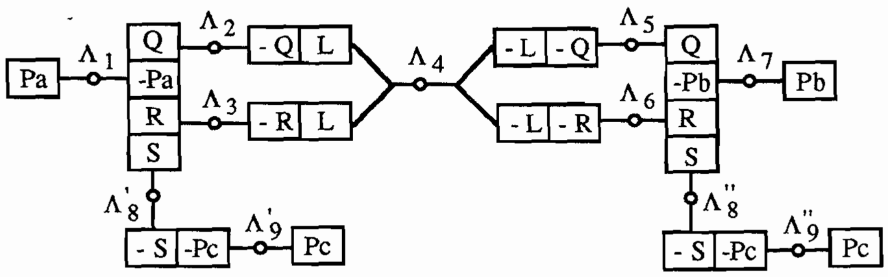
6. If Ψ_{min} is non-empty, go to 3, otherwise continue with 7.

7. Now Ψ is the complete set of candidates for lemma purposes.

4.4-2 Example: (links defining a lemma)



The set of separating links in the example graph is $\Psi_{min}=\{\Lambda_1, \Lambda_7, \Lambda_8, \Lambda_9\}$. All of them separate a trivial part from the rest of the graph, though, therefore none of them is among the candidates for a lemma. Only Λ_8 is branching on the non-trivial side, which means that the appropriate trivial part has to be duplicated. This leads to the following refutation graph.



Now Λ_4 is an additional separating link, which divides the graph into two trivial parts. So $\Psi=\{\Lambda_4\}$ is the set of candidate links for a lemma. ♦

Now the set Ψ of candidate links for the construction of a lemma has been computed, and unless it is empty, we must define an actual lemma by choosing one or several of these links and their related subgraphs. In order to do this, we must try to isolate parts of the graph containing only positively polarized clause nodes, which are connected to the rest of the graph only via Ψ-links. The following algorithm describes, how this is done.

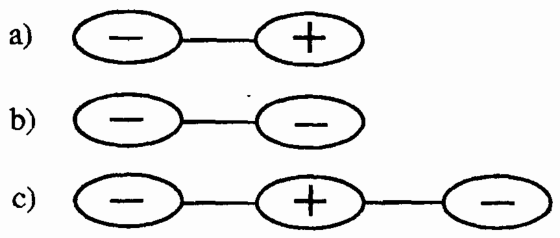
4.4-3 Definition: (maximal connected subgraphs)

Let $\Gamma=(\mathbb{N},\mathbb{C},\pounds,\Pi)$ be a clause graph and let $\Sigma(\Gamma)$ be the set of all subgraphs of Γ. Let $\Psi\subseteq\Pi$ be a set of links. Then for $C\in\mathbb{C}$ we can define $\Sigma_\Psi(C)$ as the *maximal connected subgraph* of Γ,which includes C but does not contain any $\Lambda\in\Psi$.

4.4-4 Algorithm: (choose lemma from separating links)

1. For all negatively polarized clause nodes C^- compute $\Sigma_\Psi(C^-)$. These graphs are deduction graphs, whose pure literal nodes were connected to the rest of Γ with Ψ-links.

2. For all positively polarized clause nodes C^+, not belonging to any of the $\Sigma_\Psi(C^-)$, compute $\Sigma_\Psi(C^+)$. Now the graph is divided into "positive" and "negative" subgraphs in one of three basically different ways:



In all three cases, variations may occur due to separating links that are branching.

3a) If there is only one $\Sigma_\Psi(C^-)$, the set of links attached to its pure literal nodes in Γ is used as a lemma, which can be derived from the rest of the graph, i.e. directly from axiom formulae or assumptions. In this case the lemma is the conjunction of the literal nodes in the opposite shores of the pure literal nodes.

b) If two of the $\Sigma_\Psi(C^-)$ are adjacent, then the proof is separated into cases (see next section).

c) If there is a $\Sigma_\Psi(C^+)$ between two of the $\Sigma_\Psi(C^-)$, then one has to check, if the positive part consists only of a trivial chain of ψ-links, in which case one proceeds as in b). Otherwise the positive subgraph defines a disjunctive lemma, which will then be used to perform a proof by case analysis.

4.4-5 Example: (choosing the actual lemma)

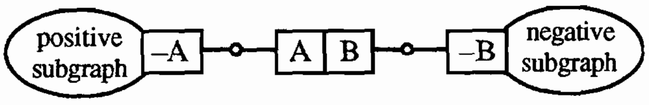
In the case of the previous example 4.4-2, the only link suggesting a lemma was Λ_4. The polarity of the resulting parts now decides the actual form of the lemma. If, for instance, [Pb] is the only negatively polarized clause node, then L is the lemma. If

[Pa] and [Pb] are both negatively polarized, it is best to conduct the proof by the cases L and –L.

4.5 Structuring Proofs Using Proof by Case Analysis

M-Cases, one of the transformation rules defined in chapter 3, leads to a division of the refutation graph by dividing an assumption formula. This rule can always be applied, when a disjunction has been derived earlier. An application is however undesirable in many cases, as can be seen from the following examples:

(a) If only one of the resulting components contains negatively polarized literal nodes, then an extra and unnecessary proof by contradiction must be performed.



Here the case B is straightforward, but A needs a proof by contradiction.

(b) If both of the resulting parts overlap widely, including negatively polarized literal nodes, then large parts of the proof will be duplicated in both cases.

A good case for the application of M-Cases appears, when both of the resulting components contain parts of the theorem, and their overlap is either small or restricted to positively polarized parts, in which case a lemma can be defined to avoid the duplication (cases 3b and 3c in the previous section).

This is the case when the formula to prove is distributed in the refutation graph with respect to the disjunctive formula which shall be used for case analysis, so that the graph meets the condition of lemma 3.6-8.

The most important case for the rule M-Cases comes up, when an existentially quantified formula cannot be proven constructively. In the refutation graph, this fact is reflected by the existence of several different instances of the theorem clauses. Similarly, when a disjunction has to be proven, case analysis may be the best solution. M-Cases can be applied with advantage if any of the components resulting from splitting contains only literal nodes related to one instance of the existentially quantified theorem or one part of the disjunction.

4.6 Example & Further Processing

A famous example in the literature on automatic theorem proving is a problem which Lenhart Schubert formulated in 1978 as a challenge to automated deduction systems. It became well-known as "Schubert's Steamroller" and for several years presented a hard problem even for the best automatic theorem provers.

For two reasons it is also an interesting problem for proof transformation and structuring. For once it is a problem stated in natural language so that there is an obvious semantics for all the formulae, which makes it easy to check whether any lemmata or subgoals are "intuitive" or not. Secondly, the problem is combinatorially difficult, and computer generated proofs involve many clauses and are difficult to understand.

Schubert's original problem,

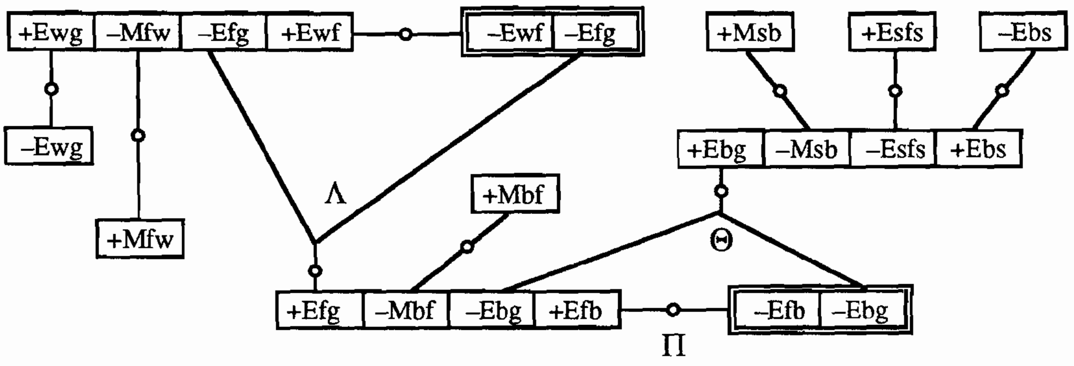
> *Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also there are some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars, but not snails. Caterpillars and snails like to eat some plants.*
>
> *Therefore there is an animal that likes to eat a grain-eating animal.*

has been coded in first order logic in several different ways. We adopt a formulation proposed by Mark Stickel in [St86] where "a grain-eating animal" means an animal that eats some grain. To make the example easier to understand we use sorts, and the names of the variables and constants indicate of which sort they are. E and M are binary relations, where Exy means "x likes to eat y" and Muv means "u is much smaller than v". Then the following first order formula represents the steamroller problem:

$$
\begin{array}{ll}
 & ((\forall a\ \ (\forall p\, Eap) \vee (\forall a'\, Ma'a \wedge \exists p'\, Ea'p' \Rightarrow\ Eaa')\,) \\
\wedge & (\forall c,s,b,f,w\ Mcb \wedge Msb \wedge Mbf \wedge Mfw) \\
\wedge & (\forall w,f,g\ \neg Ewf \wedge \neg Ewg) \\
\wedge & (\forall b,c,s\ Ebc \wedge \neg Ebs)
\end{array}
$$

