Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT

Proof Transformation with Built-In Equality Predicate

Christoph Lingenfelder, Axel Präcklein

SEKI Report SR-90-13 (SFB)

Proof Transformation with Built-in Equality Predicate

Christoph Lingenfelder
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern
phone: 49 631 205 3335
email: lingenf@informatik.uni-kl.de

Axel Präcklein
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern
phone: 49 631 205 3344
email: prckln@informatik.uni-kl.de

Abstract

One of the main reasons why computer generated proofs are not widely accepted is often their complexity and incomprehensibility. Especially proofs of mathematical theorems with equations are normally presented in an inadequate and not intuitive way. This is even more of a problem for the presentation of inferences drawn by automated reasoning components in other AI systems. For first order logic, proof transformation procedures have been designed in order to structure proofs and state them in a formalism that is more familiar to human mathematicians. In this report we generalize these approaches, so that proofs involving equational reasoning can also be handled. To this end extended refutation graphs are introduced to represent combined resolution and paramodulation proofs. In the process of transforming these proofs into natural deduction proofs with equality, the inherent structure can also be extracted by exploiting topological properties of refutation graphs.

This research was supported by the Deutsche Forschungsgemeinschaft (DFG), SFB 314

Contents

1 Introduction

With the increasing strength of Automated Deduction Systems the length and complexity of computer generated proofs has reached a degree where they become almost impossible to understand even for the expert. To add to their incomprehensibility, almost every research group uses its own format and style of stating a proof. 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.

But whenever human beings are addressed, 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.

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. There has been some success in structuring computer generated proofs, cf. [Li90], [PN90], or [Hu90], but all of these approaches are restricted to logics without equality.

We feel, however, that this is a severe restriction, as equality is essential for any natural coding of mathematical problems and of AI problems in general. Therefore, in this report, we generalize Lingenfelder's approach, so that proofs involving equational reasoning can also be structured automatically.

In chapter 2 the formal basis for this work is defined, especially the different calculi and proof representations: resolution proofs with paramodulation, refutation graphs with equality clause nodes, and natural deduction proofs. Then the system of natural deduction, that has been invented by Gerhard Gentzen for its simplicity and systematic use of the connectives, is extended by rules to handle the use of equations.

Chapter 3 extends the basic system of proof transformation so that equality reasoning is also covered. This fits well into the transformation approach, if equational reasoning is not dominating. Here equality is seen as a specific theory (in the sense of theory resolution) so that a further generalization to arbitrary theories can be envisaged.

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 the method of case analysis. This constitutes very often the only possibility to use a conditional equation without having to fall back on a proof by contradiction.

2 Proof Formats

In this chapter we will define the logic and introduce all the basic definitions for the logical calculi used in this report. Everything is standard first order predicate logic with a built-in equality predicate, and we need resolution (with paramodulation) 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 $ft_1t_2 \ldots t_n \in \mathbb{T}$.

$\mathbb{V}(t)$ 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. $\mathbb{V}(o)$ 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: $\mathbb{V} \rightarrow \mathbb{T}$ 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 $\mathbb{T} \rightarrow \mathbb{T}$ 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,* 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 σ 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* ($n = 0, 1, \ldots$). There are two special zero-place predicate symbols, *TRUE* (written $\top$) and *FALSE* (written $\bot$), $\top, \bot \in \mathbb{P}_0$, and a binary predicate symbol EQUAL (written $=$), $= \in \mathbb{P}_2$. 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. ♦

An atom $=st$ is also written in infix form, $s = t$. $(A \Leftrightarrow B)$ 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 $\neg A \wedge (B \vee C)$.

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. Similarly, to describe equational transformations, we must be able to refer to subterms within terms or atomic formulae.

2.1-4 Definition: (Subformulae, Subterms)

For any formula F, atom A or term t, we define the sets S(F) of *subformulae* of F and T(A) and T(t) of *subterms* of A and t as follows:

(σ_1) If $F \in \mathbb{A}$, then $S(F)=\{F\}$; if $t \in \mathbb{F}_0 \cup \mathbb{V}$, then $S(t)=\{t\}$.

(σ_2) If F is of the form $G \wedge H$, $G \vee H$, or $G \Rightarrow H$, then $S(F)=\{F\} \cup S(G) \cup S(H)$.
G and H are called *immediate subformulae* of F.

(σ_3) If F is of the form $\neg G$, $\forall x\, G$, or $\exists x\, G$, then $S(F)=\{F\} \cup S(G)$.
In this case G is the only *immediate subformula* of F.

(τ_1) If t is of the form $f t_1 t_2 \ldots t_n$ with $f \in \mathbb{F}_n$ and $t_1, t_2, \ldots, t_n \in \mathbb{T}$, then $T(t)=\bigcup_{i \le n} T(t_i) \cup \{t\}$.

(τ_2) If A is of the form $P t_1 t_2 \ldots t_n$ with $P \in \mathbb{P}_n$ and $t_1, t_2, \ldots, t_n \in \mathbb{T}$, then $T(A)=\bigcup_{i \le n} T(t_i)$.

If s is a subterm of a formula F, we sometimes write F(s) to denote this fact. An occurrence of F(t) in the same context will then represent the formula where s has been replaced by t.

2.1-5 Definition: (Subterm and Subformula Occurrences)

In order to specify subterm occurrences of a given term t (or formula occurrences of a formula F) we use finite sequences $<k_1.k_2. \ldots .k_n>$ of integers. We define the set of *subterm occurrences* $\Omega(t)$ as follows:

(α) $t<>\in\Omega(t)$

(β) $t<i.k_2.\dots.k_n>\in\Omega(t)$ if $t_i<k_2.\dots.k_n>\in\Omega(t_i)$,
where $t=ft_1t_2\dots t_n$, $f\in\mathbb{F}_n$, and $t_1,t_2,\dots,t_n\in\mathbb{T}$

(γ) *subformula occurrences* and subterm occurrences within formulae are defined analogously

$\Omega(F)$ can be viewed as the set of partial paths through the formula tree of F. $\Omega_a(F)\subseteq\Omega(F)$ denotes the set of *atom occurrences* within a formula F.

