# EXPANSION TREE PROOFS AND THEIR CONVERSION TO NATURAL DEDUCTION PROOFS*

**Dale A. Miller**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

**Abstract:** We present a new form of Herbrand's theorem which is centered around structures called expansion trees. Such trees contains substitution formulas and selected (critical) variables at various non-terminal nodes. These trees encode a shallow formula and a deep formula — the latter containing the formulas which label the terminal nodes of the expansion tree. If a certain relation among the selected variables of an expansion tree is acyclic and if the deep formula of the tree is tautologous, then we say that the expansion tree is a special kind of proof, called an ET-proof, of its shallow formula. Because ET-proofs are sufficiently simple and general (expansion trees are, in a sense, generalized formulas), they can be used in the context of not only first-order logic but also a version of higher-order logic which properly contains first-order logic. Since the computational logic literature has seldomly dealt with the nature of proofs in higher-order logic, our investigation of ET-proofs will be done entirely in this setting. It can be shown that a formula has an ET-proof if and only if that formula is a theorem of higher-order logic. Expansion trees have several pleasing practical and theoretical properties. To demonstrate this fact, we use ET-proofs to extend and complete Andrews' procedure [4] for automatically constructing natural deductions proofs. We shall also show how to use a mating for an ET-proof's tautologous, deep formula to provide this procedure with the "look ahead" needed to determine if certain lines are unnecessary to prove other lines and when and how backchaining can be done. The resulting natural deduction proofs are generally much shorter and more readable than proofs build without using this mating information. This conversion process works without needing any search. Details omitted in this paper can be found in the author's dissertation [16].

**Key Words:** Higher-order Logic, Expansion Trees, ET-proofs, Natural Deduction, Matings.

## 1. Introduction

Problem solving in mathematics involves many different kinds of reasoning processes: about propositional connectives, about individual objects in a given domain, about equality and order relations, about sets and functions, and, among a host of others, the more exotic reasoning by example, analogy, etc. Approaches to theorem proving have generally focused on studying the first three of these reasoning processes. Reasoning of the more exotic kinds have also been studied by various artificial intelligence researchers. Although logics based on the ability to reason about sets and functions (higher-order logics) have been studied (see [1, 2, 8, 11, 12, 14, 18, 19, 20, 22]), until very recently few implementations of theorem provers in such logics have been described in the literature.

* This work was supported by NSF grant MCS81-02870.

The importance of doing theorem proving in higher-order logic has been argued by several people, including Andrews in [15] and Robinson in [20]. In fact, Robinson concludes [20] with:

> It is important to recognize that it is higher-order logic, and not first-order logic, which is the natural technical framework for the 'mechanization of mathematics'. We have in fact attempted ... to pursuade those engaged in mechanical theorem-proving research, and those proposing to start such research, to focus their attention henceforth on mechanizing higher-order logic.

The computer system, TPS, described in [15], is the result of an ongoing project in which many automatic and interactive approaches to theorem proving in higher-order logic have been developed. Initially, the logical language described in the next section was implemented, along with Huet's higher-order unification algorithm [12]. Using this unification algorithm, a mating enumeration theorem prover, described in [5], was then implemented. The mating enumeration strategy, being conceived for first-order logic theorem proving, provided TPS with an automatic theorem prover which was complete for first-order logic. The use of Huet's algorithm also permitted genuinely higher-order theorems to be proved. In particular, TPS found a proof of Cantor's theorem, *i.e.* there is no one-to-one mapping from a power set of a set back to that set. The classical diagonal argument was found by discovering a non-trivial higher-order substitution term [3]. However, the collection of theorems for which TPS could (theoretically) find a proof was only a very modest extension of first-order logic. Currently, TPS is still very incomplete in the general higher-order setting, since substitution terms containing quantifiers and binary, logical connectives are not discovered by a straightforward use of Huet's unification algorithm.

