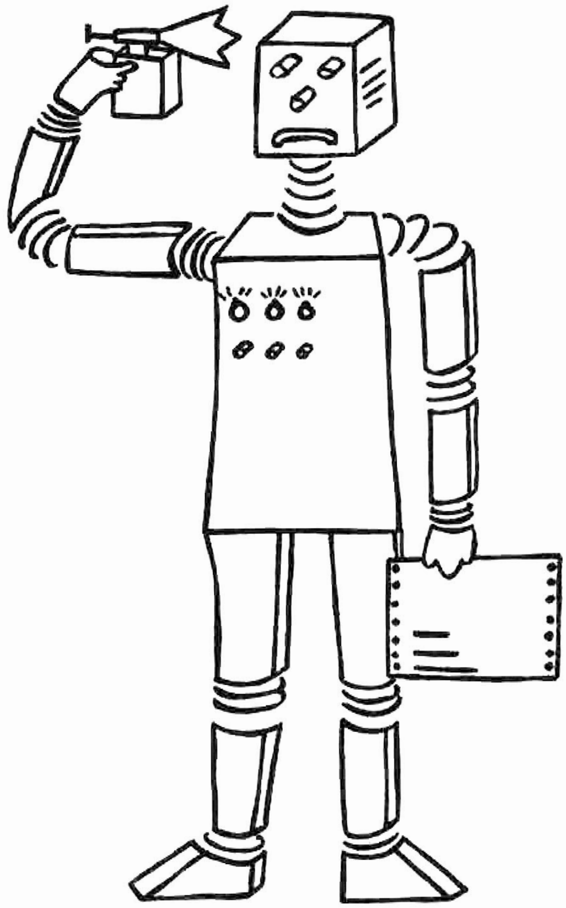
SEKI-REPORT

Artificial
Intelligence
Laboratories

Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern 1, W. Germany



Structuring Computer Generated Proofs

Christoph Lingenfelder

SEKI Report SR-88-19

Structuring Computer Generated Proofs

Christoph Lingenfelder

SEKI-Report SR-88-19 October 1988

This work was supported by the Deutsche Forschungsgemeinschaft (DFG), in project D2 within the Sonderforschungsbereich 314.

Abstract iv

1 Introduction 1

2 Definitions 2

2.1 Signature and Formulae 2

2.2 Clause Graphs 3

2.3 Formula Occurrences 6

2.4 Natural Deduction Proofs 9

3 The Structure of Proofs Inherent in Topological Properties of Refutation Graphs 12

3.1 Trivial Subproofs 12

3.2 Shared Subgraphs as Lemmata 13

3.3 Separating Links that Define Subgoals 15

3.4 Structuring Proofs by Case Analysis 17

4 Linearization of Natural Deduction Proofs 19

4.1 Definitions 19

4.2 Chester's Method 20

4.3 Using Information from Refutation Graphs and/or Transformation Processes 20

4.4 Further Processing 24

5 Summary 26

6 Literature 27

7 Table of Symbols 29

7.1 Signature and Elementary Sets of Symbols 29

7.2 Objects denoted by Single Letters: 29

7.3 Combinations of Letters and Special Symbols 30

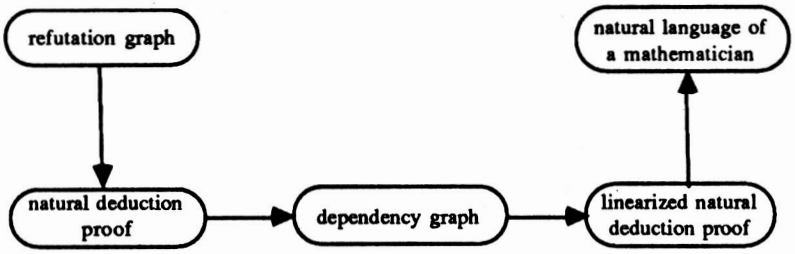
Abstract

One of the main disadvantages of computer generated proofs of mathematical theorems is their complexity and incomprehensibility. 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. During this process topological properties of the refutation graphs can be successfully exploited in order to obtain structured proofs. It is also possible to remove trivial steps from the proof formulation .

1 Introduction

A problem for the acceptance of Automated Deduction Systems has been the difficulty to understand proofs that are automatically generated. 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. The transformation of these 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, thus hiding the interesting ideas in 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:



In a first step the automatically constructed proof is transformed into a natural deduction proof, which is still formal but more human-oriented than most other formats. Then the proof lines are structured in a graph representing their mutual dependencies, which allows grouping of the single lines and a gradual linearization of the natural deduction proof. Finally this natural deduction proof is transformed into an intermediate representation, upon which simplification, structural, and stylistic procedures operate in order to find a "human like" proof style. This representation is then to be transformed into mathematical natural language.

2 Definitions

2.1 Signature and Formulae

This chapter contains the basic definitions of the underlying logic, where literals, clauses, and our notion of unifiability are defined. There are no important differences to the usual way of defining these concepts; similar definitions can for instance be found in [Lo78], and the same definitions are also used in [Li86]. In chapter 7, the symbols used are summarized in a table.

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 denumerable set of *variable symbols*. Then the *term* set $\mathbb{T}$ is the smallest set with

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

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

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 $\sigma: \mathbb{V} \to \mathbb{T}$ 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 report all substitutions will be idempotent. If a substitution maps into $\mathbb{T}_{gr}$, it is called a *ground substitution.*

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: (literals, clauses)

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 of *atoms* A. If A is an atom, then +A and –A are (complementary) *literals*. The set of all literals is $\mathbb{L}$.

A finite multiset of literals is called a *clause*, $\mathbb{C}$ is the set of all clauses. A clause with literals L1,...,Ln is written as [L1 ... Ln].

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.

2.1-4 Definition: (formulae)

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) Quantors	$\forall$	*universal quantifier*
	$\exists$	*existential quantifier*

The set Φ of *formulae* of First Order Predicate Logic is now defined as usual:

(α) $A \subseteq \Phi$

(β) If $A,B \in \Phi$, then $A \wedge B$, $A \vee B$, and $A \Rightarrow B$ are all in Φ.

(γ) If $A \in \Phi$, then $\neg A$, $\forall x\, A$, and $\exists x\, A$ are all in Φ.

(δ) All members of Φ can be described in this way.

$A \Leftrightarrow B$ 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 $\forall x_1 \forall x_2 \ldots \forall x_n A$ and similarly for the existential quantor. If $M=\{M_1,M_2,\ldots,M_n\}$ is a finite set of formulae, we write $\bigwedge \mathcal{M}$ or $\bigwedge_{1 \le i \le n} M_i$ instead of $M_1 \wedge M_2 \wedge \ldots \wedge M_n$ and likewise $\bigvee \mathcal{M}$ or $\bigvee_{1 \le i \le n} M_i$ instead of $M_1 \vee M_2 \vee \ldots \vee M_n$.