2.1-6 Example: (Formula Occurrences)

Let $F=(\forall u\, fiuu=e)\wedge(\forall v\, fve=v)\wedge(\forall xy\, Sx\wedge Sy\Rightarrow Sfiyx)\Rightarrow(\forall z\, Sz\Rightarrow Siz)$. For instance $F<1.3.1.1.2>$ denotes the occurrence of Sy within F, and $F<1.1.1.1.1>$ represents the subterm occurrence of iu within F.

Note, that in general identical formulae or terms may appear as different occurrences within a given term or formula.

2.1-7 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 $\square$.

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.

2.1-8 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 $f_ix_1\dots x_{n_i}$, 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).*

2.1-9 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 $\sigma(C \backslash L) \cup \sigma(\rho D \backslash K)$ 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 $C(\neg F)$.

2.1-10 Definition: (Paramodulation Method)

Let C be a clause with a literal L whose atom has a subterm t, and D be a clause with a literal $t' = s$. D can be *paramodulated into* C if, for a renaming ρ such that C and ρD have no variables in common, there exists a unifier σ of $\rho t'$ and t. Then the clause $\sigma(C') \cup \sigma\rho(D \backslash \{t' = s\})$ is called a *paramodulant* of C and D, where C' is constructed from C by replacing t with s in C. ♦

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 connection graph 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].

2.2 Clause Graphs and Refutation Graphs with Equality

2.2-1 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. A clause node consisting of literal nodes L_1 through L_n is denoted by $[L_1 \dots L_n]$.

(c) $\pounds\colon \mathbb{N} \to \mathbb{L}$ 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* 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 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.

2.2-2 Definition: (Clause Graphs and Clause Sets)

A clause graph $\Gamma = (\mathbb{N}, [\mathbb{N}], \pounds, \Pi)$ is said to *represent* a clause set S if every clause node $C \in [\mathbb{N}]$ has the form $[-A(s)\ s \neq t\ A(t)]$ or there is a *parent clause* $C' \in S$

and a ground substitution γ such that the restriction of $\pounds$ to C is a bijection between its literal nodes and the literals of $\gamma C'$. Clause nodes of the form $[-A(s)\ s{\neq}t\ A(t)]$ are called *equality clause nodes*.

2.2-3 Definition: (Deduction and Refutation Graphs)

A *deduction graph* is a non-empty, ground, and acyclic clause graph. A *cycle* is a sequence of clause nodes and links $C_1,\Pi_1,C_2,\ldots C_n,\Pi_n,C_1$, such that all the Π_i are different and they contain literal nodes with opposite sign in their respective neighbour clause nodes. 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.

2.2-4 Definition: (Clause Graph Relation)

For a formula F and a clause graph $\Gamma=(\mathbb{N},[\mathbb{N}],\pounds,\Pi)$ representing C(F) or $C(\neg F)$, 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 when the clause form is constructed from the formula.

It is obvious from this definition that the literal nodes of equality clause nodes are never related to atom occurrences in $\Omega_a(F)$. ♦

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,

Δ: $\Omega_a(F)\rightarrow 2^{\mathbb{N}}$ and $\qquad$ Δ: $\mathbb{N}\rightarrow 2^{\Omega_a(F)}$

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

2.3 Natural Deduction Proofs with Equality

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].

2.3-1 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 $\{\mathcal{A} \vdash \text{F Rule } \mathfrak{R}\}$. 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 2.3-2.

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.

2.3-2 Definition: (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): $$\frac{}{\mathcal{A}, F \vdash F}$$

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): $$\frac{}{\mathcal{A} \vdash F \vee \neg F}$$

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

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

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

AND-Elimination ($\wedge$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 ($\vee$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 ($\neg$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): $$\frac{\mathcal{A} \vdash \forall x\ Fx}{\mathcal{A} \vdash Ft}$$

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

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

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

In addition to Gentzen's original rules for his calculus NK we need rules to handle the equality predicate. So we add the following two rules:

Rule of Reflexivity (Ref): $$\frac{}{\mathcal{A} \vdash t = t}$$

Rule of Equality (=): $$\frac{\mathcal{A} \vdash F(s) \quad \mathcal{B} \vdash s = t}{\mathcal{A}, \mathcal{B} \vdash F(t)} \quad \text{and} \quad \frac{\mathcal{A} \vdash F(t) \quad \mathcal{B} \vdash s = t}{\mathcal{A}, \mathcal{B} \vdash F(s)}$$

2.3-3 Example: (Natural Deduction Proof)

As an example let us prove that $(\forall x\ \neg Px) \Rightarrow \neg(\exists y\ Py)$. Note that no axiom is introduced, so this formula is also valid in intuitionistic logic.

(1)	1	$\vdash$	$\forall x\ \neg Px$	Ass
(2)	2	$\vdash$	Pa	Ass
(3)	3	$\vdash$	$\exists y\ Py$	Ass
(4)	1	$\vdash$	$\neg Pa$	$\forall$E(1)
(5)	1, 2	$\vdash$	$\perp$	Contra(2,4)
(6)	1, 3	$\vdash$	$\perp$	$\exists$E(3,5)
(7)	1	$\vdash$	$\neg\exists y\ Py$	$\neg$I(6)
(8)		$\vdash$	$(\forall x\ \neg Px) \Rightarrow \neg(\exists y\ Py)$	$\Rightarrow$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.

2.3-4 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.