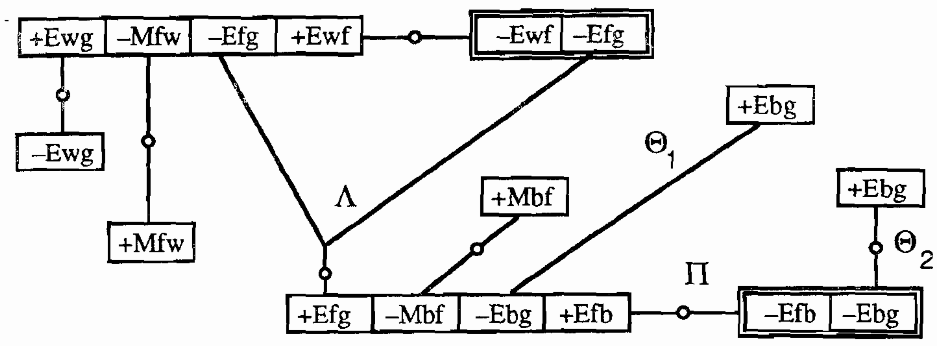
$\wedge \quad (\forall c \exists p\, Ecp \wedge \forall s \exists p'\, Esp'))$
$\Rightarrow \quad (\exists a,a',p\; Eaa' \wedge Ea'p)$

A refutation graph proving this formula is shown below. A resolution proof has been automatically found by our theorem prover "Markgraf Karl Refutation Procedure" [MKRP84], and the graph was generated automatically as described in [Le88].



The polarization is shown as it is after application of the first rule, E⇒, which moves all the axioms to the left side as assumptions. At this point the formula to prove is an existentially quantified formula, and the total substitution tells us that the proof is not done constructively. Therefore algorithm 4.4-1 is used to find any separating links defining a lemma. Only Λ and Θ separate non-trivial parts from the refutation graph, and an analysis of the polarization shows that Θ segregates a positive part from the rest, so that a first lemma Ebg, i.e. "birds eat grain" can be generated.

Actually there is a universally quantified lemma $\forall p\, Ebp$ as discussed in section 4.3. The subterm g can be replaced by a variable without violating the polylink condition in the deduction graph proving Ebg. As this formula is not needed in any other instantiation there is nothing to choose between Ebg and $\forall p\, Ebp$ as a lemma. After the lemma has been proven, the clause node [Ebg] representing it can be duplicated with the effect that in addition to Λ, Π becomes separating, viz.



This time situation 3b) of algorithm 4.4-4 is present, so we derive the lemma Efg ∨ Efb, which means that a fox eats either grain or birds. This formula is then used to divide the rest of the proof into two cases. In the first case (Efg) the theorem holds because wolves eat foxes and these eat grain; in the other case (Efb) foxes eat birds, which in turn eat grain, so that the theorem is fulfilled. The basic structure of the proof turns out to be:

$$\begin{tikzcd}
 & & \text{Axioms} \arrow[d] & & \\
 & & \text{Birds eat grain (Lemma 1)} \arrow[d] & & \\
 & \text{Foxes eat birds} & \text{OR} & \text{Foxes eat grain} \arrow[dd] & \\
\underline{\text{Case 1:}} & \text{ok} \arrow[ddr] & & & \underline{\text{Case 2:}} \\
 & & & \text{Wolves eat foxes} \arrow[dl] & \\
 & & \text{Therfore: } \exists\, a, a', p \ a \text{ eats } a' \text{ AND } a' \text{ eats } p & &
\end{tikzcd}$$

Here one might ask why the proof had to be done in such a roundabout way, the second of the cases above seems to make no sense. And indeed, there is a constructive proof of the theorem, which uses the fact that "wolves don't eat foxes" to show that case 2 is impossible. But then the proof would be completely different from the one found by MKRP[1], where this fact was not used at all. And proof transformation is not supposed to find the "best" proof, but to rewrite a given proof in a different form.

The transformation results in a natural deduction proof, where it is known that certain lines are lemmata or subgoals in the proof.

(1)	1	$\vdash$	$\forall a\,(\forall p\ Eap) \vee (\forall a'\ Ma'a \wedge \exists p'\ Ea'p' \Rightarrow Eaa')$	Ass
(2)	2	$\vdash$	$\forall c,s,b,f,w\ Mcb \wedge Msb \wedge Mbf \wedge Mfw$	Ass
(3)	3	$\vdash$	$\forall w,f,g\ \neg Ewf \wedge \neg Ewg$	Ass
(4)	4	$\vdash$	$\forall b,c,s\ \neg Ebs \wedge Ebc$	Ass
(5)	5	$\vdash$	$\forall c\,\exists p\ Ecp \wedge \forall s\,\exists p'\ Esp'$	Ass
(6)	2	$\vdash$	Msb	$\forall\wedge E(2)$
(7)	5	$\vdash$	$\exists p'\ Esp'$	$\wedge\forall E(5)$
(8)	4	$\vdash$	$\neg Ebs$	$\forall\wedge E(4)$
(9)	2,4,5	$\vdash$	$\neg\,(Msb \wedge \exists p'\ Esp' \Rightarrow Ebs)$	Tau(6,7,8)
(10)	2,4,5	$\vdash$	$\exists a'\ \neg\,(Ma'b \wedge \exists p'\ Ea'p' \Rightarrow Eba')$	$\exists I(9)$
(11)	2,4,5	$\vdash$	$\neg\,\forall a'\ (Ma'b \wedge \exists p'\ Ea'p' \Rightarrow Eba')$	Neg(10)
(12)	1	$\vdash$	$\forall p\ Ebp \vee (\forall a'\ Ma'b \wedge \exists p'\ Ea'p' \Rightarrow Eba')$	$\forall E(1)$
(13)	1,2,4,5	$\vdash$	$\forall p\ Ebp$	Tau(11,12)
(14)	1,2,4,5	$\vdash$	Ebg	$\forall E(13)$

(15)	2	$\vdash$	Mbf	$\forall\wedge E(2)$
(16)	1,2,4,5	$\vdash$	$\exists p'\ Ebp'$	$\exists I(14)$
(17)	1	$\vdash$	$(\forall p\ Efp) \vee (\forall a'\ Ma'f \wedge \exists p'\ Ea'p' \Rightarrow Efa')$	$\forall E(1)$
(18)	1	$\vdash$	$(\forall p\ Efp) \vee (Mbf \wedge \exists p'\ Ebp' \Rightarrow Efb)$	$\vee\forall E(17)$
(19)	1,2,4,5	$\vdash$	$(\forall p\ Efp) \vee Efb$	Tau(15,16,18)
(20)	1,2,4,5	$\vdash$	$Efg \vee Efb$	$\forall E(19)$

[1] The Markgraf Karl Refutation Procedure finds this proof first because of the "Set-of-Support" strategy. There are even problems, where certain proofs cannot be found at all, if the "Set-of-Support" strategy is used in combination with subsumption.

We consider separately the cases of (20)

Case 1:

(21)	1,2,4,5,20	$\vdash$	Efg	Ass
(22)	3	$\vdash$	$\neg$Ewg	$\forall\wedge$E(3)
(23)	1	$\vdash$	($\forall$p Ewp) $\vee$ ($\forall$a' Ma'w $\wedge$ $\exists$p' Ea'p' $\Rightarrow$ Ewa')	$\forall$E(1)
(24)	1	$\vdash$	Ewg $\vee$ ($\forall$a' Ma'w $\wedge$ $\exists$p' Ea'p' $\Rightarrow$ Ewa')	$\vee\forall$E(23)
(25)	1,3	$\vdash$	$\forall$a' Ma'w $\wedge$ $\exists$p' Ea'p' $\Rightarrow$ Ewa'	Tau(22,24)
(26)	1,3	$\vdash$	Mfw $\wedge$ $\exists$p' Efp' $\Rightarrow$ Ewf	$\forall$E(25)
(27)	2	$\vdash$	Mfw	$\forall\wedge$E(2)
(28)	1,2,4,5,20	$\vdash$	$\exists$p' Efp'	$\exists$I(21)
(29)	1-5,20	$\vdash$	Ewf	Tau(26,27,28)
(30)	1-5,20	$\vdash$	Ewf $\wedge$ Efg	$\wedge$I(29)
(31)	1-5,20	$\vdash$	$\exists$a,a',p Eaa' $\wedge$ Ea'p	$\exists$I(30)