One way to describe this inadequacy is to say that TPS was not searching through the proper "search space" of proof structures for higher-order theorems. What we shall present in this paper is a useful characterization of the search space for a theorem prover in higher-order logic. This characterization is based on a structure called an expansion tree. Among the set of expansion trees for a given theorem, certain ones will be considered proofs, called ET-proofs. In this paper we shall define and present several important properties of expansion trees, but we shall not discuss the many issues surrounding how to automate the search for ET-proofs. Approaches to this problem are currently being studied and implemented in TPS.

In the next section, we shall describe the higher-order logic $\mathcal{T}$ on which we base the rest of this paper. In Section 3, we present the definition of expansion trees and ET-proofs. As it turns out, expansion trees and ET-proofs are useful structures for the study of proofs in both first-order and higher-order logic. All the results in this and following sections will work equally well in both logics. (See [17] for examples of how expansion trees can be used in first-order logic metatheory.) The fact that this one kind of structure can actually be used in both setting is clearly one of its strengths, since most definitions of search spaces for first-order logic do not work in the higher-order case. In Section 4, we present a list representation for expansion trees which are succinct and easily implemented. We warn the reader that these 3 sections may prove to be difficult to read, especially for a reader not familiar with higher-order logic and $\lambda$-conversion.

In the remaining sections of this paper, we show another strength of expansion trees: Given an ET-proof, it is easily converted to a natural deduction proof. Such a feature is also very important to the TPS project since a considerable amount of effort has gone into providing the TPS user with not only automatic tools for proving theorems but also interactive tools (see [4] and [15]). For example, TPS provides a interactive editor for constructing natural deduction proofs in a top-down and bottom-up fashion (much as is also done in [6]). At any point in editing such a proof, the user can have an unfinished portion of the proof given to the automatic theorem prover. If the theorem prover finds a proof — that is, an ET-proof — the methods described in the last sections can be used to finish the unfinished portion of the natural deduction proof which the user started. In this way, the theorem prover can explain the proof it found in a readable fashion. This capability is clearly valuable not only to researcher using TPS but also to beginning logic students who use the interactive proof editor to learn the process of building proofs. If the student gets stuck, the automatic theorem prover could provide hints or complete instructions on how to complete the proof. See [17] for more discussion of this feature, along with the description of how it is possible to convert natural deduction proofs to ET-proofs. In that paper, Pfenning also describes an algorithm which will convert a resolution-style refutation of a theorem into an ET-proof of that theorem. Thus the results about expansion trees mentioned in this paper and Pfenning's can be made available to those systems which are based on resolution theorem proving.

All the results described below are contained in [16], and the reader is referred to this dissertation for details of proofs which are omitted below.

## 2. Logical Preliminaries

It has often be observed that first-order logic is inadequate for formulating mathematics. For example, consider Tarski's lattice-theoretical fixpoint theorem [23]:

> If $\langle L, \leq \rangle$ is a complete lattice and if $f$ is an increasing function on $L$, then $f$ has a fixpoint, *i.e.* there is an $x \in L$ such that $f(x) = x$.

One difficult in representing this theorem and its proof in first-order logic is the need to quantify over a set variable in the axiom concerning a *complete* lattice. If we let $\langle L, \leq \rangle$ be a lattice and let $B$ be a set variable (a higher-order variable), an informal mathematical representation of the completeness axiom would be:

$$\begin{aligned}\forall B\,[\forall x\,[x \in B \supset x \in L] \supset \\ \exists z\,[z \in L \wedge \forall x\,[x \in B \supset x \leq z] \\ \wedge \forall y\,[[y \in L \wedge \forall x\,[x \in B \supset x \leq y]] \supset z \leq y]]]\end{aligned}$$

In the proof of the fixpoint theorem, this axiom is used by applying it to the set $\{x | x \in L \wedge x \leq f(x)\}$. Informally this is done referring to the property used to define $B$ wherever $x \in B$ appears in this instance of the axiom. In other words, we actually replace $x \in B$ with $x \in L \wedge x \leq f(x)$. Here, again first-order logic is inadequate

to represent this kind of substitution — an atomic formula $x \in B$ becomes the non-atomic formula $x \in L \wedge x \leq f(x)$. We now present a higher-order logical system which solves these two problems: explicit quantification of set (and function) variables and the possible change of atomic subformula occurrences into non-atomic subformulas under substitution.