<u>Rule of Propositional Calculus (Tau):</u> $$\frac{\mathcal{A}_1 \vdash F_1 \ \ldots\ \mathcal{A}_n \vdash F_n}{\mathcal{A}_1, \ldots, \mathcal{A}_n \vdash G}$$

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:

2.3-5 Example: (Rule of Propositional Calculus)

In order to prove that $(F \vee G)$ follows from $(\neg F \Rightarrow G)$ 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)	$\mathcal{A}, F$	$\vdash$	$F \vee G$	$\vee I(\gamma)$
Case 2:				
(ε)	$\mathcal{A}, \neg F$	$\vdash$	$\neg F$	Ass
(ζ)	$\mathcal{A}, \neg F$	$\vdash$	G	$\Rightarrow E(\alpha,\varepsilon)$
(η)	$\mathcal{A}, \neg F$	$\vdash$	$F \vee G$	$\vee I(\zeta)$
End of Cases(1,2)				
(ϑ)	$\mathcal{A}$	$\vdash$	$F \vee G$	$\vee E(\beta,\delta,\eta)$

♦

The rule below handles negation in connection with quantifiers. Similar to the rule of propositional calculus it leads to considerably shorter 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.

3 Proof Transformation

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 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 use Generalized Natural Deduction Proofs as defined in [Li90].

3.1 General Procedure

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, 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 $F = (\forall u\ fiuu = e) \land (\forall v\ fve = v) \land (\forall xy\ Sx \land Sy \Rightarrow Sfiyx) \Rightarrow (\forall z\ Sz \Rightarrow Siz)$, with a constant symbol e and function symbols f and 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	$\vdash$	$(\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx)$	Ass
Let a be an arbitrary constant				
(2)	2	$\vdash$	Sa	Ass
(3)	1	$\vdash$	$\forall v\, fve = v$	$\wedge E(1)$
(4)	1, 2	$\vdash$	$fiae = ia$	$\forall I(3)$
(12)	1, 2	$\vdash$	$Sfiae$	π
(13)	1, 2	$\vdash$	Sia	$=(12,4)$
(14)	1	$\vdash$	$Sa \Rightarrow Sia$	$\Rightarrow I(13)$
(15)	1	$\vdash$	$\forall z\, Sz \Rightarrow Siz$	$\forall I(14)$
(16)		$\vdash$	$(\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx)$ $\Rightarrow (\forall z\, Sz \Rightarrow Siz)$	$\Rightarrow I(15)$

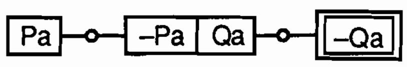
For a proof of π see the refutation graph in example 3.4-1.

In the following we always assume that for a refutation graph Γ proving a formula F we know a clause graph relation $\Delta \subseteq \Omega_a(F) \times \mathbb{N}$ establishing a clear connection between the formula F 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 F into clause form and must be maintained throughout the search for the proof.

3.1-3 Definition: (Polarization of Clause Nodes)

Given a refutation graph Γ justifying an external line α of a GNDP with assumptions F_i, conclusion G, and a clause graph relation Δ, relating all the literal nodes of Γ to atom occurrences in $\Omega_a(H)$ with $H := F_1 \wedge F_2 \wedge \ldots \wedge F_n \Rightarrow G$. Then a clause node is *negatively polarized* if any of its literal nodes is related to an atom occurrence $H<2. \ldots>$. Otherwise the clause node is said to be *positively polarized.* In particular equality clause nodes are positively 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



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 one such transition between two consecutive GNDPs is shown.

3.1-4 Example: (Proof Transformation)

At a certain point during the proof transformation process the following GNDP has been arrived at:

$$\begin{array}{lllr}
(1) & 1 & \vdash & (\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx) & \text{Ass} \\
(15) & 1 & \vdash & \forall z\, Sz \Rightarrow Siz & \pi \\
(16) & & \vdash & (\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx) & \\
 & & & \Rightarrow (\forall z\, Sz \Rightarrow Siz) & \Rightarrow\text{I}(15)
\end{array}$$

Now in order to prove the universally quantified formula in (15) the formula is derived for a (new) arbitrary constant. Then it is possible to generalize the result. This leads to the next GNDP.

$$\begin{array}{lllr}
(1) & 1 & \vdash & (\forall u\, fiuu = e) \wedge (\forall v\, fev = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx) & \text{Ass} \\
\multicolumn{4}{l}{\text{Let a be an arbitrary constant}} \\
(14) & 1 & \vdash & Sa \Rightarrow Sia & \pi' \\
(15) & 1 & \vdash & \forall z\, Sz \Rightarrow Siz & \forall\text{I}(14) \\
(16) & & \vdash & (\forall u\, fiuu = e) \wedge (\forall v\, fev = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx) & \\
 & & & \Rightarrow (\forall z\, Sz \Rightarrow Siz) & \Rightarrow\text{I}(15)
\end{array}$$

♦

In the following section we shall give a formal account of some of these transition rules. In their description, $\mathcal{A}$ is a list of assumption formulae, capital letters indicate single formulae, α, β, γ, ... 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 Transformation Rules

The transformation rules are to be read as follows: the lines on the left hand side of the arrow ($\rightarrow$) are replaced by those on the right hand side in the next generalized NDP of the sequence. Some of the rules simply add a new internal line, they are also marked with an arrow and written below their parent lines.

The rules can be divided into three classes. Internal rules introduce new internal lines without any relation to current external lines, this corresponds to forward reasoning. External rules try to reduce a current external line, this realizes backward reasoning. Mixed rules depend on both, external and internal lines, reducing an external line in the light of previously derived formulae.