Parentheses are 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 quantors $\forall$ and $\exists$ are the weakest. Parentheses may be omitted according to this binding hierarchy, so that the above formula could be written as $\neg A \wedge (B \vee C)$.

2.2 Clause Graphs

In the following subsections (2.2 through 2.4) the formats used for proof representation will be defined. All these definitions concerning the representation of proofs are identical to those used in [Li86], in particular the notions of clause graphs with clause nodes, literal nodes, and links, and of natural deduction proofs defined in chapters 3 and 4.1 of [Li86]. Some further definitions are added in this report, however. They are either stated in this chapter or introduced whenever they are needed for the first time.

2.2-1 Definition: (clause graph)

A *clause graph* is a quadruple $\Gamma = (\mathbb{N},[\mathbb{N}],\pounds, \mathbb{\Pi})$, 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 $[\mathbb{N}]$, which are classes of literal nodes are called the *clause nodes* of Γ. Contrary to the standard definition of a partition, $\emptyset \in [\mathbb{N}]$ is allowed. The clause node of a literal node L is denoted by [L].

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

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

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

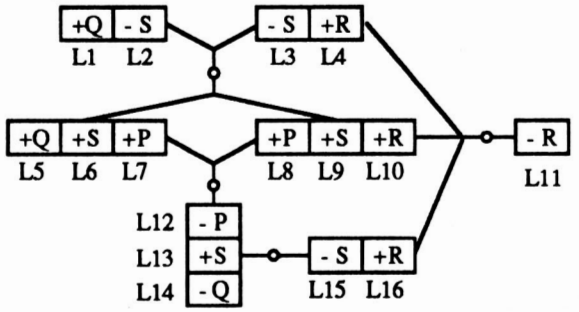
(π_2) There must be at least one positive and one negative literal in a polylink.

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.

These clause graphs, developed in [Ei88] are a generalization of Kowalski's connection graphs, [Ko75], and R. Shostak's refutation graphs, [Sh76].

2.2-2 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 any parts. Since this involves several sets of nodes, one has to define carefully what the resulting graph will be. Adding a polylink Λ to a clause graph Γ means to change Π by adding to it a new set of literal nodes of the previously pure literal nodes; the polylink conditions π_1 and π_2 (cf. 2.2-1) must of course be obeyed and at the same time the relation £ must be extended.

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

Similarly, to remove a polylink Λ from a clause graph Γ means to make its literal nodes pure, i.e. to add them to $\mathbb{N}_p$. Removing a literal node L from Γ is to remove it from its clause node and to change its polylink, unless it is pure. L is simply removed from its shore and, if the shore becomes empty, the whole polylink is removed. Note that, if L was the only literal node in a clause node, then the empty clause remains in the graph.

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

2.2-3 Definition: (altering clause graphs)

Let $\Gamma=(\mathbb{N},[\mathbb{N}],\mathfrak{L},\Pi)$, and let $\Lambda\subseteq \mathbb{N}_p$ fulfil the polylink conditions π_1 and π_2. Let $L\notin\mathbb{N}$, $D\in[\mathbb{N}]$, and $C\cap\mathbb{N}=\emptyset$. Then

$\Gamma+\Lambda$:= $(\mathbb{N},[\mathbb{N}],\mathfrak{L},\Pi\cup\{\Lambda\})$,

$\Gamma,D+L$:= $(\mathbb{N}\cup\{L\},[\mathbb{N}]\setminus\{D\}\cup\{D\cup\{L\}\},\mathfrak{L}',\Pi)$,
where $\mathfrak{L}'$ is an extension of $\mathfrak{L}$ with $\mathfrak{L}'L\in\mathbb{L}$.

$\Gamma+C$:= $(\mathbb{N}\cup C,[\mathbb{N}]\cup\{C\},\mathfrak{L}',\Pi)$,
where $\mathfrak{L}'$ is an extension of $\mathfrak{L}$ with $\mathfrak{L}'L_i\in\mathbb{L}$ for all $L_i\in C$.

Let $\Gamma=(\mathbb{N},[\mathbb{N}],\mathfrak{L},\Pi)$ with $L\in C\in[\mathbb{N}]$ and $L\in\Lambda\in\Pi$. Then

$\Gamma-\Lambda$:= $(\mathbb{N},[\mathbb{N}],\mathfrak{L},\Pi\setminus\{\Lambda\})$

$\Gamma,C-L$:= $(\mathbb{N}\setminus\{L\},[\mathbb{N}]\setminus\{C\}\cup\{C\setminus\{L\}\},\mathfrak{L}|_{\mathbb{N}\setminus\{L\}},\Pi')$

with $\Pi'=\begin{cases}\Pi, & \text{if L was pure}\\ \Pi\setminus\{\Lambda\}, & \text{if } S^+(\Lambda)=\{L\} \text{ or } S^-(\Lambda)=\{L\}\\ \Pi\setminus\{\Lambda\}\cup\{\Lambda\setminus\{L\}\}, & \text{else}\end{cases}$

$\Gamma-C$:= $(\mathbb{N}\setminus C,[\mathbb{N}]\setminus\{C\},\mathfrak{L}|_{\mathbb{N}\setminus C},\Pi'')$,
where Π'' is constructed similar to Π' by removing all literal nodes of C.

2.2-4 Definition: (subgraphs)

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

2.2-5 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 clauses 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. Seeing clauses 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 of links or a link Λ is *separating* Γ, if there exist two clause nodes C and D connected by a walk in Γ, that are no longer connected in $\Gamma-\Lambda$.

2.2-6 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.

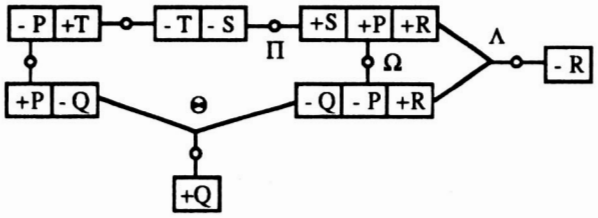
A *component* of a clause graph Γ is a maximal connected subgraph of Γ.

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.