The higher-order logic, $\mathcal{T}$, which we shall consider here is essentially the simple theory of types given by Church in [10], except that we do not use the axioms of extensionality, choice, descriptions, or infinity. $\mathcal{T}$ contains two base types, $o$ for boolean and $\iota$ for individuals. All other types are functional types, *i.e.* the type $(\beta\alpha)$ is the type of a function with domain type $\alpha$ and codomain type $\beta$. In particular, the type $(o\alpha)$, being the type of a function from type $\alpha$ to a boolean, *i.e.* a characteristic function, is used in $\mathcal{T}$ to represent the type for sets (predicates) of elements of type $\alpha$. For example, if the lattice $L$ mentioned earlier is a set of elements of type $\alpha$, then we say that $L$ has type $(o\alpha)$. Formulas are built up from logical constants, variables, and parameters (non-logical constants) by $\lambda$-abstraction and function application. Hence, the type of $[\lambda x_\alpha A_\beta]$ is $(\beta\alpha)$ while the type for $[A_{(\beta\alpha)} B_\alpha]$ is $\beta$. (We shall seldom adorn formulas with type symbols, but rather, when the type of a formula, say $A$, cannot be determined from context, we will add the phrase "where $A$ is a formula$_\alpha$" to indicate that $A$ has type $\alpha$.) For the convenience of making definitions in the next section, the formulas of $\mathcal{T}$ which we shall consider contain only the logical constants $\sim_{oo}$ (negation), $\vee_{(oo)o}$ (disjunction), and $\Pi_{o(o\alpha)}$ (the "universal $\alpha$-type set recognizer"). Other logical constants will be considered abbreviations, *i.e.* $A \wedge B$ stands for $\sim.\sim A \vee \sim B$, $A \supset B$ stands for $\sim A \vee B$, $\forall x\ P$ stands for $\Pi[\lambda x P]$, and $\exists x\ P$ stands for $\sim\Pi[\lambda x.\sim P]$. In particular, we write $L_{o\alpha} x_\alpha$ to denote the expression $x \in L$. This definition of the universal and existential quantifier may look rather peculiar, but it is very simple to explain. The meaning of the logical constant $\Pi$ is such that $\Pi_{o(o\alpha)} B_{o\alpha}$ is true if and only if $B_{o\alpha}$ is the "universal" set of type $o\alpha$. Hence, $\Pi[\lambda x_\alpha P_o]$ is true if and only if $\lambda x_\alpha P_o$ is the universal set of type $(o\alpha)$, *i.e.* $P_o$ is true for all $x_\alpha$. We shall take as axioms of $\mathcal{T}$ the following formulas ($p$, $q$, and $r$ are formulas$_o$):

$$p \vee p \supset p$$
$$p \supset p \vee q$$
$$p \vee q \supset .q \vee p$$
$$p \supset q \supset .r \vee p \supset .r \vee q$$
$$\Pi_{o(o\alpha)} f_{o\alpha} \supset f_{o\alpha} x_\alpha$$
$$\forall x_\alpha\ [p \vee f_{o\alpha} x_\alpha] \supset p \vee \Pi_{o(o\alpha)} f_{o\alpha}$$

Here, $\alpha$ is a type variable, and the last two axioms represent axiom schemes. The rules of inference are substitution, modus ponens, universal generalization, and $\lambda$-conversion. We shall write $\vdash_{\mathcal{T}} A$ to denote that $A$ has a Hilbert-style proof using these axioms and inference rules. The deduction theorem holds for $\mathcal{T}$.