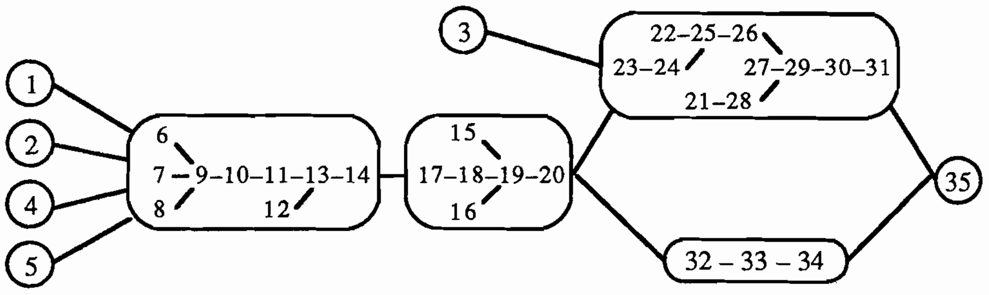
Case 2:

(32)	1,2,4,5,30	$\vdash$	Efb	Ass
(33)	1,2,4,5,30	$\vdash$	Efb $\wedge$ Ebg	$\wedge$I(14,32)
(34)	1,2,4,5,30	$\vdash$	$\exists$a,a',p Eaa' $\wedge$ Ea'p	$\exists$I(33)

End of Cases (23,34)

(35)	1-5	$\vdash$	$\exists$a,a',p Eaa' $\wedge$ Ea'p	$\vee$E(20,31,34)

We can then start the process of linearization with a prestructured generalized natural deduction graph, where only the lemmata have to be arranged in a linear order, and of course, their internal structure must be linearized. Using this information, our starting point is the following GNDG:



Now the problem of imposing a total order on the natural deduction proof has become much simpler. At first a total order is imposed upon the global proof structure, for which the only decision to be made is the sequential order of the two cases in the case analysis. Then the internal structure of the nodes has to be linearized, which again is not difficult as their structure is simple enough.

The linearization of this proof starting with an unstructured natural deduction graph by the method explained in 4.1-2 also leads to the subgoals 20 and 31, but misses 14 and ends up with two additional meaningless lemmata. The details of the linearization process are described in Appendix D.

Now that the natural deduction graph has been linearized, some further steps are required in order to make the proof really understandable. The main drawback of natural deduction proofs is their length, and the size of the individual steps is too small, remaining altogether on a purely syntactical level. One has to raise therefore the granularity of single reasoning steps to a "conceptual level", which is not straightforward, as the solution depends on the context of the proof as well as on the knowledge of the intended reader.

An interesting approach has been put forward by Xiaorong Huang in [Hu89a] and [Hu89b], who builds upon the work proposed in this thesis. He combines several steps in the natural deduction system so that they represent the "application" of an axiom or a lemma. This typically involves the combination of the instantiation of premise formulae together with some propositional reasoning to a single proof step.

When all of this is done one can then tackle the problem to state the formal proof in the natural language of a mathematician. The following is the result of a hand-simulation of Huang's proposed transformation steps. A proof in (mathematical) natural language might then read as follows, the numbers in parentheses indicate the line number of the natural deduction proof.

<u>Axioms:</u> *Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also there are some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants (1). Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves (2). Wolves do not like to eat foxes or grains (3), while birds like to eat caterpillars, but not snails (4). Caterpillars and snails like to eat some plants (5).*

<u>Theorem:</u> *Therefore there is an animal that likes to eat a grain-eating animal (35).*

Proof: *It is given that snails are much smaller than birds (6) and they eat some plant (7), but birds don't eat snails (8). Now every animal either likes to eat all plants or all herbivorous animals much smaller than itself (1). Therefore birds like to eat all plants (13), and in particular, they like to eat grains (14). [lemma 1]*

Birds are also much smaller than foxes (15), hence foxes eat birds or they eat all plants (19), especially grains (20). [lemma 2]

Let us assume first that foxes eat grains (21). Now wolves don't eat grain (22) and must therefore eat all smaller herbivorous animals (25), in particular they eat foxes (29). Hence there is an animal (the wolf) eating a grain-eating animal (the fox) (31). [end of case 1]

If on the other hand foxes eat birds (32), which by lemma 1 are known to eat grains (14), we know again of an animal (the fox) who eats grain-eating animals (34) [end of case 2], which completes the proof.

q.e.d.

5 Conclusions and Open Problems

In this thesis a method is described to transform a proof represented in the resolution calculus or as a refutation graph into a natural deduction proof in Gentzen's system NK. Starting from a basic proof transformation mechanism as published by Peter Andrews [An80], Dale Miller [Mi83], Christoph Lingenfelder [Li86], Amy Felty [Fe86], or Frank Pfenning [Pf87], the necessary changes and additions are made to meet the special needs of the transformation guided by refutation graphs.

The proof representation in form of refutation graphs lends itself perfectly well to avoiding proofs by contradiction and to deciding when an "integral" formula can be proven undivided, i.e. without breaking it up into its very literals. The introduction and formalization of additional, and sometimes incomplete, transformation rules, as for instance M-Inf, is made possible because the topology of the refutation graph, in connection with the clause graph relation that ties it to the original formula, allows an efficient check of their applicability. The refutation graph also helps to decide in an intelligent way, when to do a proof by case analysis using a disjunction, or it helps to choose a suitable formula G when an axiom $(G \vee \neg G)$ must be introduced.

In addition to the technical questions of proof transformation this thesis shows how to organize a proof by breaking it up into smaller lemmata, thereby laying open its general structure and its main ideas. In particular this may avoid the need to prove a subgoal more than once, when it is shared by different branches of the proof. To this end both the information implicit in the topological properties of refutation graphs and the history of finding the proof is used, for instance in form of resolvents that were used in different instantiations during the proof.

This is done by dividing the graph into disjoint parts to be proved separately, either sequentially, as a lemma cited later in the proof, or as a proof by case analysis. In order to achieve this, the algorithm for the transformation of refutation graphs into natural deduction proofs had to be extended.

The same information also facilitates the process of linearizing natural deduction proofs. The parts to be linearized (lemmata) are much smaller, thus reducing the number of arbitrary decisions that have to be made in order to choose an actual sequence of proof lines. In fact one has to solve several smaller linearization problems instead of a single large one; of course one also has to linearize the sequence of lemmata or proof cases in the end.

Parallel to the development of the theory we implemented a proof transformation system to check many of the ideas and heuristics, which made it possible to evaluate the additional transformation rules and the lemma mechanism. In a large number of examples the resulting lemmata actually made sense semantically.

An open question with respect to the structuring of proofs is the presentation of equality proofs, regardless whether they were found by rewriting techniques or by paramodulation. The representation of pure unconditioned equality proofs in equality graphs, as in Karl-Hans Bläsius' dissertation, [Bl86], seems to be a promising starting point to construct a procedure analogous to the algorithm developed in this thesis.

Also the extension of the proof transformation process to resolution with paramodulation and to theory resolution in general remains to be dealt with, but this appears to be a straightforward affair. The main problem will be to extract the necessary information. Theory resolution was invented to incorporate the use of algorithmic knowledge for decidable subproblems; therefore not all the proof steps are made explicit and a more or less important part of the proof will be hidden by the algorithm.

The most important future research topic is, of course, the further transformation into natural language. The final natural language proof of example 4.6 was hand-simulated, but in its origination we employed an approach of Xiaorong Huang, cf. [Hu89a] and [Hu89b], to combine several applications of inference rules so that the reasoning is raised to a "conceptual" level. In the later stages of the transformation one has to apply methods of natural language processing. For every single argument one has to decide how it should be presented. Is it necessary to repeat all or some of the premise formulae? Should the rule itself be mentioned or is it self-evident in the context?

A focus mechanism can be used to minimize the need for repetition of formulae. If at all possible one should try to construct a chain of arguments, so that the result of one step is a direct prerequisite of the next one. These are techniques well-known from natural language processing, cf. for example [McK85], that can be used for the purpose of proof presentation as well as in the context of speech planning.

Finally one must take into account the prospective reader of the proof. After all, a mathematician will consider a lot of proof steps trivial, that inexperienced readers might not find easy at all. This raises the question how this distinction can be made automatically. For one thing a (knowledge-)base of previously proved theorems is of the essence, allowing simply to cite a lemma and doing away with the respective part of the proof altogether. But in general this can only be done with the help of a user model. So it turns out that proof transformation is not purely a question of logic after all, but that methods of different fields of Artificial Intelligence are required to come up with a really satisfying result.