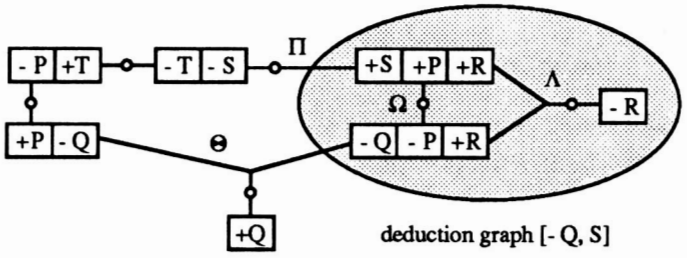
A *deduction graph* is a non-empty, ground, and acyclic clause graph. A *refutation graph* is a deduction graph without pure literal nodes. We sometimes speak of deduction or refutation graphs even if they are not ground, but then the existence of a substitution is required that transforms them into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

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

2.2-7 Example:

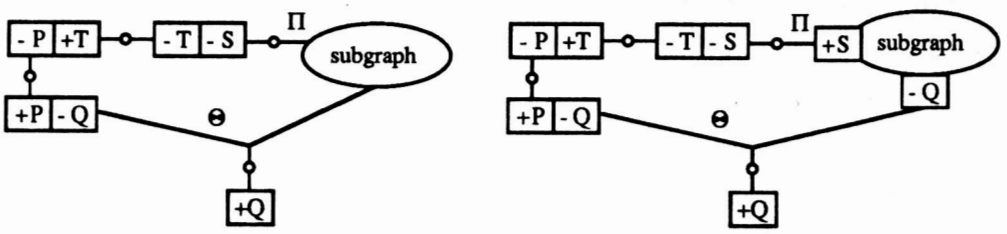


In the above graph all the literal nodes belong to a link. Only Λ and Θ are separating the graph. There is no cycle, since it is not allowed to use a link more than once in a trail, and when a link is entered, it must be exited via the opposite shore. So the graph constitutes an example for a (minimal) refutation graph.



The subgraph marked in the second drawing, consisting of the clauses [+S +P +R], [- R], and [- Q - P +R], as well as the links Λ and Ω, is an example for 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 further 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.



2.3 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 form of a graph, [Sh79], or matrix, [An81] and [Bi81], but usually starts directly with the normalized set of formulae. 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.

2.3-1 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 subformulae* 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 quantor, in the following way:

(α) if $A \in \mathbb{A}$, then $\tau(A)$ is the one node tree labeled with A.
(β) 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.
(γ) 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 itself the root of $\tau(B)$.

2.3-2 Definition: (formula occurrences)

A finite sequence $\omega = \langle \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$ of formulae is called a *formula occurrence* of a formula φ_n within a formula $\varphi_1 \in \Phi$, if

(α) φ_1 is the first element and φ_n the last element of ω.
(β) Every element of ω is an immediate subformula of its predecessor.

It is called an *atom occurrence* within φ_1, if the following additional property holds:

(γ) The last element of ω is an atom.

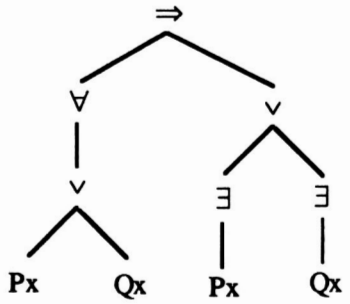
$\Omega(\varphi)$ and $\Omega_a(\varphi)$ denote the sets of formula occurrences and atom occurrences within φ. ω_1 is a *specialized formula occurrence* of ω_2 within a formula φ, $\omega_1 \supset \omega_2$, if both are formula occurrences within a common formula φ, and ω_2 is a subsequence of ω_1.

In most cases there will be no disambiguity 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 don't need to make a distinction between several identical immediate subformulae, as the position in $A \wedge A$, $A \vee A$, or $A \Rightarrow A$ is unimportant.

2.3-3 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.

2.3-4 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, $m \geq 1$.

$$\omega^\varphi = <\varphi, \varphi_1, \ldots, \varphi_n, \phi_1, \ldots, \phi_m> \qquad \omega^\psi = <\psi, \psi_1, \ldots, \psi_k, \phi_1, \ldots, \phi_m>$$

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. $\forall \omega^\psi \in \Omega_a(\psi)\ \exists \omega^\varphi \in \Omega_a(\varphi)$ such that ω^ψ and ω^φ share a tail. An *anchorage* is described as a function $\nu: \Omega_a(\psi) \rightarrow \Omega_a(\varphi)$.

2.3-5 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 ω 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 $\omega_i \supset \omega_j$.

2.3-6 Lemma: (anchored if tails are shared for a basis)

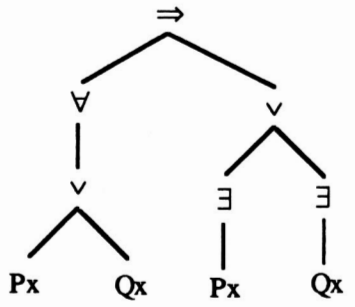
Let φ and ψ be formulae and $\{\omega_1^\psi, \ldots, \omega_n^\psi\} \subseteq \Omega(\psi)$ a basis of ψ. Then ψ is anchored in φ, if $\forall \omega_i^\psi \exists \omega_i^\varphi \in \Omega(\varphi)$ which shares a tail with ω_i^ψ.

In the following we always assume that for a refutation graph Γ proving a formula φ we know a relation $\Delta \subset N \times \Omega_a(\varphi)$ 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 conjunctive normal form and must be maintained throughout the search for the proof.

2.3-3 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.

2.3-4 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, $m \geq 1$.

$$\omega^\varphi = <\varphi, \varphi_1, \ldots, \varphi_n, \phi_1, \ldots, \phi_m> \qquad \omega^\psi = <\psi, \psi_1, \ldots, \psi_k, \phi_1, \ldots, \phi_m>$$

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. $\forall \omega^\psi \in \Omega_a(\psi)\ \exists \omega^\varphi \in \Omega_a(\varphi)$ such that ω^ψ and ω^φ share a tail. An *anchorage* is described as a function ν: $\Omega_a(\psi) \rightarrow \Omega_a(\varphi)$.

2.3-5 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 ω 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 $\omega_i \supset \omega_j$.

2.3-6 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 $\forall \omega_i^\psi\, \exists \omega_i^\varphi \in \Omega(\varphi)$ which shares a tail with ω_i^ψ.

In the following we always assume that for a refutation graph Γ proving a formula φ we know a relation $\Delta \subset \mathbb{N} \times \Omega_a(\varphi)$ 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 conjunctive normal form and must be maintained throughout the search for the proof.