E$\wedge$:

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

E$\Rightarrow$:

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

E$\forall$:

$$(\beta)\ \mathcal{A} \vdash \forall xFx \quad \pi \quad \longrightarrow \quad \left\{\begin{array}{lllr} \multicolumn{4}{l}{\text{Let c be an arbitrary constant}} \\ (\alpha) & \mathcal{A} & \vdash Fc & \pi' \\ (\beta) & \mathcal{A} & \vdash \forall xFx & \forall I(\alpha) \end{array}\right.$$

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

M-Cases:

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

I$\wedge$left:

$$\begin{array}{llllr} & (\alpha) & \mathcal{A} & \vdash F\wedge G & \text{Rule } \mathfrak{R} \\ \longrightarrow & (\beta) & \mathcal{A} & \vdash F & \wedge E(\alpha) \end{array}$$

I$\forall$:

$$\begin{array}{llllr} & (\alpha) & \mathcal{A} & \vdash \forall xFx & \text{Rule } \mathfrak{R} \\ \longrightarrow & (\beta) & \mathcal{A} & \vdash Ft & \forall E(\alpha) \end{array}$$

for an arbitrary term t.

E$\exists$-constructive:

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

M-Inf:

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

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

I-Tau:

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

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

As we are mainly concerned with the effect of equality reasoning in a proof, we will now state the rules handling the application of an equation.

E-=⊥:

$$\left.\begin{array}{llll} (\gamma) & \mathcal{A} & \vdash \bot & \pi \end{array}\right\} \longrightarrow \left\{\begin{array}{lllr} (\alpha) & & \vdash s=s & \text{Ref} \\ (\beta) & \mathcal{A} & \vdash s\neq s & \pi' \\ (\gamma) & \mathcal{A} & \vdash \bot & \text{Contra} \end{array}\right.$$

E-=:

$$\left.\begin{array}{llll} (\alpha) & \mathcal{A} & \vdash s=t & \text{Rule } \mathfrak{R} \\ (\gamma) & \mathcal{A} & \vdash F(t) & \pi \end{array}\right\} \longrightarrow \left\{\begin{array}{lllr} (\alpha) & \mathcal{A} & \vdash s=t & \text{Rule } \mathfrak{R} \\ (\beta) & \mathcal{A} & \vdash F(s) & \pi' \\ (\gamma) & \mathcal{A} & \vdash F(t) & =(\beta,\alpha) \end{array}\right.$$

As we are mainly concerned with the effect of equality reasoning in a proof, we will now state the rules handling the application of an equation.

3.3 A Proof Transformation System with Equality

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 can be constructed 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. Some of the rules (called "automatic") are always useful and can automatically be applied by the system. For the others (called "user guided") the user must make a decision or provide more information. So 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.

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

3.3-1 Algorithm: (Basic Proof Transformation)

1. Start with GNDP = $\{\emptyset \vdash F\}$.
2. Apply "automatic" external rules as long as possible.[1]
3. Decide whether to apply any of the "user guided" external rules. If so, do it , then go to 2.
4. Now apply mixed rules until no longer possible. 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. Choose any number of the internal rules to be applied. Then go to 4.

The transformation rules are selected 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 definition 3.1-3.

In [Li90] it is shown how a proof represented as a refutation graph without equality 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.

After an application of rule E-= replacing a goal formula F(t) by F(s) a refutation graph for F(s) can be constructed by removing the equality clause node [$-$P(s) s$\neq$t P(t)] and the equation clause node [s$=$t] unless used more than once.

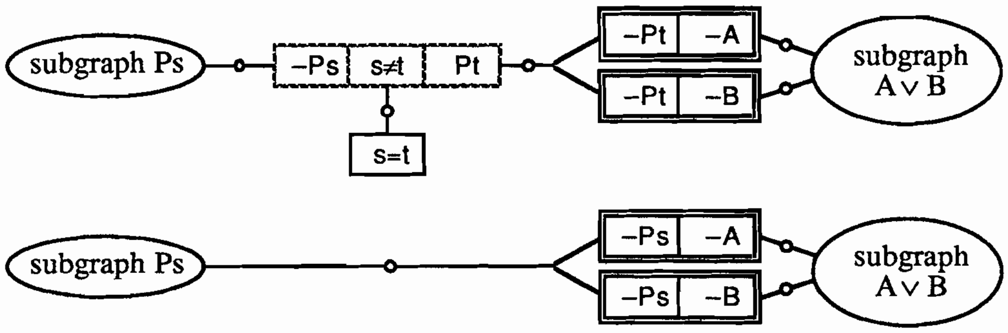
[1] A formula is not further divided if it is "integral", i.e. it can be derived without using its subformulae. For a definition of the notion of integrality see [Li90].

Then the literal node –P(s) is substituted for all the literal nodes linked to P(t) from the equality clause node. In doing so –P(s) remains linked as before and –P(s), which was previously not related to an atom occurrence, inherits the relation from –P(t) and also becomes negatively polarized.

Similarly, when E-=⊥ is applied, the polarization of the clause node [s=s] is changed from positive to negative. Note that the refutation graph has no negatively polarized part when a proof by contradiction is represented. The structure of the graph remains unchanged.

3.3-2 Example: (Graph Update for E-=)

In this example we show a refutation graph for $F(t) = Pt \wedge (A \vee B)$ and the resulting graph proving $Ps \wedge (A \vee B)$.



3.4 Example

As an example of the proof transformation procedure, we use again a part of the subgroup criterion. The formula to prove is

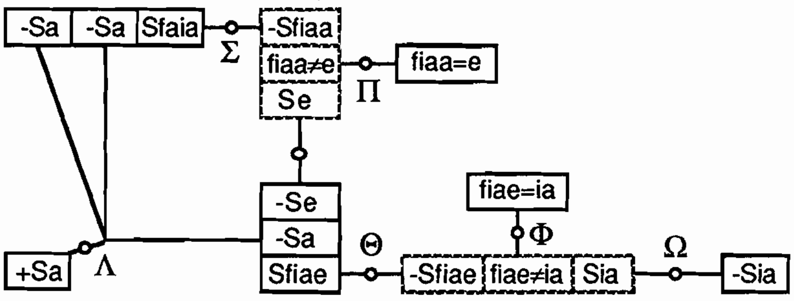
$$F = (\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx) \Rightarrow (\forall z\, Sz \Rightarrow Siz).$$

In order to develop a natural deduction proof of this formula we start with the following information:

- a refutation graph Γ proving F,
- the set of equality clause nodes in Γ,
- a clause graph relation $\Delta \subseteq \Omega_a(F)$, and
- 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 $\forall xyz\, Sx \wedge Sy \Rightarrow Sfiyx$, which is instantiated twice, are renamed to x_1, y_1, and x_2, y_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 shown; theory clause nodes are indicated by dashed lines.