A Literature

[An76]	Peter B. Andrews	*Refutations by Matings* JACM **15**, 3 (1983)
[An80]	Peter B. Andrews	*Transforming Matings into Natural Deduction Proofs* Lecture Notes in Comp. Sci. **87** (CADE 1980), 281-292
[An81]	Peter B. Andrews	*Theorem Proving via General Matings* Journal of the ACM **28**, (1981), 193-214
[Bi81]	Wolfgang Bibel	*On Matrices with Connections* Journal of the ACM **28**, (1981), 633-645
[Bi82]	Wolfgang Bibel	*Automated Theorem Proving* vieweg, Braunschweig, Wiesbaden (1982)
[Bl86]	Karl-Hans Bläsius	*Equality Reasoning Based on Graphs* PhD Thesis, Universität Kaiserslautern, 1986 SEKI Report SR-87-01
[Ch75]	David Chester	*The Translation of Formal Proofs into English* Artificial Intelligence **7** (1976), 261-278
[Da81]	Martin Davis	*Obvious Logical Inferences* Proc. of the 7th IJCAI, Vancouver 1981
[De71]	Peter Deussen	*Halbgruppen und Automaten* Heidelberger Taschenbücher **99**, Springer 1971
[Ga86]	Jean H. Gallier	*Logic for Computer Science* *– Foundations of Automatic Theorem Proving* Harper & Row, Publishers, New York 1986
[Ge35]	Gerhard Gentzen	*Untersuchungen über das logische Schließen I* Mathematische Zeitschrift **39**, pp.176-210, 1935
[Ei88]	Norbert Eisinger	*Completeness, Confluence, and Related Properties of Clause Graph Resolution* PhD Thesis, Universität Kaiserslautern, 1988 SEKI Report SR-88-07

[EO88] Norbert Eisinger, Hans J. Ohlbach
The Markgraf Karl Refutation Procedure
Lecture Notes in Comp. Sci. **230** (CADE 86), 682-683

[Fe86] Amy P. Felty *Using Extended Tactics to do Proof Transformations*
Master's Thesis, University of Pennsylvania, Philadelphia, MS-CIS-86-89, 1986

[He87] Alexander Herold *Combination of Unification Algorithms in Equational Theories*
PhD Thesis, Universität Kaiserslautern, 1987
SEKI Report SR-87-05

[Hu89a] Xiaorong Huang *A Human Oriented Proof Presentation Model*
SEKI Report SR-89-11,
Universität Kaiserslautern, 1989

[Hu89b] Xiaorong Huang *Proof Transformation Towards Human Reasoning Style*
Proc. of the 13th GWAI,
Informatik Fachberichte **216**, 1989

[Ja33] Stanislaw Jaskowski *On the Rules of Suppositions in Formal Logic*
Studia Logica, no. 1. Warsaw 1934

[Ko75] Robert Kowalski *A Proof Procedure Using Connection Graphs*
JACM **22**, No. 4 (1975), 572-595

[KH69] R. Kowalski, P. Hayes *Semantic Trees in Automatic Theorem Proving*
Machine Intelligence **4**, American Elsevier, New York, 1979

[KK71] R. Kowalski, D. Kuehner *Linear Resolution with Selection Function*
Artificial Intelligence **2**, pp. 227-260, 1971

[Le88] Siegfried Lehr *Transformation von Resolutionsbeweisen des MKRP*
Studienarbeit, Universität Kaiserslautern 1988

[Li86] Christoph Lingenfelder *Transformation of Refutation Graphs into Natural Deduction Proofs*
SEKI-Report SR-86-10, Universität Kaiserslautern 1986

[Li88] Christoph Lingenfelder *Structuring Computer Generated Proofs*
SEKI-Report SR-88-19, Universität Kaiserslautern 1988

[Li89] Christoph Lingenfelder *Structuring Computer Generated Proofs*
Proc. of 11th IJCAI, Detroit, 1989

[Lo78] Donald W. Loveland *Automated Theorem Proving: A Logical Basis*
North Holland, 1978

[McC88] William McCune *Otter User's Manual*
Argonne Report ANL 88-44, 1988

[McK85] K. R. McKeown *Text Generation*
Cambridge University Press, 1985

[Mi83] Dale Miller *Proofs in Higher Order Logic*
Ph.D. Thesis, Carnegie Mellon University (1983),
Tech Report MS-CIS-83-87, University of Pennsylvania, Philadelphia

[Pf87] Frank Pfenning *Proof Transformations in Higher-Order Logic*
Ph.D. Thesis, Carnegie Mellon University (1987)

[MKRP84] Karl Mark G Raph *The Markgraf Karl Refutation Procedure*
Memo-SEKI-Mk-84-01,
Universität Kaiserslautern 1984

[OS89] Hans J. Ohlbach, Jörg H. Siekmann
The Markgraf Karl Refutation Procedure
SEKI-Report SR-89-19,
Universität Kaiserslautern 1989

[PN90] Frank Pfenning, Daniel Nesmith
Presenting Intuitive Deductions via Symmetric Simplification
to appear in Proc. of 10th CADE, 1990

[Po85] Joachim Posegga *Using Deduction Graphs as a Representation for Resolution Proofs*
Diploma Thesis, Universität Kaiserslautern 1986

[Pr65] Dag Prawitz *Natural Deduction – A Proof Theoretical Study*
Almqvist & Wiksell, Stockholm, 1965

[PR86] Francis Jeffrey Pelletier, Piotr Rudnicki
Non-Obviousness, AAR Newsletter **6**, September 1986

[Ro65] J. A. Robinson *A Machine Oriented Logic Based on the Resolution Principle*
JACM **12**, No.1 (1965), pp. 23-41

[Sh76] Robert E. Shostak *Refutation Graphs*
Artificial Intelligence 7 (1976), pp. 51-64

[Sh79] Robert E. Shostak *A Graph Theoretic View of Resolution Theorem-Proving*
Report SRI International, Menlo Park, CA (1979)

[Sm68] Raymund Smullyan *First Order Logic*
Springer-Verlag, New York, 1968

[St86] Mark Stickel *Schubert's Steamroller Problem: Formulations and Solutions*
JAR, Vol.2, No.1, 1968

[WCR64] L.T. Wos, D.F. Carson, G.A. Robinson
The Unit Preference Strategy in Theorem Proving
Proc. FJCC, Thompson Book Co., New York, 1964

[WCR65] L.T. Wos, D.F. Carson, G.A. Robinson
Efficiency and Completeness of the Set-of-Support Strategy in Theorem Proving
JACM **12**, 4 (1965)

B Table of Transformation Rules

B.1 External Rules

B.1-1 Automatic External Rules

$\underline{E\land:}$

$$(\gamma)\ \mathcal{A} \vdash F\land G \quad \pi \quad \rightarrow \quad \left\{\begin{array}{lllll} (\alpha) & \mathcal{A} & \vdash & F & \pi_1 \\ (\beta) & \mathcal{A} & \vdash & G & \pi_2 \\ (\gamma) & \mathcal{A} & \vdash & F\land G & \land I(\alpha,\beta) \end{array}\right.$$

$\underline{E\Rightarrow:}$

$$(\gamma)\ \mathcal{A} \vdash F\Rightarrow G \quad \pi \quad \rightarrow \quad \left\{\begin{array}{lllll} (\alpha) & \mathcal{A}, F & \vdash & F & \text{Ass} \\ (\beta) & \mathcal{A}, F & \vdash & G & \pi' \\ (\gamma) & \mathcal{A} & \vdash & F\Rightarrow G & \Rightarrow I(\beta) \end{array}\right.$$

$\underline{E\forall:}$

$$(\beta)\ \mathcal{A} \vdash \forall x Fx \quad \pi \quad \rightarrow \quad \left\{\begin{array}{lllll} \multicolumn{5}{l}{\text{Let c be an arbitrary object}} \\ (\alpha) & \mathcal{A} & \vdash & Fc & \pi' \\ (\beta) & \mathcal{A} & \vdash & \forall x Fx & \forall I(\alpha) \end{array}\right.$$

$\underline{E\neg\neg:}$

$$(\beta)\ \mathcal{A} \vdash \neg\neg F \quad \pi \quad \rightarrow \quad \left\{\begin{array}{lllll} (\alpha) & \mathcal{A} & \vdash & F & \pi' \\ (\beta) & \mathcal{A} & \vdash & \neg\neg F & \text{Tau}(\alpha) \end{array}\right.$$

$\underline{E\neg\land:}$

$$(\beta)\ \mathcal{A} \vdash \neg(F\land G) \quad \pi \quad \rightarrow \quad \left\{\begin{array}{lllll} (\alpha) & \mathcal{A} & \vdash & \neg F\lor\neg G & \pi' \\ (\beta) & \mathcal{A} & \vdash & \neg(F\land G) & \text{Tau}(\alpha) \end{array}\right.$$