At first glance $\mathcal{T}$ may look rather esoteric, but it can be described as being simply first-order logic in which we permit unrestricted comprehension via the use of $\lambda$-terms. The type structure is necessary here in order to avoid the paradoxes (like Russell's paradox) which arise from unrestricted comprehension. The use of $\lambda$-terms in substitutions can make the nature of deductions in $\mathcal{T}$ more complex than in first-order logic. In the fixpoint example, the result of substituting $B$ with the term $[\lambda x. Lx \wedge x \leq f(x)]$ in the

completeness axiom will change the subformulas of the form $Bx$ to $[\lambda x.Lx \wedge x \leq f(x)]x$ which $\lambda$-converts to $Lx \wedge x \leq f(x)$. For another example, if we have the formula (where $Y$ is a variable$_{o\iota}$ and $D$ and $T$ are variables$_{o(o\iota)}$)

$$\forall D\,[DY \supset TY]$$

and we wished to do a universal instantiation (a derived rule of inference) of this formula with the term $\lambda Z[TZ \wedge \forall x\,.Zx \supset Ax]$, *i.e.* the set of all sets of individuals which are members of $T$ and are subsets of $A$, we would then have

$$[\lambda Z.\ TZ \wedge \forall x\,.Zx \supset Ax]Y \supset TY.$$

We can now apply the $\lambda$-conversion inference rule to this formula to deduce

$$[TY \wedge \forall x\,.Yx \supset Ax] \supset TY.$$

Notice how the structure of this last formula is much more complex than that of the formula it was deduced from. This last formula contains occurrences of logical connectives and quantifiers which are not present in the original formula. Notice also that $Y$ now has the role of a predicate where this was not the case in the first formula. None of these structural changes can occur in first-order logic. The discovery of such substitution terms as the one used to instantiate $D$ is a much more complex problem than can be achieved by simply applying unification. TPS, for example, cannot currently discover terms of this kind. Radical new heuristics for finding substitutions must be developed, and we hope that expansion trees will provide a vehicle for formalizing such attempts. Bledsoe in [8] and [9] has made some exciting progress in the development of just such heuristics.

## 3. Expansion Trees and ET-Proofs

All references to trees below will actually refer to finite, ordered, rooted trees in which the nodes and arcs may or may not be labeled, and that labels, if present, are formulas. In particular, nodes may be labeled with simply the logical connectives $\sim$ and $\vee$. We shall picture our trees with their roots at the top and their leaves (terminal nodes) at the bottom. In this setting, we say that one node *dominates* another node if it they are on a common branch and the first node is *higher* in the tree than the second. This dominance relation shall be considered reflexive. All nodes except the root node will have *in-arcs* while all nodes except the leaves will have *out-arcs*. A node labeled with $\sim$ will always have one out-arc, while a node labeled with $\vee$ will always have two out-arcs. We shall also say that an arc dominates a node if the node which terminates the arc dominates the given node. In particular, an arc dominates the node in which it terminates. Also, we say that an arc dominates another arc if their respective terminal nodes dominate each other in the same order.

**3.1. Definition.** Let $A$ be a formula$_o$. An occurrence of a subformula $B$ in $A$ is a *boolean subformula occurrence* if it is in the scope of only $\sim$ and $\vee$, or if $A$ is $B$. A formula$_o$ $A$ is an *atom* if its leftmost non-bracket symbol is a variable or a parameter.

A formula $B$ is a ***boolean atom*** (***b-atom***, for short) if its leftmost non-bracket symbol is a variable, parameter or $\Pi$. A ***signed atom*** (***b-atom***) is a formula which is either an atom (b-atom) or the negation of an atom (b-atom). ∎

**3.2. Definition.** Formulas$_o$ of $\mathcal{T}$ can be considered as trees in which the non-terminal nodes are labeled with $\sim$ or $\vee$, and the terminal nodes are labeled with b-atoms. Given a formula$_o$, $A$, we shall refer to this tree as the ***tree representation of*** $A$. ∎

**3.3. Example.** Figure 1 is the tree representation of $\sim[\Pi B \vee Ax] \vee \sim\sim\Pi[\lambda x.Ax \vee Bx]$. This formula is equivalent to $\sim[\forall y\, By \vee Ax] \vee \sim\sim\forall x\, .Ax \vee Bx$. ∎