3.4-1 Example: (Refutation Graph)



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

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

(16)		$\vdash$	$(\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx) \Rightarrow (\forall z\, Sz \Rightarrow Siz)$	Γ

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

(1)	1	$\vdash$	$(\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx)$	Ass
(15)	1	$\vdash$	$\forall z\, Sz \Rightarrow Siz$	Γ
(16)		$\vdash$	$(\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx) \Rightarrow (\forall z\, Sz \Rightarrow Siz)$	⇒I(15)

The new external line is justified by the same refutation graph Γ, Δ and γ have also not changed.

With some further steps not involving equality, the following GNDP is constructed:

(1)	1	$\vdash$	$(\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx)$	Ass

Let a be an arbitrary constant

(2)	2	$\vdash$	Sa	Ass
(13)	1, 2	$\vdash$	Sia	π
(14)	1	$\vdash$	$Sa \Rightarrow Sia$	⇒I(13)
(15)	1	$\vdash$	$\forall z\, Sz \Rightarrow Siz$	∀I(14)
(16)		$\vdash$	$(\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx) \Rightarrow (\forall z\, Sz \Rightarrow Siz)$	⇒I(15)

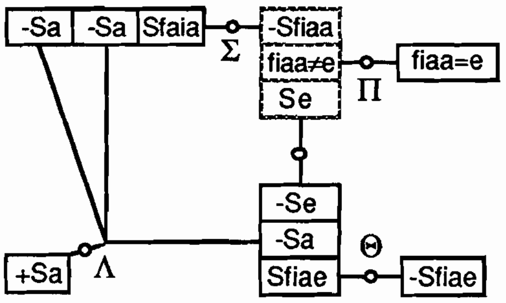
At this point the formula to prove, Sia, cannot further be processed by external rules. A consultation of the refutation graph, which has not changed yet, tells us that the literal node corresponding to Sia is only connected to a theory clause node. Therefore Sia must be derived using the rule of equality. As a preliminary operation we must isolate the equation $feia = ia$, using internal rules, then E-= can be applied. Now the GNDP takes on the form shown below:

(1)	1	$\vdash$	$(\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx)$	Ass

Let a be an arbitrary constant

(2)	2	$\vdash$	Sa	Ass
(3)	1	$\vdash$	$\forall v\, fve = v$	$\wedge$E(1)
(4)	1, 2	$\vdash$	$fiae = ia$	$\forall$I(3)
(12)	1, 2	$\vdash$	$Sfiae$	π'
(13)	1, 2	$\vdash$	Sia	=(12,4)
(14)	1	$\vdash$	$Sa \Rightarrow Sia$	$\Rightarrow$I(13)
(15)	1	$\vdash$	$\forall z\, Sz \Rightarrow Siz$	$\forall$I(14)
(16)		$\vdash$	$(\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx) \Rightarrow (\forall z\, Sz \Rightarrow Siz)$	$\Rightarrow$I(15)

The refutation graph π has now changed to the graph π':



Now the transformation process continues as in the first order case, with just one more application of an equality rule to produce the final natural deduction proof:

(1)	1	$\vdash$	$(\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx)$	Ass

Let a be an arbitrary constant

(2)	2	$\vdash$	Sa	Ass
(3)	1	$\vdash$	$\forall v\, fve = v$	$\wedge$E(1)
(4)	1, 2	$\vdash$	$fiae = ia$	$\forall$I(3)
(5)	1	$\vdash$	$\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx$	$\wedge$E(1)
(6)	1	$\vdash$	$Se \wedge Sa \Rightarrow Sfiae$	$\forall$I(5)
(7)	1	$\vdash$	$\forall u\, fiuu = e$	$\wedge$E(1)
(8)	1	$\vdash$	$fiaa = e$	$\forall$I(7)

(9)	1	$\vdash$	$Sa \wedge Sa \Rightarrow Sfiaa$	$\forall I(5)$
(10)	1	$\vdash$	$Sfiaa$	$\Rightarrow E(2,9)$
(11)	1	$\vdash$	Se	$=(10,8)$
(12)	1, 2	$\vdash$	$Sfiae$	$\Rightarrow E(11,2,6)$
(13)	1, 2	$\vdash$	Sia	$=(12,4)$
(14)	1	$\vdash$	$Sa \Rightarrow Sia$	$\Rightarrow I(13)$
(15)	1	$\vdash$	$\forall z\, Sz \Rightarrow Siz$	$\forall I(14)$
(16)		$\vdash$	$(\forall u\, fiuu = e) \wedge (\forall v\, fve = v) \wedge (\forall xy\, Sx \wedge Sy \Rightarrow Sfiyx)$ $\Rightarrow (\forall z\, Sz \Rightarrow Siz)$	$\Rightarrow I(15)$

4 Structuring the Proof

4.1 General Procedure

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. 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 section 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 [Li90]. In the simplest case the refutation graph proving $F \wedge G$ is "cut" through the clause $[\neg F \neg G]$, 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[1], 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, the transformation rule E-Lemma is introduced.