2.4 Natural Deduction Proofs

2.4-1 Definition: (Natural Deduction Proof)

A *proof line* of natural deduction 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 $\mathfrak{A}$, conclusion F and justification Rule X is written $\mathfrak{A} \vdash F$ Rule X. 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 Proof* (NDP) of a formula F, if

(α) F is the conclusion of the last line of S,
(β) the set of assumptions of this last line is empty,
(γ) every line in S is justified by one of the rules below.

Hypothesis Rule (Hyp): $\dfrac{}{\mathfrak{A}, F \vdash F}$

This rule introduces a new assumption. It replaces the axioms of other calculi.

Deduction Rule (Ded): $\dfrac{\mathfrak{A}, F \vdash G}{\mathfrak{A} \vdash F \Rightarrow G}$

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

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

Negation Rule (Neg): $\dfrac{\mathfrak{A} \vdash F}{\mathfrak{A} \vdash G}$

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.

Rule of Indirect Proof (IP): $\dfrac{\mathfrak{A}, \neg F \vdash \bot}{\mathfrak{A} \vdash F}$

Rule of Cases (Cas): $\dfrac{\mathfrak{A} \vdash F \vee G \quad \mathfrak{A}, F \vdash H \quad \mathfrak{A}, G \vdash H}{\mathfrak{A} \vdash H}$

Universal Generalization ($\forall$G): $\dfrac{\mathfrak{A} \vdash G \vee F \vee H}{\mathfrak{A} \vdash G \vee (\forall x\, F) \vee H}$

provided that x is not free in $\mathfrak{A}$, G, or H.

Existential Generalization ($\exists$G): $\dfrac{\mathfrak{A} \vdash G \vee Ft \vee H}{\mathfrak{A} \vdash G \vee (\exists x Fx) \vee H}$

provided that x is not free in Ft.

Universal Instantiation (∀I):

$$\frac{\mathcal{A} \vdash \forall x\ Fx}{\mathcal{A} \vdash Ft}$$

for arbitrary terms t.

Rule of Choice (Sel):

$$\frac{\mathcal{A} \vdash \exists x\ Fx \quad \mathcal{A}, Fc \vdash G}{\mathcal{A} \vdash G}$$

when $c \in F_0$ is not free in $\mathcal{A}$ or G.

Rule of alphabetic Change of bound Variables (αβ):

This rule allows to change the names of bound variables.

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.

2.4-2 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

(a) F is the conclusion of the last line of S,
(b) the last line of S has no assumption,
(c) 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 premises.

This allows lines not correctly justified within the calculus, but it is assumed that these lines are "correct", in the sense that a proof for ($\bigwedge$premises $\Rightarrow$ conclusion) exists. 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 premises and with conclusion F, is called the *trivial GNDP* for F.

In order to find a natural deduction proof for a formula F, for which a proof π has been found, a finite sequence of generalized NDPs can be constructed, whose first element is the trivial GNDP, and whose last element is an NDP for F. The transition between consecutive GNDPs is governed by the set of rules described in [Li86]. As an example, here are three of the rules:

2.4-3 Example: (Transformation Rules)

In the description of the transformation rules, $\mathcal{A}$ 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_1, and π_2 represent proofs of the respective lines. In any case one must make sure that the proofs π_1 and π_2 can be constructed from π or are otherwise known.

E-$\wedge$:

$$(\gamma)\ \mathcal{A} \vdash F\wedge G\ \pi \longrightarrow \begin{cases} (\alpha)\ \mathcal{A} \vdash F\ \pi_1 \\ (\beta)\ \mathcal{A} \vdash G\ \pi_2 \\ (\gamma)\ \mathcal{A} \vdash F\wedge G\ \mathrm{Tau}(\alpha,\beta) \end{cases}$$

M-Cases:

$$\begin{array}{l} (\alpha)\ \mathcal{A} \vdash F\vee G\ \text{Rule}\ \Re \\ (\zeta)\ \mathcal{A} \vdash H\ \pi \end{array} \longrightarrow \begin{cases} (\alpha)\ \mathcal{A} \vdash F\vee G\ \text{Rule}\ \Re \\ \text{We consider separately the cases of } (\alpha) \\ \underline{\text{Case 1:}} \\ (\beta)\ \mathcal{A}, F \vdash F\ \text{Hyp} \\ (\gamma)\ \mathcal{A}, F \vdash H\ \pi_1 \\ \underline{\text{Case 2:}} \\ (\delta)\ \mathcal{A}, G \vdash G\ \text{Hyp} \\ (\varepsilon)\ \mathcal{A}, G \vdash H\ \pi_2 \\ \text{End of cases (1, 2) of } (\alpha) \\ (\zeta)\ \mathcal{A} \vdash H\ \mathrm{Cas}(\alpha,\gamma,\varepsilon) \end{cases}$$

I-$\forall$:

$$(\alpha)\ \mathcal{A} \vdash \forall xFx\ \text{Rule}\ \Re \longrightarrow \begin{cases} (\alpha)\ \mathcal{A} \vdash \forall xFx\ \text{Rule}\ \Re \\ (\beta)\ \mathcal{A} \vdash Fa\ \forall I(\alpha) \end{cases}$$

The selection between different rules that might be applicable is guided by the refutation graphs representing the proofs for the external lines in the GNDPs. The assumptions of such an external line 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 define the notion of polarization of clause nodes, such that clause nodes are positively polarized, if they stem from an axiom, and negatively polarized, if they represent a part of the (negated) theorem.

2.4-4 Definition: (Polarization of Clause Nodes)

Given a refutation graph Γ justifying an external line (α) of a GNDP with assumptions A_i, and conclusion F, and a 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 $<\varphi, A_1\wedge A_2\wedge\ldots\wedge A_n>$.

3 The Structure of Proofs Inherent in Topological Properties of Refutation Graphs

3.1 Trivial Subproofs

In the following subsections 3.2 through 3.4, subgraphs of the orginal 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 is non-trivial. To make a decision we use a heuristic approach taking into account several properties of the refutation graph and the deduction graph.

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 for 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 in case. 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 brought into its final linear form, as explained in chapter 4.

3.2 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]. In addition to the refutation graph, we need a correspondence between the literal nodes of Γ and the atom occurrences within φ. This correspondence is established by a relation $\Delta \subset \mathbb{N} \times \Omega_a(\varphi)$, 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 now be constructed to start a transformation process as described in [Li86]. After each application of a transformation rule a number of tasks need to be performed in order to guarantee a smooth transformation process.

1. Every proof line has a related formula occurrence (anchored in φ) so that the corresponding literal nodes can be obtained using Δ.
2. The refutation graph is changed or divided according to the rule applied; this happens in a way, such that justifications for external proof lines are always minimal refutation graphs.
3. Additional parts of the refutation graph(s) may become positively polarized. 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.

