Proof Transformation with Built-in Equality Predicate

Christoph Lingenfelder
IBM Deutschland GmbH
Institute for Knowledge Based Systems
P.O. Box 103068
D-6900 Heidelberg
phone: 49 6221 404 359
email: LINGENF@DHDIBMI.bitnet

Axel Pracklein*
Fachbereich Informatik
Universitat 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.

1 Introduction

With the increasing strength of Automated Deduction Systems the length and complexity of computer generated proofs has reached a degree where they have become almost impossible to understand even for the expert. This has led to a stale where only specialists are capable to understand and check a proof found by an automated deduction system.

But whenever human beings are addressed, the need for easily understandable and clearly structured arguments is apparent. Therefore it is necessary to be able to present proofs in a 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 is useful to transform the computer generated proof into natural deduction 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 [Andrews, 1980; Miller, 1983; Lingenfelder, 1986], 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. [Lingenfelder, 1989; Pfenning and Nesmith, 1990; Huang, 1991], 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, we generalize Lingenfelder's approach, so that proofs involving equational reasoning can also be structured automatically.

In section 2 the different calculi and proof representations are introduced. Section 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 [Stickel, 1985]) so that a further generalization to arbitrary theories can be envisaged.

The task of finding the underlying proof structure is presented in section 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 a proof 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 section we will describe the logical calculi used in this paper. Everything is standard first order predicate logic with equality, and we need resolution (with paramodulation) and a natural deduction system based on Gerhard Gentzen's calculus NK [Gentzen, 1935]. There are no differences from the usual way of defining these concepts as done for instance in [Gallier, 1986] or [Loveland, 1978].

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.

Supported by the Deutsche Forschungsgemeinschaft, SFB 314

2.1 Clause and Refutation Graphs with Equality

Definition: A *clause graph* consists of a set of literal nodes, that are partitioned into clause nodes. Each literal node is labeled with a literal; the distinction between the literal nodes and the literals themselves is needed because the same literal may be attached to several literal nodes. Finally the links of the clause graph connect disjoint sets of literal nodes, such that for all links the following conditions hold:

(KI) The literal nodes in a link are labeled with literals with unifiable atoms.

(7C2) A link must connect at least one positive and one negative literal.

Definition: A clause graph is said to *represent* a clause set S if every clause node C has the form [–A(s) s≠t A(t)] or there is a *parent clause* C ∈ S and a ground substitution γ such that the restriction of the literal labeling to C is a bijection between its literal nodes and the literals of γC'. Clause nodes of the form (-A(s) s≠t A(t)J are called *equality clause nodes.*

Definition: A *deduction graph* is a non-empty, ground, and acyclic clause graph. A *cycle* is a sequence of clause nodes and links $C_1,\Pi_1,C_2,.C_n,\Pi_n,Ci$, such that all the rii are different and they contain literal nodes with opposite sign in their respective neighbour clause nodes. A *refutation graph* is a deduction graph where all literal nodes belong to a link.

Definition: For a formula F and a clause graph *T* representing C(F), a relation A between the literal nodes of *F* and the atom occurrences in F is a *clause graph relation* if it is compatible with the relation established by the normalization process, by which 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 of F.

2.2 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 [Gentzen, 1935J: *"der mdglichst genau das richtige logische SchlieBen bei mathematischen Beweisen wiedergibt"*.

The actual form of the proof lines is taken from Andrews [Andrews, 1980], but they differ only in their syntax from Gentzen's rule system NK in [Gentzen, 1935J.

Definition: A natural deduction proof line consists of a finite set of formulae, called the *assumptions,* a single formula, called *conclusion,* and a *justification,* written *{A* + F Rule R}. A finite sequence S of proof lines is a *natural deduction derivation* of a formula F from assumptions J3, if F is the conclusion of the last line of S, *A* is the set of assumptions of this last line, and every line in S is correctly justified by one of the rules of the calculus.

A proof line k={#t- F Rule 9?) within a sequence of proof lines is correctly justified iff ^f F matches the lower part of Rule 9t and there are proof lines before *X* in the sequence matching the upper part of Rule 9?.