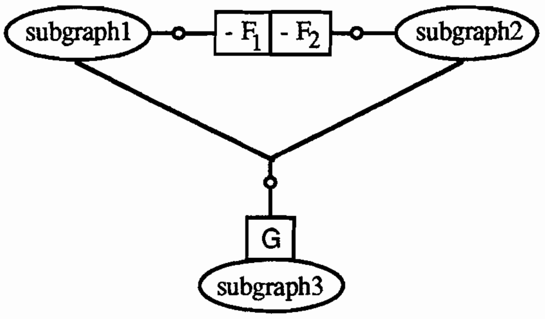
[1] A lemma only makes sense, when the deduction graph in question is complex enough to warrant a separate proof. Otherwise it may be better to repeat a trivial argument instead of using a lemma. It is of course not straightforward to decide which deduction graph (or lemma) is non-trivial. To make a decision we use a heuristic approach taking into account several properties of the refutation graph and its subgraph, the deduction graph proving the lemma. As in [Li90] our method is based on Davis' definition of trivial proofs [Da81].

<u>E-Lemma:</u>

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

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



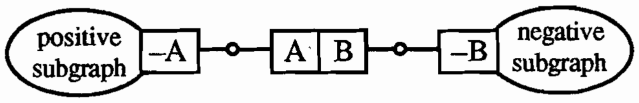
Up to now, the main incentive for the introduction of a lemma was to avoid an unnecessary duplication of parts of the proof. But this is not the only reason why mathematicians use lemmata. In many cases they are used purely to structure the proof, so that the idea behind a proof becomes better visible.

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. If paramodulation steps are represented using equality clause nodes in the refutation graph the search algorithm in [Li90] can be used unchanged.

M-Cases, one of the transformation rules defined in section 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.

It is advantageous to apply M-Cases, 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.

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 in [Li90].

The most important case for the rule M-Cases in pure first order logic comes up, when an existentially quantified formula cannot be proven constructively. With built-in equalíty there is one additional reason for a case analysis. As the natural deduction calculus only allows the application of unit equations, special considerations are needed for conditional equations. One solution of this problem could be the division of the proof into cases such that the equation is assumed to hold in one of them.

4.2 Special Considerations Induced by Equality Clause Nodes

As mentioned above paramodulation steps of the computer generated proof are represented in the refutation graph using special "equality" clause nodes. For example the combination of a paramodulation step Pa to Pb via a=b and the resolution step between Pb and some literal –Pb is simulated as a sequence of three resolution steps of the equality clause node [–Pa a≠b Pb] with the unit clauses [Pa], [–Pb], and [a=b]. The special clause node denotes the trivial fact that $Pa \wedge a{=}b \Rightarrow Pb$. The symmetry of the equality predicate is incorporated into the unification algorithm and hence this additional property must not be considered in the graph. In the natural deduction calculus this fact is reflected by the existence of two symmetric rules for the application of an equation. Alternatively one might have chosen a rule axiomatizing the symmetry explicitely, viz.

$$\frac{\mathcal{A} \vdash s=t}{\mathcal{A} \vdash t=s}$$

But this does not comply with intuitive mathematical reasoning, where equation are rarely oriented and therefore such a symmetry rule never needs to be explicitly used.

Conditional equations, that is, equational literals in non unit clauses, need no special handling, because the only difference is that the negated equation in the equality clause node is connected to a non unit clause node and therefore to a whole refutation graph. However, the formulation of the proof can be more difficult because the equation is not necessarily true. One can either prove the equality as a lemma or divide the proof into cases such that the equality holds in one of the cases.

The decision between these possibilities depends on general considerations, as for example the complexity of resulting lemmata or the position of negatively polarized clause nodes in the graph. Yet there is one heuristic depending on equality. Case analysis is most profitable if the condition for an equation is itself an equation used for paramodulation. Then both obstructing conditions are removed in parallel.

Usually mathematicians employ case analysis only when the disjunction is an axiom or has been previously derived. Equality clause nodes, however, represent implications and therefore are unattractive for this purpose. But if –Pa or a≠b is first derived from the contrapositive of $Pa \wedge a{=}b \Rightarrow Pb$, a case analysis may then be the best choice.

Often several equations are successively applied to a formula leading to chains of equality clause nodes. If any of their connecting links are separating, and therefore candidates for lemmata, only the links joining the chain to the rest of the graph should be selected. Otherwise the equality argument would be torn asunder.

A more syntactical criterion is the distinction between completion and rewriting steps, which can be made if the underlying paramodulation rule discriminates these steps according to the Knuth-Bendix algorithm. Completion steps are more important and substantial while rewriting steps can usually be considered a calculation rather than a proof.

The structuring procedure can be generalized to theory resolution with arbitrary theories. A resolution step between two literals that are complementary in the given theory is represented with a clause node containing the residue and a syntactically complementary literal for each resolution literal.

Another possibility to represent such theory resolution steps would be to use pointers from the links to the residue literals. But this method does not fit similarly well into our framework of refutation graphs as the simulation via resolutions. In addition the sequence of resolutions can be simply transformed into a Gentzen style proof.

It is clear that this method can only handle proofs with a relatively small number of paramodulation steps. Otherwise a large number of equality clause nodes would obscure the structure of the proof. This is especially the case when paramodulation steps are performed into other equations. Therefore purely or even substantially equational proofs need special considerations due to their inherent internal structure.