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. In the simplest case the refutation graph proving $F_1 \wedge F_2$ is "cut" through the clause $[\neg F_1 \neg F_2]$, such that the two resulting components are refutation graphs for F_1 and F_2, respectively. In general, however, the two components may have a non-empty intersection, and this is similarly the case for other rules leading to a division of the refutation graph.

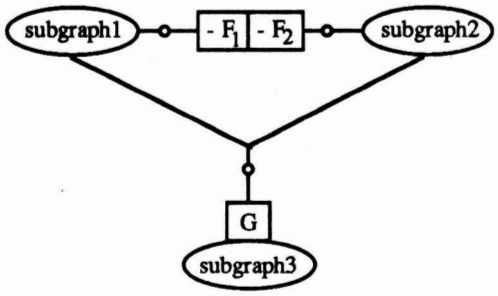
If this intersection is comparatively small, it may easily be duplicated and then used twice in the two subproofs. If it is relatively large, however, it may be sensible to prove a lemma first and then use it in both proofs. In order to formalize such a procedure, a new transformation rule E-Lemma is introduced.

E-Lemma:

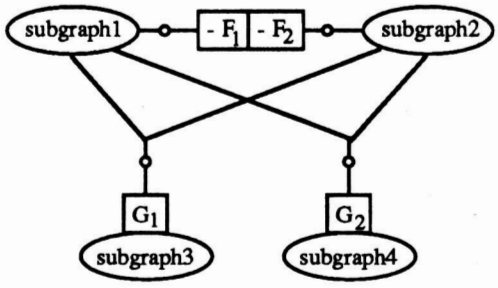
$$\begin{array}{llll} (\beta_1) & \mathcal{A}_1 & \vdash F_1 & \pi_1 \\ & & \cdots & \\ (\beta_n) & \mathcal{A}_n & \vdash F_n & \pi_n \end{array} \longrightarrow \left\{ \begin{array}{llll} (\alpha) & \bigcap \mathcal{A}_i & \vdash G & \pi_0 \\ (\beta_1) & \mathcal{A}_1 & \vdash F_1 & \pi_1' \\ & &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

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 used, 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 3.3.

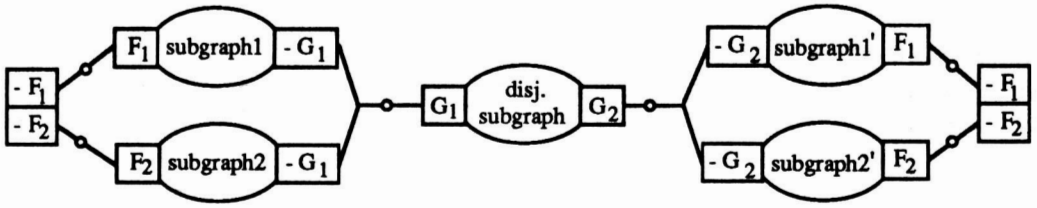
Let us consider for a moment what these shared subgraphs may look like. We always assume that a cut is being made in order to apply E$\wedge$. In the simplest case the lemma consists of just one atom G. Then the graph has the form



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. [Ei88]. 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 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 clauses $[-F_1\ -F_2]$ the graph divides into four components, only two of which are needed. The components containing both clauses $[-F_1]$ and $[-F_2]$ may be discarded. The other two components represent proofs for F_1 and F_2, respectively. In both cases the proof can be done by cases after the lemma $G_1 \vee G_2$ has been introduced.

Already Shostak [Sh79] 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.

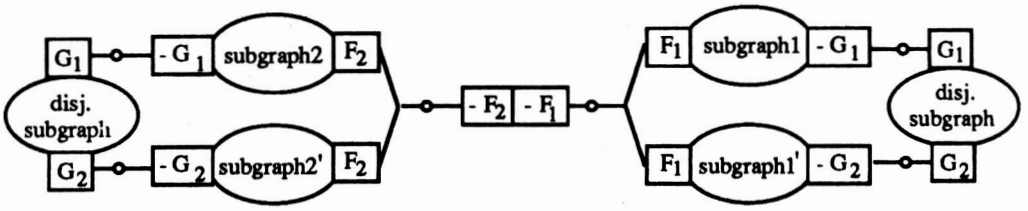
3.2-1 Lemma: (choose clause not to duplicate)

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 $C = [L_1\ L_2\ \ldots\ L_n]$, 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

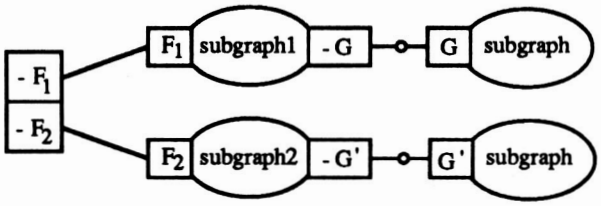
the copies of $[L_i]$ could be combined using a branching link. This holds for all i. We now construct a refutation graph for S by combining all the refutation graphs for S_i so that all the single graphs are only connected via C, which by construction appears only once. □

If one choses 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. This condition may, however, be relaxed, allowing the term arguments of the free (lemma) literal nodes to be different. In this case these differences can be removed by inserting variables, so the lemma becomes a quantified formula used more than once in different instantiations.

Such a lemma corresponds to a resolvent used more than once 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.



3.3 Separating Links that Define Subgoals

In the previous chapter, 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.

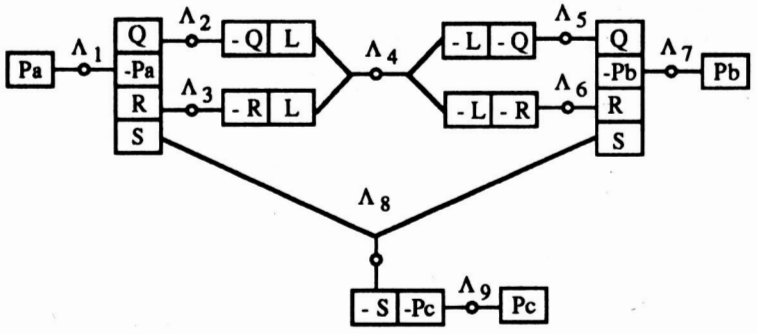
In an automatic proof transformation the difficulty is obviously to find meaningful lemmata. And it is here again that the topological structure of the refutation graph may successfully be exploited. The task is to find parts of the refutation graph that are sufficiently complex in order to justify the 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. When no more automatic external rules, nor E$\vee$, or E$\exists$ can be applied, the search for such links is done using the following algorithm:

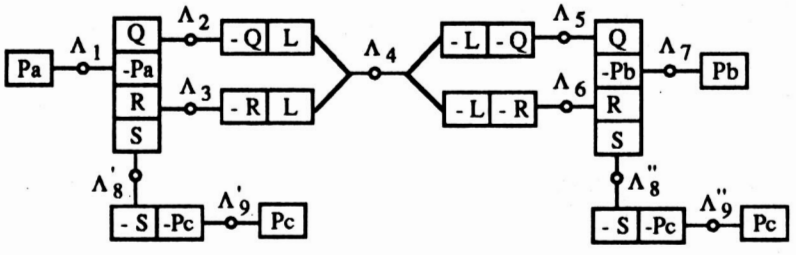
3.3-1 Algorithm: (find separating links to define lemmata)

1. Starting with a non-trivial refutation graph $\Gamma=(\mathbb{N},C,\pounds,\Pi)$, 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 w.r.t. the nesting order $\prec$ 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.
4. Let $\Psi:=\Psi\cup(\Psi_{min}\setminus\Psi_0)$.
5. For each $\Lambda\in\Psi_0$ duplicate the trivial subgraph Γ^{tr}_Λ, if Λ is branching on the non-trivial side. Note that Γ is changed by this operation. Let $\Psi_{min}:=\{\Theta\in\Pi^{ntr}_\Lambda \mid \Theta$ is minimal w.r.t. the nesting order $\prec$ on $\Pi^{ntr}_\Lambda\}$. 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.

3.3-2 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.

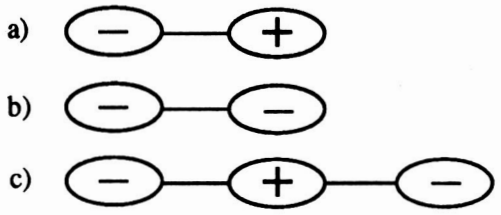
Now the set Ψ of candidate links for the construction of a lemma has been computed, and unless it is empty, we must define an actual lemma by choosing one or several of these links and their related subgraphs. In order to do this, we must try to isolate parts of the graph containing only positively polarized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

3.3-3 Definition: (maximal connected subgraphs)

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

3.3-4 Algorithm: (choose lemma from separating links)

1. Let Ψ be the set of links obtained by algorithm 3.3-1. 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.

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

3.3-5 Example: (choosing the actual lemma)

In the case of the previous example (3.3-2), 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.

3.4 Structuring Proofs by Case Analysis

One of the transformation rules defined in [Li86], that is called M-Cases, leads to a division of the refutation graph. This rule can always be applied, when a disjunction has been derived earlier. An application is undesirable, however, in most 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).

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 copies of the theorem clauses. M-Cases can now be applied, if all the resulting components contain just one of these copies.

4 Linearization of 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 labelled with proof lines of the NDP, incoming edges represent the steps by 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 Hypothesis Rule) have no incoming edges, and the only node with no outgoing edge represents the theorem. In the following subsection this idea, first used by Chester in [Ch75] is defined more exactly.

4.1 Definitions

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 Hyp), which are removed (by Rules Ded, IP, Cas, or Sel) 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 Sel, when

$n_1 =$	$\mathcal{A}$ $\vdash$	$\exists x\, Fx$	Rule $\Re$
$n_2 =$	$\mathcal{A}$,Fc $\vdash$	Fc	Hyp
$n_3 =$	$\mathcal{A}$,Fc $\vdash$	G	Rule $\Re'$
$n_4 =$	$\mathcal{A}$ $\vdash$	G	Sel (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 even 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 $\{n \mid (n,n') \in E\}$.

4.2 Chester's Method

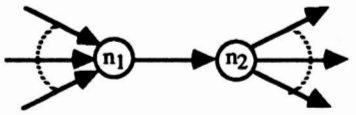
In order to linearize natural deduction proofs, Chester [Ch75] starts with an NDG and constructs a sequence of GNDGs until the graph is (almost) linear. A new element in this sequence is obtained by application of one of the following rules. If both rules can be applied, rule 1 is always preferred.

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 $(n_1,n_2) \in E$, if for all $n \in N$ $(n_1,n) \in E$ implies $n=n_2$, $size(n_1) \leq maxsize$, and n_1 contains no Ded line. 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 Ded, IP, Cas, or Sel), and n_1 is not too large. Otherwise the last line of n_1 is considered a lemma.

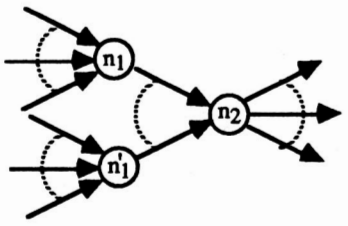
Rule 1:



Rule 2:

$rank(n_1) \leq rank(n_1')$

$size(n_1) \leq maxsize$



4.3 Using Information from Refutation Graphs and/or Transformation Processes

Chester [Ch75] starts his transformation process from a given natural deduction proof having no information of how the proof was found. However, if the NDP was constructed as described in [Li86] and in this report, we already know about lemmata, either from the process of finding the proof or they are defined from the topological structure of the refutation graph as described before in chapter 3.

As an example we start with a refutation graph, and show the graph and the generalized natural deduction proof at the point where the lemma is detected. Finally we give the natural deduction

proof, its natural deduction graph, and its final generalized natural deduction graph. The theorem in the example is a part of the subgroup criterion, cf. [De71]:

> *Let G be a group, and let S be a subset of G. If for all elements x and y in S $x \circ y^{-1}$ is also in S, then for every x in S its inverse x^{-1} is also in S.*

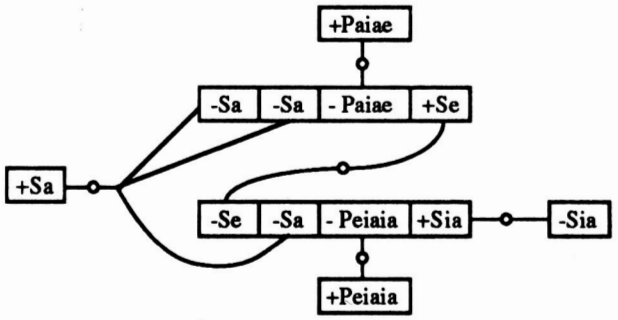
4.3-1 Example: (Natural Deduction Graph)

Theorem: $(\forall u\ Puiue) \wedge (\forall w\ Peww) \wedge (\forall xyz\ Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz) \Rightarrow (\forall x\ Sx \Rightarrow Six)$

Here Pxyz means $x \circ y = z$ in a group, Sx means $x \in S$, a subset of the group, and the function i calculates the inverse of the group elements.