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.

Now we list some rules of the natural deduction calculus, the letters F, G, and H represent formulae and $\mathcal{A}$ represents a finite set of formulae.

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

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

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

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

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

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

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

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

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

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

In addition to the rules for Gentzen's calculus NK we add the following rules to handle the equality predicate:

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

Rule of Equality (=):

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

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 { F}. Such a line is obviously no proof, because it is not correctly justified. Now the proof is constructed by deriving subgoals until it 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 [Lingenfelder, 1990].

3.1 General Procedure

Definition: A finite sequence S of proof lines is called a *Generalized Natural Deduction Proof (GNDP)* of a formula F, if F is the conclusion of the last line of S, the last line of S has no assumptions, and 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 calcu-

lus, but it is assumed that these lines are "sound", in the sense that ($\wedge$assumptions $\Rightarrow$ conclusion) is a valid formula. 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.

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

3.2 Transformation Rules

The transformation rules are to be read as follows: the lines before the arrow ($\longrightarrow$) are replaced by those following it in the next generalized NDP of the sequence. Some of the rules add a new internal line which is simply written below its parent lines. In the 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 $\Re$ 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. If the proof is given in form of a refutation graph this can be accomplished automatically.

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.

I$\forall$:

To	(α)	$\mathcal{A}$	$\vdash$	$\forall x Fx$	Rule $\Re$
add	(β)	$\mathcal{A}$	$\vdash$	Ft	$\forall E(\alpha)$

for an arbitrary term t.

M-Cases:

	(α)	$\mathcal{A}$	$\vdash$	$A \vee B$	$\Re$
	(ζ)	$\mathcal{A}$	$\vdash$	F	π
$\longrightarrow$	(α)	$\mathcal{A}$	$\vdash$	$A \vee B$	$\Re$
	We consider separately the cases of (α)				
	Case 1:				
	(β)	$\mathcal{A}$, A	$\vdash$	A	Hyp
	(γ)	$\mathcal{A}$, A	$\vdash$	F	π_1
	Case 2:				
	(δ)	$\mathcal{A}$, B	$\vdash$	B	Hyp
	(ε)	$\mathcal{A}$, B	$\vdash$	F	π_2
	End of cases (1, 2) of (α)				
	(ζ)	$\mathcal{A}$	$\vdash$	F	Cas(α,γ,ε)

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-=$\perp$:

	(γ)	$\mathcal{A}$	$\vdash$	$\perp$	π
$\longrightarrow$	(α)		$\vdash$	$s = s$	Ref
	(β)	$\mathcal{A}$	$\vdash$	$s \neq s$	π'
	(γ)	$\mathcal{A}$	$\vdash$	$\perp$	Contra

E-=:

	(α)	$\mathcal{A}$	$\vdash$	$s = t$	Rule $\Re$
	(γ)	$\mathcal{A}$	$\vdash$	F(t)	π
$\longrightarrow$	(α)	$\mathcal{A}$	$\vdash$	$s = t$	Rule $\Re$
	(β)	$\mathcal{A}$	$\vdash$	F(s)	π'
	(γ)	$\mathcal{A}$	$\vdash$	F(t)	=(β,α)

This is not the complete set of transition rules; all the other rules necessary to describe a complete system for proof transformation can be found in [Lingenfelder, 1990].