$\underline{E\neg\lor:}$

$$(\beta)\ \mathcal{A} \vdash \neg(F\lor G) \quad \pi \quad \rightarrow \quad \left\{\begin{array}{lllll} (\alpha) & \mathcal{A} & \vdash & \neg F\land\neg G & \pi' \\ (\beta) & \mathcal{A} & \vdash & \neg(F\lor G) & \text{Tau}(\alpha) \end{array}\right.$$

$\underline{E\neg\Rightarrow:}$

$$(\beta)\ \mathcal{A} \vdash \neg(F\Rightarrow G) \quad \pi \quad \rightarrow \quad \left\{\begin{array}{lllll} (\alpha) & \mathcal{A} & \vdash & F\land\neg G & \pi' \\ (\beta) & \mathcal{A} & \vdash & \neg(F\Rightarrow G) & \text{Tau}(\alpha) \end{array}\right.$$

$\underline{E\neg\exists:}$

$$(\beta)\ \mathcal{A} \vdash \neg\exists x Fx \quad \pi \quad \rightarrow \quad \left\{\begin{array}{lllll} (\alpha) & \mathcal{A} & \vdash & \forall x\neg Fx & \pi' \\ (\beta) & \mathcal{A} & \vdash & \neg\exists x Fx & \text{Tau}(\alpha) \end{array}\right.$$

B.1-2 Proof-Driven Rules

E¬:

$$(\gamma)\ \mathcal{A} \vdash \neg F \qquad \pi \qquad \longrightarrow \qquad \left\{\begin{array}{llll} (\alpha) & \mathcal{A},F & \vdash F & \text{Ass} \\ (\beta) & \mathcal{A},F & \vdash \bot & \pi' \\ (\gamma) & \mathcal{A} & \vdash \neg F & \neg I(\beta) \end{array}\right.$$

E∨left:

$$(\beta)\ \mathcal{A} \vdash F\vee G \qquad \pi \qquad \longrightarrow \qquad \left\{\begin{array}{llll} (\alpha) & \mathcal{A} & \vdash F & \pi' \\ (\beta) & \mathcal{A} & \vdash F\vee G & \vee E(\alpha) \end{array}\right.$$

E∨right:

$$(\beta)\ \mathcal{A} \vdash F\vee G \qquad \pi \qquad \longrightarrow \qquad \left\{\begin{array}{llll} (\alpha) & \mathcal{A} & \vdash G & \pi' \\ (\beta) & \mathcal{A} & \vdash F\vee G & \vee E(\alpha) \end{array}\right.$$

E∨1:

$$(\delta)\ \mathcal{A} \vdash F\vee G \qquad \pi \qquad \longrightarrow \qquad \left\{\begin{array}{llll} (\alpha) & \mathcal{A},\neg F & \vdash \neg F & \text{Ass} \\ (\beta) & \mathcal{A},\neg F & \vdash G & \pi' \\ (\gamma) & \mathcal{A} & \vdash \neg F\Rightarrow G & \Rightarrow I(\beta) \\ (\delta) & \mathcal{A} & \vdash F\vee G & \text{Tau}(\gamma) \end{array}\right.$$

E∨2:

$$(\delta)\ \mathcal{A} \vdash F\vee G \qquad \pi \qquad \longrightarrow \qquad \left\{\begin{array}{llll} (\alpha) & \mathcal{A},\neg G & \vdash \neg G & \text{Ass} \\ (\beta) & \mathcal{A},\neg G & \vdash F & \pi' \\ (\gamma) & \mathcal{A} & \vdash \neg G\Rightarrow F & \Rightarrow I(\beta) \\ (\delta) & \mathcal{A} & \vdash F\vee G & \text{Tau}(\gamma) \end{array}\right.$$

E∃-constructive:

$$(\beta)\ \mathcal{A} \vdash \exists xFx \qquad \pi \qquad \longrightarrow \qquad \left\{\begin{array}{llll} (\alpha) & \mathcal{A} & \vdash Ft & \pi' \\ (\beta) & \mathcal{A} & \vdash \exists xFx & \exists I(\alpha) \end{array}\right.$$

E∃:

$$(\beta)\ \mathcal{A} \vdash \exists xFx \qquad \pi \qquad \longrightarrow \qquad \left\{\begin{array}{llll} (\alpha) & \mathcal{A} & \vdash \neg\forall x\neg Fx & \pi' \\ (\beta) & \mathcal{A} & \vdash \exists xFx & \text{Neg}(\alpha) \end{array}\right.$$

E¬∀:

$$(\gamma)\ \mathcal{A} \vdash \neg\forall xFx \qquad \pi \qquad \longrightarrow \qquad \left\{\begin{array}{llll} (\alpha) & \mathcal{A} & \vdash \neg Ft & \pi' \\ (\beta) & \mathcal{A} & \vdash \exists x\neg Fx & \exists I(\alpha) \\ (\gamma) & \mathcal{A} & \vdash \neg\forall xFx & \text{Neg}(\beta) \end{array}\right.$$

E-Lemma:

$$\begin{array}{lll} (\beta_1)\ \mathcal{A}_1 & \vdash F_1 & \pi_1 \\ & \dots & \\ (\beta_n)\ \mathcal{A}_n & \vdash F_n & \pi_n \end{array} \qquad \longrightarrow \qquad \left\{\begin{array}{llll} (\alpha) & \bigcap\mathcal{A}_i & \vdash G & \pi_0 \\ (\beta_1) & \mathcal{A}_1 & \vdash F_1 & \pi_1' \\ & & \dots & \\ (\beta_n) & \mathcal{A}_n & \vdash F_n & \pi_n' \end{array}\right.$$

B.2 Mixed Rules

M-Cases:

$$\left.\begin{array}{llll}(\alpha) & \mathcal{A} & \vdash F\vee G & \text{Rule } \mathfrak{R}\\ (\zeta) & \mathcal{A} & \vdash H & \pi\end{array}\right\} \rightarrow \left\{\begin{array}{llll}(\alpha) & \mathcal{A} & \vdash F\vee G & \text{Rule } \mathfrak{R}\\ \multicolumn{4}{l}{\text{We consider separately the cases of } (\alpha)}\\ \multicolumn{4}{l}{\underline{\text{Case 1:}}}\\ (\beta) & \mathcal{A}, F & \vdash F & \text{Ass}\\ (\gamma) & \mathcal{A}, F & \vdash H & \pi_1\\ \multicolumn{4}{l}{\underline{\text{Case 2:}}}\\ (\delta) & \mathcal{A}, G & \vdash G & \text{Ass}\\ (\varepsilon) & \mathcal{A}, G & \vdash H & \pi_2\\ \multicolumn{4}{l}{\text{End of cases (1, 2) of } (\alpha)}\\ (\zeta) & \mathcal{A} & \vdash H & \vee E(\alpha,\gamma,\varepsilon)\end{array}\right.$$

M-Divide:

$$\begin{array}{llll}(\zeta) & \mathcal{A} & \vdash F & \pi\end{array} \rightarrow \left\{\begin{array}{llll}(\alpha) & \mathcal{A} & \vdash G\vee\neg G & \text{Axiom}\\ \multicolumn{4}{l}{\text{We consider separately the cases of } (\alpha)}\\ \multicolumn{4}{l}{\underline{\text{Case 1:}}}\\ (\beta) & \mathcal{A}, G & \vdash G & \text{Ass}\\ (\gamma) & \mathcal{A}, G & \vdash F & \pi_1\\ \multicolumn{4}{l}{\underline{\text{Case 2:}}}\\ (\delta) & \mathcal{A}, \neg G & \vdash \neg G & \text{Ass}\\ (\varepsilon) & \mathcal{A}, \neg G & \vdash F & \pi_2\\ \multicolumn{4}{l}{\text{End of cases (1, 2) of } (\alpha)}\\ (\zeta) & \mathcal{A} & \vdash F & \vee E(\alpha,\gamma,\varepsilon)\end{array}\right.$$

M-Choose:

$$\left.\begin{array}{llll}(\alpha) & \mathcal{A} & \vdash \exists xFx & \text{Rule } \mathfrak{R}\\ (\delta) & \mathcal{A} & \vdash G & \pi\end{array}\right\} \rightarrow \left\{\begin{array}{llll}(\alpha) & \mathcal{A} & \vdash \exists xFx & \text{Rule } \mathfrak{R}\\ (\beta) & \mathcal{A}, Fc & \vdash Fc & \text{Ass}\\ (\gamma) & \mathcal{A}, Fc & \vdash G & \pi'\\ (\delta) & \mathcal{A} & \vdash G & \exists E(\alpha,\gamma)\end{array}\right.$$

c may not occur in $\mathcal{A}$, Fx, or G.