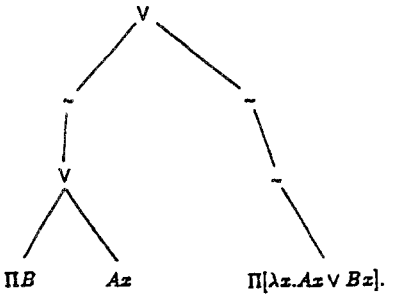


Figure 1

We shall adopt the following linear representation for trees. If the root of the tree $Q$ is labeled with $\sim$, we write $Q = \sim Q'$, where $Q'$ is the proper subtree dominated by $Q$'s root. Likewise, if the root of $Q$ is labeled with $\vee$, we write $Q = Q' \vee Q''$, where $Q'$ and $Q''$ are the left and right subtree of $Q$. The expression $Q' \wedge Q''$ is an abbreviation for the tree $\sim[\sim Q' \vee \sim Q'']$.

**3.4. Definition.** Let $Q$, $Q'$ be two trees. Let $N$ be a node in $Q$ and let $l$ be a label. We shall denote by $Q +^l_N Q'$ the tree which results from adding to $N$ an arc, labeled $l$, which joins $N$ to the root of the tree $Q'$. This new arc on $N$ comes after the other arcs from $N$ (if there are any). In the case that the tree $Q$ is a one-node tree, $N$ must be the root of $Q$, and we write $A +^l Q'$ instead of $Q +^l_N Q'$, where $A$ is the formula which labels $N$. ∎

**3.5. Example.** Figure 2 contains three trees, $Q$, $Q'$ and $Q +^c_N Q'$, where $N$ is a node of $Q$ and $c$ is some label. The nodes and arcs of $Q$ and $Q'$ may or may not have their own labels.

∎

**3.6. Definition.** Let $Q$ be a tree, and let $N$ be a node in $Q$. We say that $N$ occurs positively (negatively) if the path from the root of $Q$ to $N$ contains an even (odd) number of nodes labeled with $\sim$. We shall agree that the root of $Q$ occurs positively in $Q$. If a node $N$ in $Q$ is labeled with a formula of the form $\Pi B$, then we say that $N$ is ***universal*** (***existential***) if it occurs positively (negatively) in $Q$. A terminal node which is not labeled with a formula of the form $\Pi B$ is called a ***neutral*** node. A universal

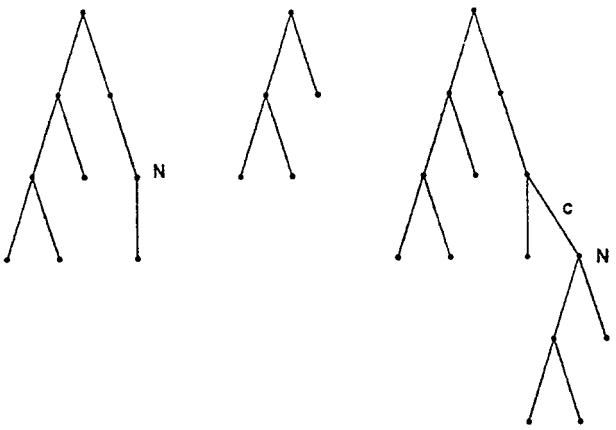


Figure 2: The trees $Q$, $Q'$, and $Q +^{c}_{N} Q'$.

(existential) node which is not dominated by any universal or existential node is called a *top-level* universal (existential) node. A labeled arc is a *top-level labeled arc* if it is not dominated by any other labeled arc. ∎

**3.7. Definition.** Let $Q$ be a tree with a terminal node $N$ labeled with the formula $\Pi B$, for some formula$_{o\alpha}$ $B$. If $N$ is existential, then an *expansion* of $Q$ at $N$ with respect to the list of formulas$_{\alpha}$, $\langle t_1, \ldots, t_n \rangle$, is the tree $Q +^{t_1}_{N} Q_1 +^{t_2}_{N} \cdots +^{t_n}_{N} Q_n$ (associating to the left), where, for $1 \leq i \leq n$, $Q_i$ is the tree representation for some $\lambda$-normal form of $Bt_i$. The formulas $t_1, \ldots, t_n$ are called *expansion terms* of the resulting tree. We say that each of these terms are used to *expand* $N$.