3.3 Proof Transformation System with Equality

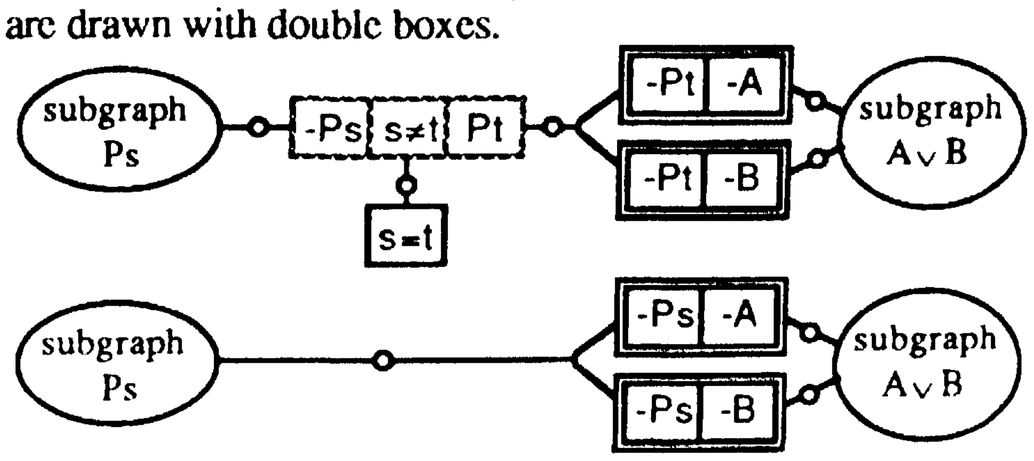
The complete set of transformation rules constitutes a proof system for natural deduction proofs. This means that for any valid formula F, there is a finite sequence of GNDPs starting with { $\vdash$ F } and ending with an NDP for F. Every element in this sequence can be constructed from its predecessor by application of one of the transition rules.

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. In [Lingenfelder, 1990] 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 t by s in a goal formula F a refutation graph for F(s) can be constructed by removing the equality clause node [–L(s) $s \neq t$ L(t)] and the equation clause node [s = t] unless used elsewhere. Then in all the literal nodes adjacent to L(t) t is substituted by s. Now the link at –L(s) is combined with that at L(t), thus closing the gap resulting from the removal of the equality clause node. The resulting graph proves the formula F(s).

Similarly, when E-=$\perp$ is applied, the polarization of the clause node [$s \neq s$] 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.

Example: In this example we show a refutation graph for F(t), which equals $Pt \wedge (A \vee B)$, and the resulting graph proving F(s): $Ps \wedge (A \vee B)$. The negatively polarized clause nodes are drawn with double boxes.



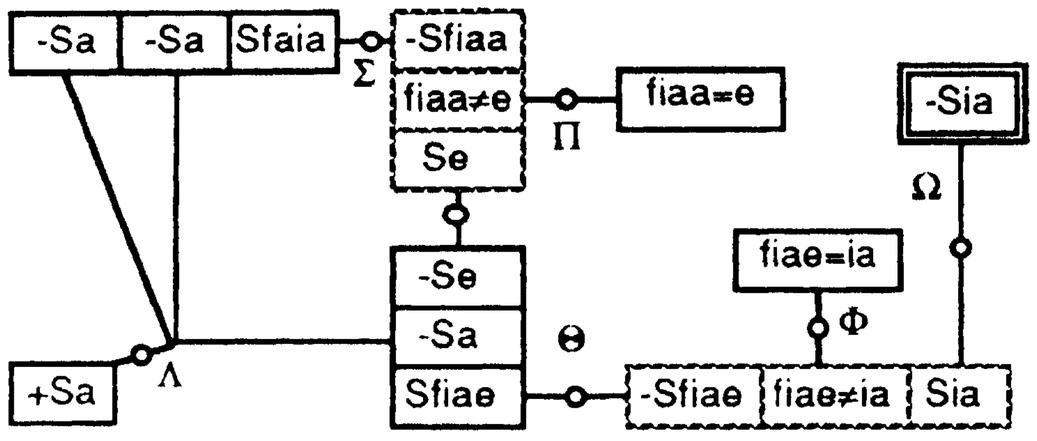
3.4 Application of Transition Rules

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

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

The refutation graph, the clause graph relation, and the

ground substitution have been automatically generated by the theorem prover MKRP, see [Ohlbach and Sickmann, 1989; Lehr, 1988]. Below the refutation graph is shown; theory clause nodes are indicated by dashed lines.