M-Inf:

$$\left.\begin{array}{llll}(\alpha) & \mathcal{A} & \vdash F\Rightarrow G & \text{Rule } \mathfrak{R}\\ (\gamma) & \mathcal{A} & \vdash G & \pi\end{array}\right\} \rightarrow \left\{\begin{array}{llll}(\alpha) & \mathcal{A} & \vdash F\Rightarrow G & \text{Rule } \mathfrak{R}\\ (\beta) & \mathcal{A} & \vdash F & \pi'\\ (\gamma) & \mathcal{A} & \vdash G & \text{Tau}(\alpha,\beta)\end{array}\right.$$

M-Unless:

$$\left.\begin{array}{llll}(\alpha) & \mathcal{A} & \vdash F\vee G & \text{Rule } \mathfrak{R}\\ (\zeta) & \mathcal{A} & \vdash G & \pi\end{array}\right\} \rightarrow \left\{\begin{array}{llll}(\alpha) & \mathcal{A} & \vdash F\vee G & \text{Rule } \mathfrak{R}\\ (\beta) & \mathcal{A}, F & \vdash F & \text{Ass}\\ (\gamma) & \mathcal{A}, F & \vdash G & \pi'\\ (\varepsilon) & \mathcal{A} & \vdash F\Rightarrow G & \Rightarrow I(\gamma)\\ (\zeta) & \mathcal{A} & \vdash G & \text{Tau}(\alpha,\varepsilon)\end{array}\right.$$

B.3 Internal Rules

$\underline{\text{I}\bot\text{:}}$

	(α)	$\mathcal{A}$	$\vdash$	F	Rule $\mathfrak{R}$
	(β)	$\mathcal{B}$	$\vdash$	$\neg F$	Rule $\mathfrak{R}'$
$\rightarrow$	(γ)	$\mathcal{A}, \mathcal{B}$	$\vdash$	$\bot$	Contra(α,β)

B.3-1 Analytic Rules

$\underline{\text{I}\wedge\text{left:}}$

	(α)	$\mathcal{A}$	$\vdash$	$F \wedge G$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	F	$\wedge$E(α)

$\underline{\text{I}\wedge\text{right:}}$

	(α)	$\mathcal{A}$	$\vdash$	$F \wedge G$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	G	$\wedge$E(α)

$\underline{\text{I}\neg\neg\text{:}}$

	(α)	$\mathcal{A}$	$\vdash$	$\neg(\neg F)$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	F	$\neg$E(α)

$\underline{\text{I}\forall\text{:}}$

	(α)	$\mathcal{A}$	$\vdash$	$\forall x Fx$	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	Ft	$\forall$E(α)

for an arbitrary term t.

B.3-2 Synthetic Rules

$\underline{\text{Is}\wedge\text{:}}$

	(α)	$\mathcal{A}$	$\vdash$	F	Rule X_1
	(β)	$\mathcal{B}$	$\vdash$	G	Rule X_2
$\rightarrow$	(γ)	$\mathcal{A}, \mathcal{B}$	$\vdash$	$F \wedge G$	Tau(α, β)

$\underline{\text{Is}\vee\text{1:}}$

	(α)	$\mathcal{A}$	$\vdash$	F	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	$F \vee G$	Tau(α)

$\underline{\text{Is}\vee\text{2:}}$

	(α)	$\mathcal{A}$	$\vdash$	G	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	$F \vee G$	Tau(α)

$\underline{\text{Is}\Rightarrow\text{:}}$

	(α)	$\mathcal{A}$	$\vdash$	G	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	$F \Rightarrow G$	Tau(α)

$\underline{\text{Is}\neg\neg\text{:}}$

	(α)	$\mathcal{A}$	$\vdash$	F	Rule $\mathfrak{R}$
$\rightarrow$	(β)	$\mathcal{A}$	$\vdash$	$\neg(\neg F)$	Tau(α)

B.3-3 Converting Rules

I⇒:		(α)	𝒜	⊢	F ⇒ G	Rule ℜ
	→	(β)	𝒜	⊢	¬F ∨ G	Tau(α)
I¬∧:		(α)	𝒜	⊢	¬(F ∧ G)	Rule ℜ
	→	(β)	𝒜	⊢	¬F ∨ ¬G	Tau(α)
I¬∨:		(α)	𝒜	⊢	¬(F ∨ G)	Rule ℜ
	→	(β)	𝒜	⊢	¬F ∧ ¬G	Tau(α)
I¬⇒:		(α)	𝒜	⊢	¬(F ⇒ G)	Rule ℜ
	→	(β)	𝒜	⊢	F ∧ ¬G	Tau(α)
I¬∀:		(α)	𝒜	⊢	¬(∀xFx)	Rule ℜ
	→	(β)	𝒜	⊢	∃x(¬Fx)	Neg(α)
I¬∃:		(α)	𝒜	⊢	¬(∃xFx)	Rule ℜ
	→	(β)	𝒜	⊢	∀x(¬Fx)	Neg(α)
Ic∨1:		(α)	𝒜	⊢	F ∨ G	Rule ℜ
	→	(β)	𝒜	⊢	¬F ⇒ G	Tau(α)
Ic∨2:		(α)	𝒜	⊢	F ∨ G	Rule ℜ
	→	(β)	𝒜	⊢	¬G ⇒ F	Tau(α)
Ic∨∧:		(α)	𝒜	⊢	E∨(F∧G)	Rule ℜ
	→	(β)	𝒜	⊢	(E∨F)∧(E∨G)	Tau(α)
Ic∧∨:		(α)	𝒜	⊢	E∧(F∨G)	Rule ℜ
	→	(β)	𝒜	⊢	(E∧F)∨(E∧G)	Tau(α)

C Example of the Unstructured Linearization Process

In this part of the appendix we want to go through the linearization process of the dependency graph for the "steamroller" example of section 4.6 without the benefit of a structured natural deduction proof. It turns out that it is difficult to extract the internal proof structure purely from the natural deduction proof if it has not been structured during the process of its generation.

For easy reference the original natural deduction proof is repeated below. The numbers in the natural deduction graphs represent the conclusion formulae of the corresponding proof lines.

(1)	1	$\vdash$	$\forall a\,(\forall p\, Eap) \vee (\forall a'\, Ma'a \wedge \exists p'\, Ea'p'\ y{\Rightarrow}\ Eaa')$	Ass
(2)	2	$\vdash$	$\forall c,s,b,f,w\ Mcb \wedge Msb \wedge Mbf \wedge Mfw$	Ass
(3)	3	$\vdash$	$\forall w,f,g\ \neg Ewf \wedge \neg Ewg$	Ass
(4)	4	$\vdash$	$\forall b,c,s\ \neg Ebs \wedge Ebc$	Ass
(5)	5	$\vdash$	$\forall c\,\exists p\, Ecp \wedge \forall s\,\exists p'\, Esp'$	Ass
(6)	2	$\vdash$	Msb	$\forall\wedge E(2)$
(7)	5	$\vdash$	$\exists p'\, Esp'$	$\wedge\forall E(5)$
(8)	4	$\vdash$	$\neg Ebs$	$\forall\wedge E(4)$
(9)	2,4,5	$\vdash$	$\neg\,(Msb \wedge \exists p'\, Esp' \Rightarrow Ebs)$	Tau(6,7,8)
(10)	2,4,5	$\vdash$	$\exists a'\ \neg\,(Ma'b \wedge \exists p'\, Ea'p' \Rightarrow Eba')$	$\exists I(9)$
(11)	2,4,5	$\vdash$	$\neg\ \forall a'\ (Ma'b \wedge \exists p'\, Ea'p' \Rightarrow Eba')$	Neg(10)
(12)	1	$\vdash$	$\forall p\, Ebp \vee (\forall a'\, Ma'b \wedge \exists p'\, Ea'p' \Rightarrow Eba')$	$\forall E(1)$
(13)	1,2,4,5	$\vdash$	$\forall p\, Ebp$	Tau(11,12)
(14)	1,2,4,5	$\vdash$	Ebg	$\forall E(13)$

(15)	2	$\vdash$	Mbf	$\forall\wedge E(2)$
(16)	1,2,4,5	$\vdash$	$\exists p'\ Ebp'$	$\exists I(14)$
(17)	1	$\vdash$	$(\forall p\, Efp) \vee (\forall a'\, Ma'f \wedge \exists p'\, Ea'p' \Rightarrow Efa')$	$\forall E(1)$
(18)	1	$\vdash$	$(\forall p\, Efp) \vee (Mbf \wedge \exists p'\, Ebp' \Rightarrow Efb)$	$\vee\forall E(17)$
(19)	1,2,4,5	$\vdash$	$(\forall p\, Efp) \vee Efb$	Tau(15,16,18)
(20)	1,2,4,5	$\vdash$	$Efg \vee Efb$	$\forall E(19)$

We consider separately the cases of (20)