If $N$ is universal, then a *selection* of $Q$ at $N$ with respect to the variable$_{\alpha}$ $y$, is the tree $Q +^{y}_{N} Q'$, where $Q'$ is the tree representation of some $\lambda$-normal form of $By$, and $y$ does not label an out-arc of any universal node in $Q$. We say that the node $N$ *is selected by* $y$.

The set of all *expansion trees* is the smallest set of trees which contains the tree representations of all $\lambda$-normal formulas$_o$ and which is closed under expansions and selections. ∎

Expansion trees are, in a sense, generalized formulas. The main difference is that expansion trees can contain labeled arcs. An expansion tree which contains no labeled arcs can easily be interpreted as a formula.

**3.8. Definition.** Assume that $Q$ is an expansion tree. Let $\mathbf{S}_Q$ be the set of all variable occurrences which label the out-arcs from (non-terminal) universal nodes in $Q$, and let $\Theta_Q$ be the set of all occurrences of expansion terms in $Q$. ∎

Expansion trees, a generalization of Herbrand instances, do not use Skolem func-

tions as is customary in Herbrand instances. Skolem functions can be used in this setting, but their occurrences in substitution terms must be restricted in ways that are not apparent from the first-order use of Skolem functions. The reader is referred to [16] for details. In order to do without Skolem functions, we need to place a restriction on selected variables which models the way in which Skolem terms would imbed themselves in other Skolem terms. This restriction amounts to requiring that the following binary relation on $\mathbf{S}_Q$ be acyclic.

**3.9. Definition.** Let $Q$ be an expansion tree and let $\prec_Q^0$ be the binary relation on $\mathbf{S}_Q$ such that $z \prec_Q^0 y$ if there exists an expansion term occurrence $t \in \Theta_Q$ such that $z$ is free in $t$ and $y$ is selected for a node dominated by the arc labeled by $t$. Let $\prec_Q$ be the transitive closure of $\prec_Q^0$. $\prec_Q$ is called the *imbedding relation* because it reflects how Skolem terms, represented by the variables in $\mathbf{S}_Q$, are imbedded in one another. ∎

We next define two formulas which are encoded in an expansion tree. The "deep" formula $Dp(Q)$ of the expansion tree $Q$ is a formula whose b-atoms correspond to the leaves of $Q$. The "shallow" formula $Sh(Q)$ of $Q$ is a formula whose b-atoms correspond to the top-level universal, existential, and neutral nodes in $Q$.

**3.10. Definition.** Let $Q$ be a tree such that either $Q$ or $\sim Q$ is an expansion tree. We define $Dp(Q)$ by induction on the structure of $Q$.

(1) If $Q$ is a one-node tree, then $Dp(Q) = A$, where $A$ is the formula which labels that one-node.

(2) If $Q = \sim Q'$ then $Dp(Q) := \sim Dp(Q')$.

(3) If $Q = Q' \vee Q''$ then $Dp(Q) := Dp(Q') \vee Dp(Q'')$.

(4) If $Q = \Pi B +^{l_1} Q_1 + \ldots +^{l_n} Q_n$ then $Dp(Q) := Dp(Q_1) \wedge \ldots \wedge Dp(Q_n)$. ∎

**3.11. Definition.** Let $Q$ be a tree such that either $Q$ or $\sim Q$ is expansion tree. We define $Sh(Q)$ by induction on the top-level boolean structure of $Q$.

(1) If $Q$ is a one-node tree, then $Sh(Q) = A$, where $A$ is the formula which labels that one-node.

(2) If $Q = \sim Q'$ then $Sh(Q) := \sim Sh(Q')$.

(3) If $Q = Q' \vee Q''$ then $Sh(Q) := Sh(Q') \vee Sh(Q'')$.

(4) If $Q = \Pi B +^{l_1} Q_1 + \ldots +^{l_n} Q_n$ then $Sh(Q) := \Pi B$. ∎

Notice, that if $A$ is a formula$_o$, and $Q$ is the tree representation of $A$, then $Dp(Q) = A = Sh(Q)$.