Refutation Graph:

The following refutation graph was automatically generated by our theorem prover MKRP, cf. [EO86]. For the purpose of this report, we assume, that it is just given.

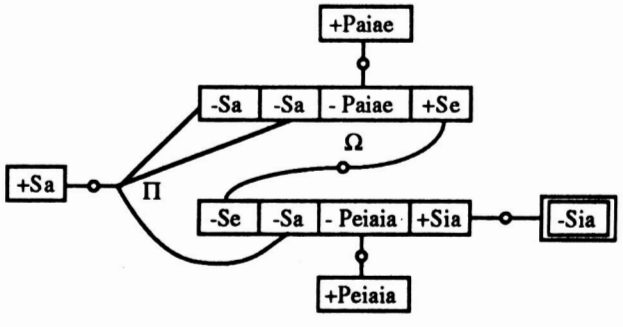


Generalized Natural Deduction Proof after some transformation steps:

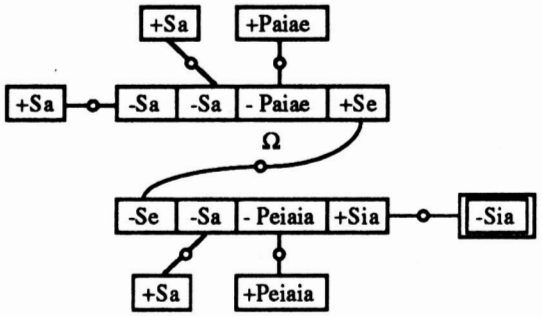
(1)	1	⊢	$(\forall uPuiue) \wedge (\forall wPeww) \wedge (\forall xyz\ Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)$	Hyp
Let a be an arbitrary constant				
(2)	2	⊢	Sa	Hyp
(13)	1, 2	⊢	Sia	Γ_1
(14)	1	⊢	$Sa \Rightarrow Sia$	Ded(13)
(15)	1	⊢	$\forall x\ Sx \Rightarrow Six$	∀G(14)
(16)		⊢	$(\forall uPuiue) \wedge (\forall wPeww) \wedge (\forall xyz\ Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz) \Rightarrow (\forall x\ Sx \Rightarrow Six)$	Ded(15)

Up to here only automatic transformation rules were applied. As this is no longer possible at this stage, we try to detect separating links applying algorithms 3.3-1 and 3.3-4. The actual refutation graph Γ_1 looks like this.

Graph Γ_1



The clause node [–Sia] in double bars is the only negatively polarized one. All the links except Ω are separating a trivial part from the rest of the graph. According to algorithm 3.3-1, Π is broken up and the clause [Sa] is triplicated, viz.



Now Ω is separating, one part is completely positive, and both parts are trivial in itself. Using Davis' definition of obviousness, one has to check that the literal nodes of all clause nodes correspond to different atom occurrences of the original formula. In the complete graph the two clauses [-Sa -Sa -Paiae Se] and [-Se -Sa -Paiae Sia] are different instances of the same formula occurrence in the original fornula. Thus Ω represents a lemma, namely Se. The two graphs below result from cutting the graph at Ω, representing proofs for Se and Sia (using Se) respectively.

Now one proceeds as follows using the new transformation rule E-Lemma:

(1)	1	⊢ (∀uPuiue) ∧ (∀wPeww) ∧ (∀xyz Sx ∧ Sy ∧ Pxiyz ⇒ Sz)	Hyp
Let a be an arbitrary constant			
(2)	2	⊢ Sa	Hyp
(8)	1, 2	⊢ Se	Γ_2
(13)	1, 2	⊢ Sia	Γ_3
(14)	1	⊢ Sa ⇒ Sia	Ded(13)
(15)	1	⊢ ∀x Sx ⇒ Six	∀G(14)
(16)		⊢ (∀uPuiue) ∧ (∀wPeww) ∧ (∀xyz Sx ∧ Sy ∧ Pxiyz ⇒ Sz) ⇒ (∀x Sx ⇒ Six)	Ded(15)

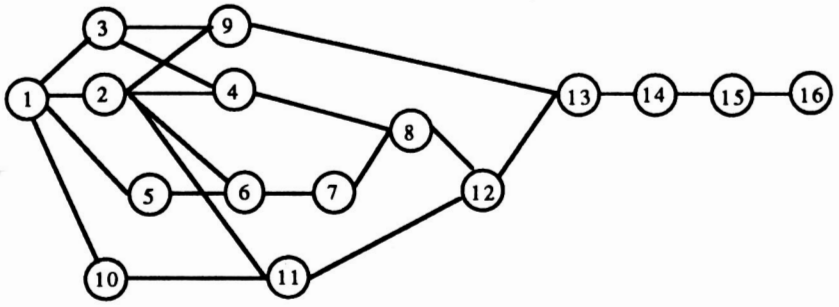
In order to prove Se we first work on Γ_2. This finally leads to the following natural deduction proof.

Natural Deduction Proof:

(1)	1	⊢	(∀uPuiue) ∧ (∀wPeww) ∧ (∀xyz Sx ∧ Sy ∧ Pxiyz ⇒ Sz)	Hyp
Let a be an arbitrary constant				
(2)	2	⊢	Sa	Hyp
(3)	1	⊢	∀xyz Sx ∧ Sy ∧ Pxiyz ⇒ Sz	Tau(1)
(4)	1	⊢	Sa ∧ Paiae ⇒ Se	∀I(3)
(5)	1	⊢	∀uPuiue	Tau(1)
(6)	1	⊢	Paiae	∀I(5)
(7)	1, 2	⊢	Sa ∧ Paiae	Tau(2,6)
(8)	1, 2	⊢	Se	Tau(4,7)
(9)	1	⊢	Se ∧ Sa ∧ Peiaia ⇒ Sia	∀I(3)
(10)	1	⊢	∀wPeww	Tau(1)
(11)	1	⊢	Peiaia	∀I(10)
(12)	1, 2	⊢	Se ∧ Sa ∧ Peiaia	Tau(2,8,11)
(13)	1, 2	⊢	Sia	Tau(9,12)
(14)	1	⊢	Sa ⇒ Sia	Ded(13)
(15)	1	⊢	∀x Sx ⇒ Six	∀G(14)
(16)		⊢	(∀uPuiue) ∧ (∀wPeww) ∧ (∀xyz Sx ∧ Sy ∧ Pxiyz ⇒ Sz) ⇒ (∀x Sx ⇒ Six)	Ded(15)

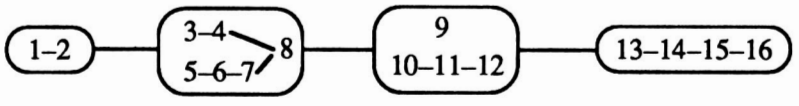
Natural Deduction Graph:

This graph represents the dependency relation between the lines of the natural deduction proof. In line (2) a new (arbitrary) constant is introduced, so all the lines using it 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.