<u>Case 1:</u>

(21)	1,2,4,5,20	$\vdash$	Efg	Ass
(22)	3	$\vdash$	$\neg Ewg$	$\forall\wedge E(3)$
(23)	1	$\vdash$	$(\forall p\, Ewp) \vee (\forall a'\, Ma'w \wedge \exists p'\, Ea'p' \Rightarrow Ewa')$	$\forall E(1)$
(24)	1	$\vdash$	$Ewg \vee (\forall a'\, Ma'w \wedge \exists p'\, Ea'p' \Rightarrow Ewa')$	$\vee\forall E(23)$
(25)	1,3	$\vdash$	$\forall a'\, Ma'w \wedge \exists p'\, Ea'p' \Rightarrow Ewa'$	Tau(22,24)
(26)	1,3	$\vdash$	$Mfw \wedge \exists p'\, Efp' \Rightarrow Ewf$	$\forall E(25)$
(27)	2	$\vdash$	Mfw	$\forall\wedge E(2)$
(28)	1,2,4,5,20	$\vdash$	$\exists p'\ Efp'$	$\exists I(21)$
(29)	1-5,20	$\vdash$	Ewf	Tau(26,27,28)
(30)	1-5,20	$\vdash$	$Ewf \wedge Efg$	$\wedge I(29)$
(31)	1-5,20	$\vdash$	$\exists a,a',p\ Eaa' \wedge Ea'p$	$\exists I(30)$

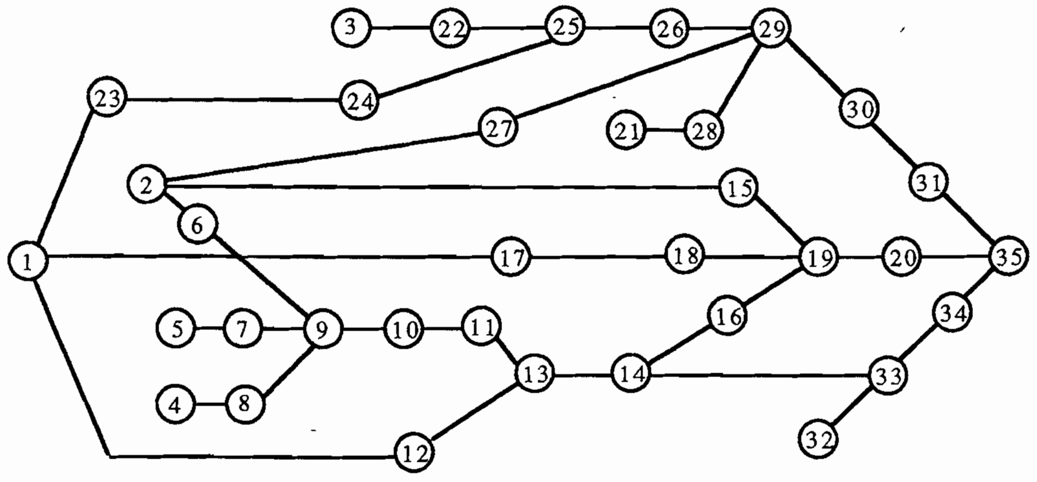
<u>Case 2:</u>

(32)	1,2,4,5,30	$\vdash$	Efb	Ass
(33)	1,2,4,5,30	$\vdash$	$Efb \wedge Ebg$	$\wedge I(14,32)$
(34)	1,2,4,5,30	$\vdash$	$\exists a,a',p\ Eaa' \wedge Ea'p$	$\exists I(33)$

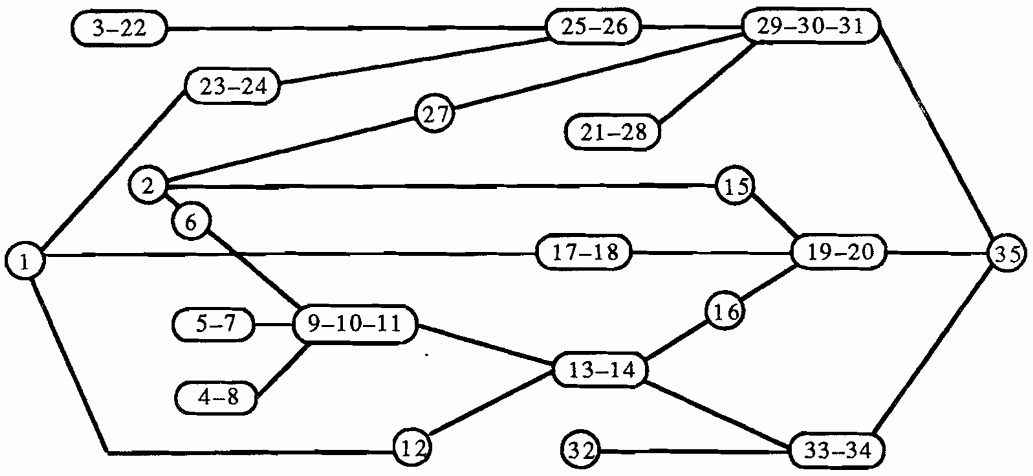
<u>End of Cases (23,34)</u>

(35)	1-5	$\vdash$	$\exists a,a',p\ Eaa' \wedge Ea'p$	$\vee E(20,31,34)$

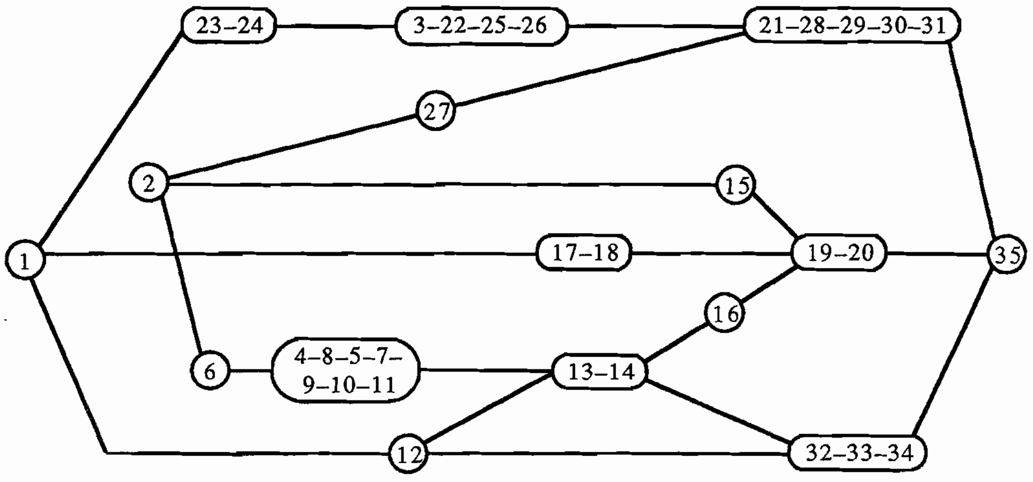
The dependency relation between the proof lines is represented in the initial natural deduction graph shown below.



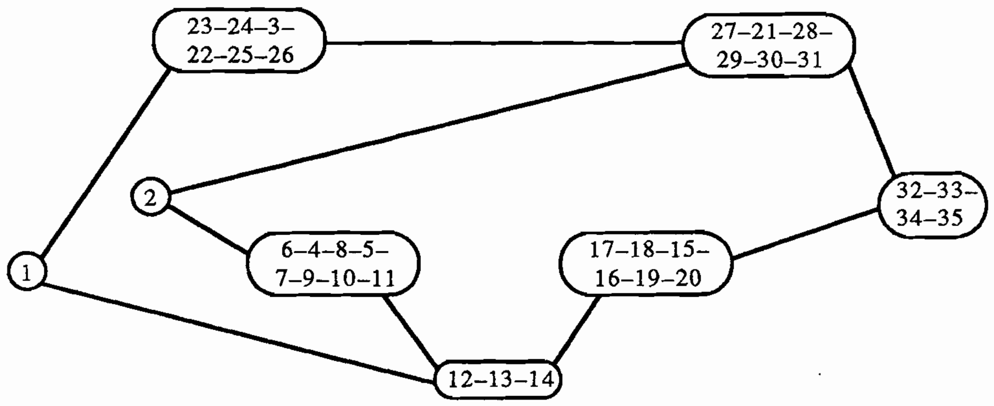
At first rule 1 of definition 4.1-2 is applied to several of the nodes in the natural deduction graph. The resulting graph shown below is still very complicated and rule 1 cannot be applied again.