**3.12. Definition.** Let $Q$ be an expansion tree. $Q$ is *sound* if no variable in $\mathbf{S}_Q$ is free in $Sh(Q)$. $Q$ is an *ET-proof* if $Q$ is sound, $Dp(Q)$ is tautologous, and $\prec_Q$ is acyclic. $Q$ is an expansion tree *for* $A$ if $Q$ is sound and $Sh(Q)$ is a $\lambda$-normal form of $A$. $Q$ is an ET-proof *for* $A$ if $Q$ is an ET-proof and $Q$ is an expansion tree for $A$. ∎

**3.13. Example.** Let $A$ be the theorem $\exists y\ \forall x\ .Px \supset Py$. An ET-proof for $A$ would then be the tree $Q$ given as:

$$\sim [[\Pi\lambda y. \sim \Pi\lambda x. \sim Px \vee Py] +^u \sim [[\Pi\lambda x. \sim Px \vee Pu] +^v [\sim Pv \vee Pu]] \\ +^v \sim [[\Pi\lambda x. \sim Px \vee Pv] +^w [\sim Pw \vee Pv]]].$$

Here, $Dp(Q) = \sim[\sim[\sim Pv \vee Pu] \wedge \sim[\sim Pw \vee Pv]]$. The imbedding relation is the pair $v \prec_Q w$. Notice, that if we had used $u$ instead of $w$, $\prec_Q$ would have been cyclic. In Example 4.2 we give a more readable representation of this expansion tree. ∎

**3.14. Soundness and Relative Completeness for ET-Proofs.** *Let $A$ be a formula$_o$. $\vdash_{\mathcal{T}} A$ if and only if $A$ has an ET-proof.*

This theorem is what we shall consider our higher-order version of Herbrand's theorem. The reader is referred to [16] for the details of this proof. The relative completeness result, *i.e.* if $\vdash_{\mathcal{T}} A$ then $A$ has an ET-proof, is proven by using the Abstract Consistency Property in [1]. The central result concerning Abstract Consistency Properties is based on Takahashi's proof of the cut-elimination theorem for higher-order logic [22]. Since $\mathcal{T}$ is non-extensional, Henkin-style general models do not correctly characterize derivability in $\mathcal{T}$. Hence, the completeness result is stated relative to the notion of derivability and is not based on a notion of validity.

**3.15. Definition.** An expansion tree is *grounded* if none of its terminal nodes are labeled with formulas of the form $\Pi B$. An ET-proof is a grounded ET-proof if it is also a grounded expansion tree. ∎

A formula has an ET-proof if and only if it has a grounded ET-proof.

## 4. List Representations of Expansion Trees

We shall now present a representation of expansion trees which is more succinct and more suitable for direct implementation on computer systems. We shall no longer consider the logic connectives $\wedge$ and $\supset$ and the quantifiers $\forall$ and $\exists$ to be abbreviations. This will help make list representations of expansion trees more compact.

The set of all list structures over a given set, $\Xi$, is defined to be the smallest set which contains $\Xi$ and is closed under building finite tuples.

Since expansion and selection nodes in an expansion tree must occur under an odd and even number of occurrences of negations respectively, we need to be careful how we imbed expansion trees under negations when we attempt to build up larger expansion trees from smaller ones. This explains why we need to consider so many cases in the following definition.

**4.1. Definition.** Let $\Xi$ be the set which contains the labels SEL and EXP and all formulas of $\mathcal{T}$. Let $\mathcal{E}$ be the smallest set of pairs $\langle R, A\rangle$, where $R$ is a list structure over $\Xi$ and $A$ is a formula$_o$, which satisfies the conditions below. We say that a variable $y$ is *selected* in the list structure $R$ if it occurs in a sublist of the form (SEL $y$ $R'$).

(1) If $A$ is a boolean atom and $R$ is a $\lambda$-normal form of $A$, then $\langle R, A\rangle \in \mathcal{E}$ and $\langle \sim R, \sim A\rangle \in \mathcal{E}$. Here, $\sim R$ is shorthand for the two element list $(\sim R)$.

(2) If $\langle R, A\rangle \in \mathcal{E}$ then $\langle R, B\rangle \in \mathcal{E}$ where $A$ conv $B$.