4.3 Example

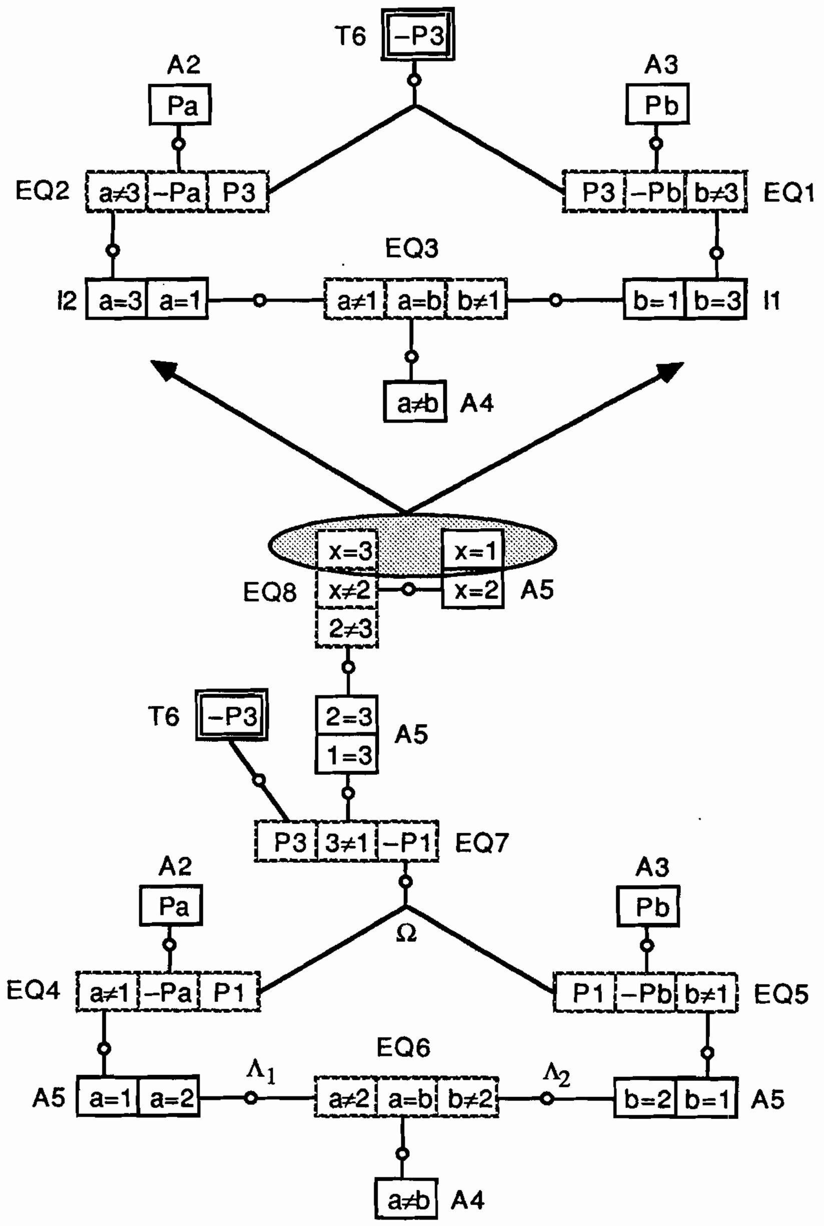
As an example we chose one of Pelletier's problems [Pe86], which can be considered one of the simpler standard examples in equality theorem proving. In this problem there are exactly two objects, and it must be shown that a property P holds for all x if it holds for two different constants. In first order notation with equality this is represented by the following formula:

$$\exists x,y\ \forall z\,(z=x \lor z=y) \land (a \neq b) \land Pa \land Pb \Rightarrow \forall w\ Pw$$

An automatically generated resolution and paramodulation proof of this formula is shown below. x, y, as well as w become Skolem constants in clause normal form, and therefore also in the refutation graph. They are named 1, 2, and 3 in the sequel.

```
Axioms:        Ex x,y (All z z = z Or z = y)
               P(a) And P(b)
               Not a = b

Theorems:      All x P(x)

Set of Axiom Clauses Resulting from Normalization:

               * A1:    All x:Any + =(x x)
               * A2:    + P(a)
               * A3:    + P(b)
               * A4:    - =(a b)
               * A5:    All x:Any + =(x 1)  + =(x 2)

Set of Theorem Clauses Resulting from Normalization:

               * T6:    - P(3)

Refutation:

A5,2 & A2,1    →        * P1:   + P(2)  + =(a 1)
A5,2 & A3,1    →        * P3:   + P(2)  + =(b 1)
A5,2 & T6,1    →        * P4:   - P(2)  + =(3 1)
P1,2 & A4,1    →        * P5:   - =(1 b)  + P(2)
P5,1 & P3,2    →        * R6:   + P(2)  + P(2)
R6 1=2         →        * D7:   + P(2)
P4,1 & D7,1    →        * R8:   + =(3 1)
A5,1 & R8      →        * Rw9:  All x:Any + =(x 3)  + =(x 2)
Rw9,2 & A4,1   →        * P11:  - =(2 b)  + =(a 3)
P11,2 & A2,1   →        * P12:  + P(3)  - =(2 b)
P12,1 & T6,1   →        * R13:  - =(2 b)
Rw9,2 & R13,1  →        * R15:  + =(b 3)
A3,1 & R15     →        * Rw16: + P(3)
Rw16,1 & T6,1  →        * R19:  □
```

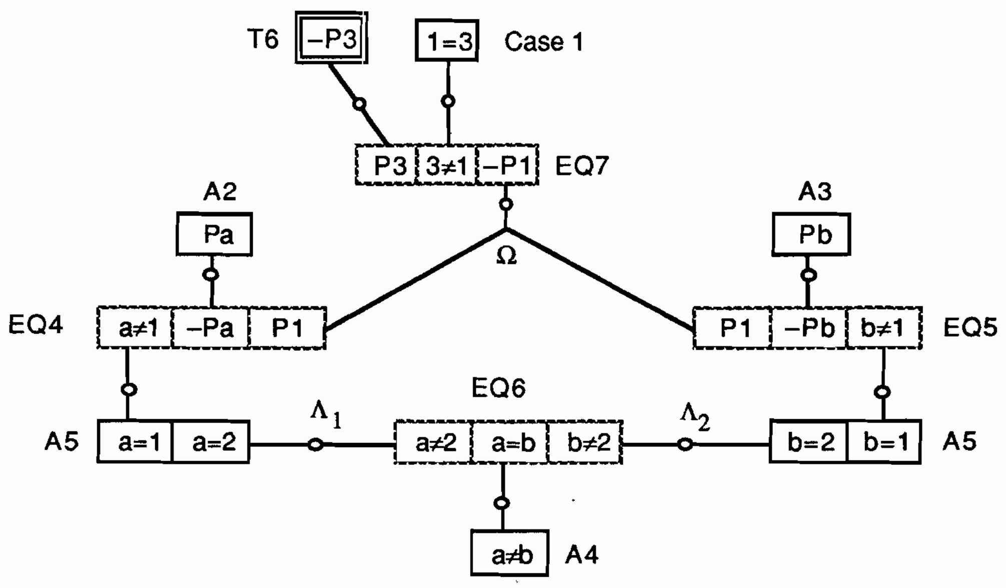
The proof is first translated into a refutation graph. The clause nodes I1 and I2 in the upper graph are both instances of the deduction graph below; a complete refutation graph can be obtained by inserting two copies of the deduction graph for I1 and I2. The relation Δ is also given; in this case the correspondence is straightforward.