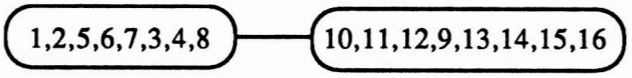


Linearized proof schema:

We can start the process of linearization with a prestructured natural deduction proof, 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 NDG:



One possible linearized form is shown below. If Chester's algorithm is directly applied to the original NDP, there is no guarantee that the proof of Se (8) is executed before any of the lines 9–11 are introduced.



At this point we have to tackle the problem of different users also, i.e. it is here that a user model about the potential mathematical competence of the reader has to come into play.

4.4 Further Processing

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 difficulty in seeing the important steps. One has to distinguish therefore between trivial proof steps and more important steps, which is not straightforward, as the answer depends on the context of the proof as well as on the intended reader. After all, a mathematician will consider a lot of proof steps as trivial, that inexperienced readers might not find easy at all. This raises the question how this distinction can be made automatically.

A first approach will group several steps, especially when only propositional reasoning is involved. But it may also be indicated to combine propositional steps with an instantiation. If this is done for the proof in example 4.3-1, the NDP will be the following:

4.4-1 Example: (Abridged NDP)

(1)	1	$\vdash$ $(\forall u Puiue) \wedge (\forall w Peww) \wedge (\forall xyz\ Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz)$	Hyp
Let a be an arbitrary constant			
(2)	2	$\vdash$ Sa	Hyp
(4)	1	$\vdash$ $Sa \wedge Paiae \Rightarrow Se$	Tau+$\forall$I(1)
(6)	1	$\vdash$ Paiae	Tau+$\forall$I(1)
(8)	1, 2	$\vdash$ Se	Tau(2,4,6)
(9)	1	$\vdash$ $Se \wedge Sa \wedge Peiaia \Rightarrow Sia$	Tau+$\forall$I(1)
(11)	1	$\vdash$ Peiaia	Tau+$\forall$I(1)
(13)	1, 2	$\vdash$ Sia	Tau(2,8,9,11)
(14)	1	$\vdash$ $Sa \Rightarrow Sia$	Ded(13)
(15)	1	$\vdash$ $\forall x\ Sx \Rightarrow Six$	$\forall$G(14)
(16)		$\vdash$ $(\forall u Puiue) \wedge (\forall w Peww) \wedge (\forall xyz\ Sx \wedge Sy \wedge Pxiyz \Rightarrow Sz) \Rightarrow (\forall x\ Sx \Rightarrow Six)$	Ded(15)

This is as much as can be done without knowing the intended reader. If it is known, however, that the proof shall be read by the above mentioned mathematician, or that it will appear in a text book immediately after a proof of Se, then it suffices to state line (8) as a lemma without further proving it.

4.4-2 Example: (Abridged NDP with lemma)

(1)	1	⊢	(∀uPuiue) ∧ (∀wPeww) ∧ (∀xyz Sx ∧ Sy ∧ Pxiyz ⇒ Sz)	Hyp
Let a be an arbitrary constant				
(2)	2	⊢	Sa	Hyp
(8)	1, 2	⊢	Se	Lemma
(9)	1	⊢	Se ∧ Sa ∧ Peiaia ⇒ Sia	Tau+∀I(1)
(11)	1	⊢	Peiaia	Tau+∀I(1)
(13)	1, 2	⊢	Sia	Tau(2,8,9,11)
(14)	1	⊢	Sa ⇒ Sia	Ded(13)
(15)	1	⊢	∀x Sx ⇒ Six	∀G(14)
(16)		⊢	(∀uPuiue) ∧ (∀wPeww) ∧ (∀xyz Sx ∧ Sy ∧ Pxiyz ⇒ Sz) ⇒ (∀x Sx ⇒ Six)	Ded(15)

In order to achieve this sort of reader dependent simplification of the proof it will be necessary however to have a model. The development of user models is a well known research problem in AI, especially in the field of natural language processing and computer interfaces. For an overview see [Ko85]. How such user models might help with an understandable presentation of mathematical proofs has not been elaborated on, yet, this will be the topic of further research.

5 Summary

In this report we have shown, that we can exploit the information of how a proof was found, in order to break it up into smaller lemmata. In particular this may avoid the need to prove a subformula more than once, when it is shared by different branches of the proof. In addition the information implicit in the topological properties of refutation graphs is used to structure 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 do this the algorithm for the transformation of refutation graphs into NDPs had to be extended.

The same information also facilitates the process of linearizing the natural deduction proof. 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 in the end.

The linearized version of the natural deduction proof is then used as a starting point for removing trivial steps from the proof. In general this can only be done with the help of a user model. When all of this is done one can tackle the problem to state this formal proof in mathematical natural language, which is the topic of our current research interest.

6 Literature

[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)

[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

[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 Jürgen Ohlbach
The Markgraf Karl Refutation Procedure
Lecture Notes in Comp. Sci. **230** (CADE 86), 682-683

[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

[Ko85] Alfred Kobsa *Benutzermodellierung in Dialogsystemen*
Informatik Fachberichte **115**, Subreihe KI, Springer 1985

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

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

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

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

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

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

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

7 Table of Symbols

7.1 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	r, s, t; g, d
$\mathbb{P}_n$, $\mathbb{P}$	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	F, G, H
Γ	clause graphs	G, D
$\mathbb{N}$	nodes of a clause graph	K, L, M, N
Π	links of a clause graph	Q, L, F, P

7.2 Objects denoted by Single Letters:

A, B	$\in \mathbb{A}$	atoms
C, D, E	$\in \mathbb{C}$	clauses or clause nodes
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	$\in \Gamma$	clause graphs
Θ, Λ, Π	$\in \Pi$	links of a clause graph
Φ	$\subseteq \Phi$	set of formulae
Ω		set of formula occurrences
Σ	$\subseteq \Sigma$	set of substitutions
Ξ, Ψ	$\subseteq \Pi$	sets of links

γ, δ	$\in \Sigma_{gr}$	ground substitutions
ε	$\in \Sigma$	the empty substitution
ζ, η, ϑ		walks in a clause graph
μ	$\in \Sigma$	matcher
π		proof
ρ, σ, τ	$\in \Sigma$	substitutions
φ, ψ	$\in \Phi$	formulae
ω	$\in \Omega$	formula occurrences

7.3 Combinations of Letters and Special Symbols

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