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

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

After some transformation steps not involving equality, the following GNDP is constructed:

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

Let a be an arbitrary constant

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

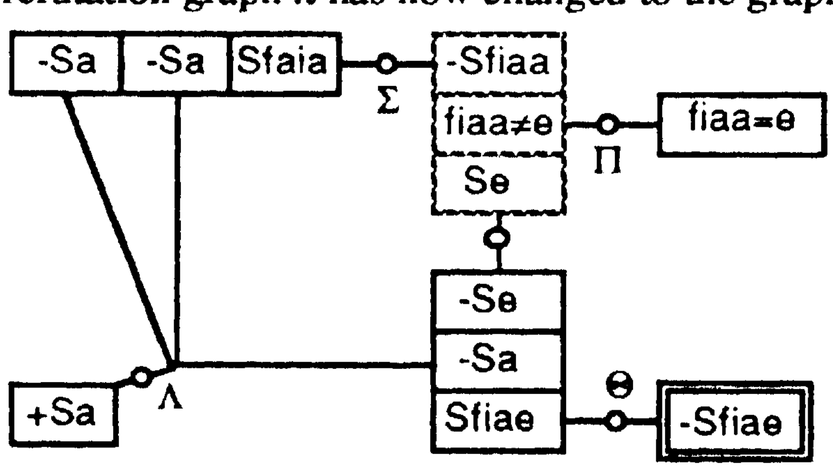
At this point the external line 13 cannot further be processed by external rules. A consultation of the refutation graph tells us that the literal node corresponding to Sia is only connected to a theory clause node. Therefore Sia must be derived using the rule of equality. As a preliminary operation we must isolate the equation fiae = ia, using internal rules, then E-= can be applied. Now the GNDP takes on the form shown below:

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

Let a be an arbitrary constant

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

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



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

4 Structuring the Proof

4.1 General Procedure

An initial "trivial" generalized natural deduction proof (GNDP) is constructed to start a transformation process as described in section 3. Now some of the transformation rules, EA for instance, lead to additional external lines, and as a consequence to a division of the refutation graph according to the splitting theorem [Lingenfelder, 1990]. In the simplest case the refutation graph proving F A G is "cut" through the clause [-F-G], such that the two resulting components are refutation graphs for F and G, respectively. In general, however, the two components may have a non-empty intersection, and this is similarly the case for the other rules leading to a division of the refutation graph. The splitting theorem does not take this into account, so that these shared subgraphs are always duplicated and therefore processed more than once.

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

E-Lemma

$$\begin{array}{llll} (\beta_1) & \mathcal{A}_1 & \vdash & F_1\ \pi_1 \\ \vdots & & & \vdots \\ (\beta_n) & \mathcal{A}_n & \vdash & F_n\ \pi_n \end{array}$$

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

This rule must of course be used with discretion, i.e. only when specifically called for by a heuristic. In [Lingenfelder, 1990] it is explained in detail how topological properties of the graph induce lemmata. If paramodulation steps are represented using equality clause nodes in the refutation graph the search algorithm in [Lingenfelder, 1990] can be used unchanged.

Another way to structure proofs is the division into the cases of a disjunction. This is formalized by the rule M-Cases. The most important case for its application in pure first order logic comes up, when an existentially quantified formula cannot be proven constructively. With built-in equality 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 Equality Clause Nodes

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 a resolution step between Pb

and some literal -Pb is simulated as a sequence of three resolution steps of the unit clauses [Pa], [a=b], and [-Pb] with the equality clause node **[-Pa a≠b Pb]**. This 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 explicitly, viz.

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

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

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 deduction graph. However, the formulation of the proof can be more difficult because the equation is not always true. One can either prove the equality as a lemma or divide the proof into cases, in one of which the equality holds.

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 $a \neq b$ is first derived from the contrapositive of $Pa \wedge a=b \Rightarrow Pb$, a case analysis may be the best choice.

Often several equations are successively applied to a formula leading to chains of equality clause nodes. If any of the chain 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.

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.