The first operations performed in the transformation process are automatic applications of rules introducing the Skolem constants in the theorem. The other Skolem constants are introduced by need, whenever they appear in a subgraph that is currently worked with. Of course it may be necessary to isolate the existentially quantified formula first.

Now we know all the prerequisites for the structuring of this proof. At first we consider the subgraph which is used in two different copies in the refutation graph. Here the disjunction of free literal nodes $(x = 1 \vee x = 3)$ cannot immediately be used as a lemma because the deduction graph contains a negatively polarized clause node (T6). One heuristic to obtain a lemma in this case would be to use a maximal positively polarized subgraph of this deduction graph instead. This corresponds to a lemma $P3 \vee \forall x\,(x = 1 \vee x = 3)$. Note that the variable situation allows the introduction of a universally quantified lemma.

Alternatively, we can split the proof into cases as the situation meets the conditions explained above. The instance $1=3 \vee 2=3$ of axiom A5 is used, so that only a very small overlap remains in one of the resulting cases. Actually this overlap corresponds only to a trivial rewrite step. Below the refutation graph for the first case is shown.



It remains to be checked whether the two cases of the proof should be further structured. The links Λ_1 and Λ_2 are found to be separating nin case 1, indicating a proof by analyzing the cases $a = 2$ and $a \neq 2$ or, alternatively, $b = 2$ and $b \neq 2$. The symmetry of the graph suggests, however, a division of the proof into cases $a \neq 2$ and $b \neq 2$. This reflects the reasoning "$a \neq b$, therefore it is impossible that both a and b are equal to 2. If $a \neq 2$...". The second case has exactly the same structure and can therefore be done using the same case analysis.

Now the following global structure of the proof has become visible:

$$\begin{tikzcd}
 & & & 1=3 \lor 2=3 \arrow[dll, Rightarrow] \arrow[dr, Rightarrow] & \\
a\neq 2 \arrow[d, Rightarrow] & \lor & b\neq 2 \arrow[d, Rightarrow] & & a\neq 2 \lor b\neq 2 \\
a=1 \arrow[d, Rightarrow] & & b=1 \arrow[d, Rightarrow] & & \vdots \\
P3 & & P3 & &
\end{tikzcd}$$

There are x and y such that any z equals x or y. Two distinct constants a and b have a property P. Then the property P holds universally.

A proof in natural language might therefore read as follows:

Let 1 and 2 be constatns such that any z equals 1 or 2. In order to prove P as a universal property it suffices to show that P holds for an arbitrary constant 3. 3 must equal either 1 or 2. Let's first consider the case 3=1. Since $a \neq b$ it is impossible that both are equal to 2. If $a \neq 2$ then a must be 1, which equals 3, and therefore P3 holds because Pa holds. If on the other hand $b \neq 2$ then b must be 1, which equals 3, and therefore P3 holds because Pb holds. The second case (3=2) can be handled analogously. Therefore P3 holds in all cases, and as 3 was chosen arbitrarily P holds universally.

5 Conclusion

In this report a method is described to transform a proof generated by a resolution-based theorem prover with a built-in paramodulation rule into a natural deduction proof in Gentzen's system NK. Starting from the basic proof transformation and structuring mechanism published in [Li90], the necessary changes and additions are made to meet the special needs of equality reasoning.

Paramodulation steps are represented in the refutation graph by means of equality clause nodes and additional links for each application of an equation. The extension of this mechanism to arbitrary theory resolution appears to be a straightforward affair. The most remarkable result is the fact that this basis allows to employ the structuring algorithm essentially unchanged. The only extensions were to handle conditional equations by case analysis and some specialized heuristics for the consideration of equational steps.

An open question with respect to the structuring of proofs is the presentation of proofs based only or mainly on the equality predicate. The representation of pure unconditional 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 described here.

6 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**, Springer-Verlag, Proc of 5th CADE (1980), pp. 281-292

[An81] Peter B. Andrews — *Theorem Proving via General Matings*
JACM **28** (1981), pp. 193-214

[Bi81] Wolfgang Bibel — *On Matrices with Connections*
JACM **28** (1981), pp. 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

[Da81] Martin Davis — *Obvious Logical Inferences*
Proc. 7th IJCAI, Vancouver (1981)

[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** (1935), pp.176-210

[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**, Springer-Verlag, Proc of 8th CADE (1986), pp. 682-683

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

[Hu90] Xiaorong Huang *On a Natural Calculus for Argument Presentation* to appear as SEKI-Report, Universität Kaiserslautern (1990)

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

[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)

[Li90] Christoph Lingenfelder *Structuring Computer Generated Proofs* PhD Thesis, Universität Kaiserslautern (1990)

[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, Philadelphia

[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
Lecture Notes in AI **449**, Springer-Verlag, Proc of 10th CADE (1990), pp. 336-350

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

[Pe86] Francis Jeffrey Pelletier *Seventy-Five Problems for Testing Automatic Theorem Provers*
JAR, Vol.2, No.2 (1986), pp. 191-216

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