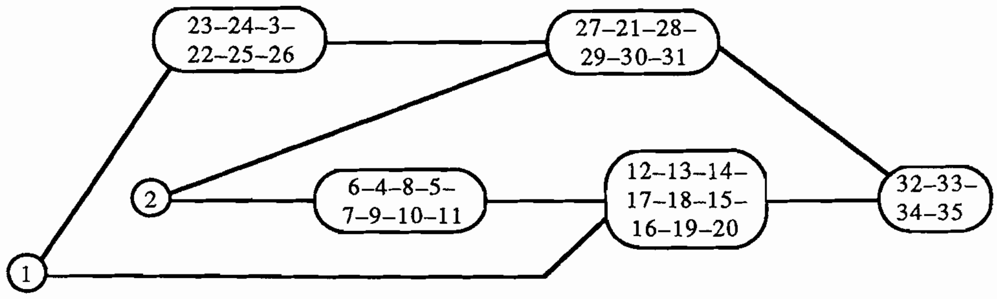
Now rule 2 is used for applicable nodes with least rank, i.e. with the smallest number of immediate predecessors. In this case there are several nodes with no predecessor at all, which correspond to lines where an assumption is introduced. This rule ensures that any new assumptions are mentioned immediately before they are needed for the first time. The next graph results from applying rule 2 to all applicable nodes of rank 0, viz.



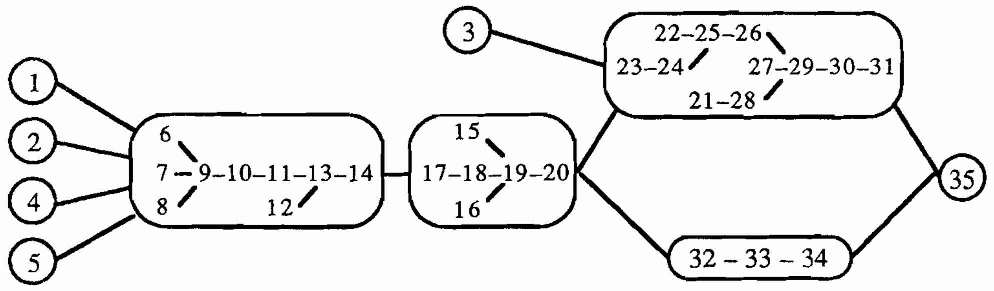
In the next step several nodes of rank 1 can be joined with their unique immediate successors. At this point one must decide the value of maxsize. Remember that a node can only be added to its successor node if its size is smaller than maxsize. There is nothing to guide us here, and we arbitrarily choose five proof lines as this maximal number. Note that this choice prohibits the combination of the two nodes on top of the next graph.



Now only one further combination for a node of rank 2 is possible without violating the maximality condition, and the final generalized natural deduction graph has the following form.



For better comparison with the earlier result (section 4.6), we repeat below the graph which was arrived at by starting with a natural deduction proof that had already been structured during the process of its generation.



While the second graph reflects a meaningful internal structure of the proof, deriving line 14 first (birds eat grain) and then doing a case analysis with respect to line 20 (foxes eat grain or birds), the unstructured version fails to come up with a well-structured result. The subgoals 11 and 26 proposed are merely intermediate results of low-level calculations, which happen to be part of an argument using more than one formula.

Choosing a larger value for maxsize would avoid these meaningless subgoals, but at the same time line 31 would be lost as a distinct subgoal, and possibly line 20 as well. In no way could one come up with the lemma "birds eat grain" of line 14. This example shows that there is more in the structuring of proofs than a mere size argument.

It has become apparent that making visible the internal proof structure during the proof transformation process by exploiting topological properties of the refutation graph is superior to the attempt to find such a structure afterwards.

D Table of Definitions, Examples, and Lemmata

E Table of Symbols

E.1 Signature and Elementary Sets of Symbols

$\mathbb{F}_0$	constant symbols,	a, b, c
$\mathbb{F}_n$, $\mathbb{F}$	n-ary function symbols, function symbols	f, g, h
$\mathbb{V}$	variable symbols	u, v, w, x, y, z
$\mathbb{T}$, $\mathbb{T}_{gr}$	terms, ground terms	s, t
Σ, Σ_{gr}	substitutions; ground substitutions	ρ, σ, τ; γ, δ
$\mathbb{P}_n$, $\mathbb{P}$	n-ary predicate symbols, predicate symbols	P, Q, R
$\mathbb{A}$, $\mathbb{A}_{gr}$	atoms, ground atoms	A, B
$\mathbb{L}$, $\mathbb{L}_{gr}$	literals, ground literals	K, L, M, N
$\mathbb{C}$, $\mathbb{C}_{gr}$	clauses, ground clauses	C, D, E
Φ, Φ_{gr}	formulae, ground formulae	φ, ψ, F, G, H
$\mathbb{F}$	clause graphs	Γ, Δ
$\mathbb{N}$	nodes of a clause graph	K, L, M, N
Π	links of a clause graph	Θ, Λ, Φ, Π

E.2 Objects Denoted by Single Letters

A, B	$\in \mathbb{A}$	atoms
C, D, E	$\in \mathbb{C}$	clauses or clause nodes
φ, ψ, F, G, H	$\in \Phi$	formulae
K, L, M, N	$\in \mathbb{N}$, $\mathbb{L}$	literals or literal nodes
P, Q, R	$\in \mathbb{P}$	predicates
S, T	$\subseteq \mathbb{C}$	sets of clauses
U, V, W, X, Y, Z	$\subseteq \mathbb{V}$	sets of variables
a, b, c, d, e	$\in \mathbb{F}_0$	constant symbols
f, g, h	$\in \mathbb{F} \setminus \mathbb{F}_0$	function symbols
i, j, k, l, m, n		indices
s, t	$\in \mathbb{T}$	terms
u, v, w, x, y, z	$\in \mathbb{V}$	variables

Γ, Δ	$\in \mathbb{\Gamma}$	clause graphs
Θ, Λ, Π	$\in \mathbb{\Pi}$	links of a clause graph
Φ	$\subseteq \mathbb{\Phi}$	set of formulae
Ω		set of formula occurrences
Σ	$\subseteq \mathbb{\Sigma}$	set of substitutions
Ξ, Ψ	$\subseteq \mathbb{\Pi}$	sets of links
γ, δ	$\in \mathbb{\Sigma}_{gr}$	ground substitutions
ε	$\in \mathbb{\Sigma}$	the empty substitution
ζ, η, ϑ		walks in a clause graph
μ	$\in \mathbb{\Sigma}$	matcher
π		proof
ρ, σ, τ	$\in \mathbb{\Sigma}$	substitutions
φ, ψ	$\in \mathbb{\Phi}$	formulae
ω	$\in \mathbb{\Omega}$	formula occurrences

E.3 Combinations of Letters and Special Symbols

[L, M,..., N]	clause (node) containing the literal (nodes) L, M,..., N
L≻≺K	link containing L and K in opposite shores

F Index

Lebenslauf

15. August 1956	geboren in Künzelsau, Hohenlohekreis Eltern: Karl und Ingeborg Lingenfelder, Geschwister: Beate (* 1958), Matthias (* 1959)
1.4.1963 - 30.11.1966	Grundschule Künzelsau
1.12.1966 - 31.7.1967	Gymnasium Künzelsau
1.8.1967 - 31.7.1968	Schloßgymnasium Kirchheim unter Teck
1.8.1968 - 27.5.1975	Ludwig-Uhland-Gymnasium Kirchheim unter Teck
27. Mai 1975	Abitur am Ludwig-Uhland-Gymnasium Kirchheim unter Teck
1.7.1975 - 30.9.1976	Grundwehrdienst
1.10.1976 - 30.9.1979	Studium der Physik und Mathematik an der Universität „Fridericiana“ in Karlsruhe
1.10.1979 - 30.9.1980	Studium der Physik und Mathematik an der University of Maryland, College Park, Md., USA
1.10.1980 - 15.11.1982	Studium der Physik und Mathematik an der Universität „Fridericiana“ in Karlsruhe
1.10.1981 - 28.2.1983	Tutor am Mathematischen Institut I und am Institut für Theorie der Kondensierten Materie (Physik) der Universität Karlsruhe
13. August 1982	Hochzeit mit Andrea Rothfuß
15. November 1982	Abschluß des Studiums mit dem 1. Staatsexamen für das Lehramt an Gymnasien
1.3.1983 - 31.7.1984	wissenschaftlicher Mitarbeiter am Institut für Informatik I der Universität Karlsruhe in der Arbeitsgruppe von Prof. Deussen, Forschungs- und Entwicklungsaufgaben auf dem Gebiet des Programmverifikationssystems SEKI
6. Oktober 1986	Geburt der Tochter Annik
18. Juli 1988	Geburt des Sohnes Maurice
seit 1.8.1984	wissenschaftlicher Mitarbeiter am Fachbereich Informatik der Universität Kaiserslautern in der Arbeitsgruppe von Prof. Siekmann, Mitarbeit in Projekten aus dem Bereich „Automatisches Beweisen“, insbesondere im SFB 314 „Künstliche Intelligenz – wissensbasierte Systeme“