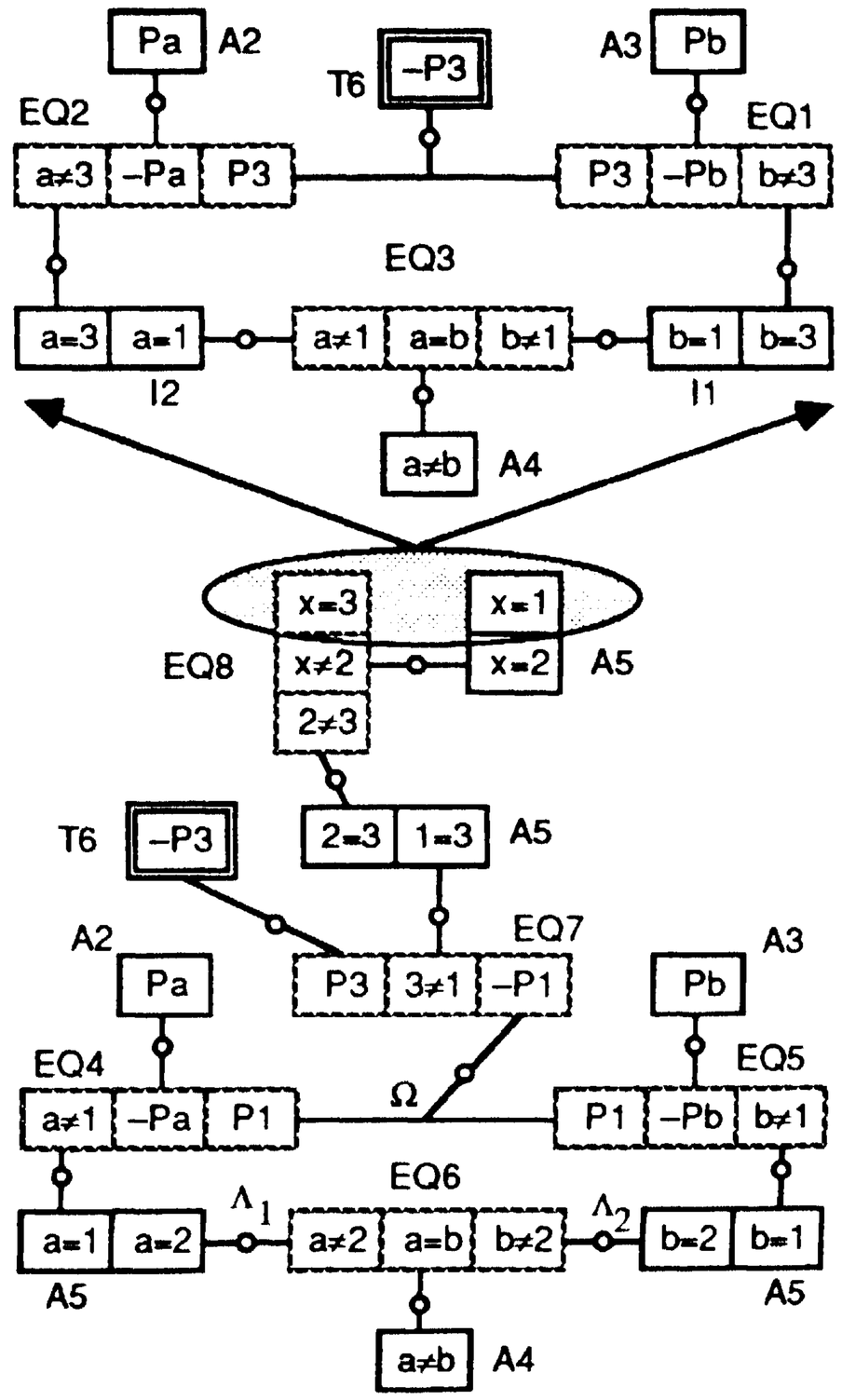
4.3 Example

As an example we chose one of Pelletier's problems [Pelletier, 1986], which is among the simpler standard examples in equality theorem proving:

There are x and y such that any z equals x or y. If two distinct constants a and b have a property P then the property P holds universally. In first order notation with equality this is represented by the following formula:

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

The resolution and paramodulation proof is first translated into a refutation graph. The clause nodes II and 12 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 II and 12. x and y, as well as w become Skolem constants in clause form, and therefore also in the refutation graph. They are named 1, 2, and 3 in the sequel.



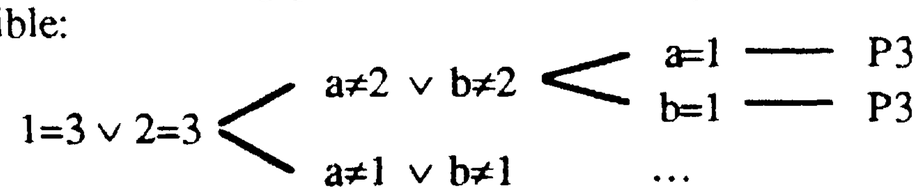
The first operations performed in the transformation process are automatic applications of rules introducing the Skolem constants in the theorem. The other Skolem constants are introduced by need, whenever they appear in a subgraph that is currently worked with. Of course it may be necessary to isolate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

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 in 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. Then the following global structure of the proof has become visible:



A proof in natural language might therefore read: let 1 and 2 be constants such that any z equals 1 or 2. In order to prove P as a universal property it suffices to show that it holds for an arbitrary constant 3. 3 must be cither 1 or 2. We consider first the case 3 = 1. As a=b it is impossible that both equal 2. If a=2 it must be 1, which equals 3, and therefore P3 holds because Pa holds. If on the other hand b=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 paper 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 [Lingenfelder, 1990], the necessary changes and additions are made to meet the special needs of equality reasoning.

Paramodulation steps are represented in the refutation graph by equality clause nodes and additional links for each application of an equation. The extension of this mechanism to arbitrary theory resolution appears to be straightforward. 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 or as a lemma 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 [Blasius, 1986], seems to be a promising starting point to construct a procedure analogous to the algorithm described here.

References

[Andrews, 1980] Peter B. Andrews. *Transforming Matings into Natural Deduction Proofs.* Proc of 5^{tn} CADE, pages 281-292, Springer-Verlag, 1980.

[Blasius, 1986] Karl-Hans Blasius. *Equality Reasoning Based on Graphs.* PhD Thesis, Uni Kaiserslautern, SEKI-ReportSR-87-01, 1986.

[Gallier, 1986] Jean H. Gallier. *Logic for Computer Science, - Foundations of Automatic Theorem Proving.* Harper & Row, Publishers, New York, 1986.

[Gentzen, 1935] Gerhard Gentzen. *Untersuchungen uber das logische SchlieBen I.* Math. Zeitschrift 39:176-210, 1935.

[Eisinger, 1988] Norbert Eisinger. *Completeness, Confluence, and Related Properties of Clause Graph Resolution.* PhD Thesis, Uni Kaiserslautern, Report SR-88-07, 1988.

[Huang, 1991] Xiaorong Huang. *On a Natural Calculus for Argument Presentation,* to appear as SEKI-Report, Uni Kaiserslautern, 1991.

[Lehr, 1988] Siegfried Lehr. *Transformation von Resolutionsbeweisen des MKRP.* Studienarbeit, Uni Kaiserslautern, 1988.

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

[Lingenfelder, 1989] Christoph Lingenfelder. *Structuring Computer Generated Proofs.* Proc of 11th IJCAI, Detroit, 1991.

[Lingenfelder, 1990] Christoph Lingenfelder. *Structuring Computer Generated Proofs.* PhD Thesis, Uni Kaiserslautern, 1990.

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

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

[Ohlbach and Siekmann, 1989] Hans J. Ohlbach and Jorg H. Siekmann. *The Markgraf Karl Refutation Procedure.* Report SR-89-19, Uni Kaiserslautern, 1989.

[Pfenning and Nesmith, 1990] Frank Pfenning and Daniel Nesmith. *Presenting Intuitive Deductions via Symmetric Simplification.,* Proc of 10th CADE, pages 336-350, Springer-Verlag, 1990.

[Pelletier, 1986] Francis Jeffrey Pelletier. *Seventy-Five Problems for Testing Automatic Theorem Provers.* Journal of Automated Reasoning, 2(2):191-216, 1986.

[Stickel, 1985] Mark E. Stickel. *Automated Deduction by Theory Resolution.* Journal of Automated Reasoning, I(4):333-356, 1985.