(3) If $\langle R, A\rangle \in \mathcal{E}$ then $\langle \sim\sim R, \sim\sim A\rangle \in \mathcal{E}$.

In cases (4), (5), and (6), we assume that $R_1$ and $R_2$ share no selected variables in common and that $A_1$ ($A_2$) has no free variable selected in $R_2$ ($R_1$).

(4) If $\langle R_1, A_1\rangle \in \mathcal{E}$ and $\langle R_2, A_2\rangle \in \mathcal{E}$ then $\langle (\vee\ R_1\ R_2), A_1 \vee A_2\rangle \in \mathcal{E}$ and $\langle (\wedge\ R_1\ R_2), A_1 \wedge A_2\rangle \in \mathcal{E}$.

(5) If $\langle \sim R_1, \sim A_1 \rangle \in \mathcal{E}$ and $\langle \sim R_2, \sim A_2 \rangle \in \mathcal{E}$ then $\langle \sim(\vee\ R_1\ R_2), \sim.A_1 \vee A_2 \rangle \in \mathcal{E}$ and $\langle \sim(\wedge\ R_1\ R_2), \sim.A_1 \wedge A_2 \rangle \in \mathcal{E}$.

(6) If $\langle \sim R_1, \sim A_1 \rangle \in \mathcal{E}$ and $\langle R_2, A_2 \rangle \in \mathcal{E}$ then $\langle (\supset\ R_1\ R_2), A_1 \supset A_2 \rangle \in \mathcal{E}$ and $\langle \sim(\supset\ R_2\ R_1), \sim.A_2 \supset A_1 \rangle \in \mathcal{E}$.

In cases (7), (8), and (9), we assume that $y$ is not selected in $R$ and that $y$ is not free in $[\lambda x P]$ or in $B$.

(7) If $\langle R, [\lambda x P]y \rangle \in \mathcal{E}$ then $\langle (\text{SEL}\ \ y\ \ R), \forall x\ P \rangle \in \mathcal{E}$.

(8) If $\langle \sim R, \sim[\lambda x P]y \rangle \in \mathcal{E}$ then $\langle (\sim(\text{SEL}\ \ y\ \ R)), \sim \exists x\ P \rangle \in \mathcal{E}$.

(9) If $\langle R, By \rangle \in \mathcal{E}$ then $\langle (\text{SEL}\ \ y\ \ R), \Pi B \rangle \in \mathcal{E}$.

In cases (10), (11), and (12), we must assume that for distinct $i, j$ such that $1 \leq i, j \leq n$, $R_i$ and $R_j$ share no selected variables and that no variable free in $[\lambda x P]t_i$ is free in $R_j$.

(10) If for $i = 1, \ldots, n$, $\langle R_i, [\lambda x P]t_i \rangle \in \mathcal{E}$ then $\langle (\text{EXP}\ \ (t_1\ R_1) \ldots (t_n\ R_n)), \exists x\ P \rangle \in \mathcal{E}$.

(11) If for $i = 1, \ldots, n$, $\langle \sim R_i, \sim[\lambda x P]t_i \rangle \in \mathcal{E}$ then $\langle \sim(\text{EXP}\ \ (t_1\ R_1) \ldots (t_n\ R_n)), \sim \forall x\ P \rangle \in \mathcal{E}$.

(12) If for $i = 1, \ldots, n$, $\langle \sim R_i, \sim Bt_i \rangle \in \mathcal{E}$ then $\langle \sim(\text{EXP}\ \ (t_1\ R_1) \ldots (t_n\ R_n)), \Pi B \rangle \in \mathcal{E}$. ∎

The pair $\langle R, A \rangle \in \mathcal{E}$ represents — in a succinct fashion — an expansion tree. Notice that the only formulas stored in the list structure $R$ are those used for expansions and selections and those which are the leaves of the expansion tree. Expansion trees as defined in §2 contain additional formulas which are used as "shallow formulas" to label expansion and selection nodes. These formulas, however, can be determined up to $\lambda$-convertibility if we know what the expansion tree is an "expansion" for. Notice, that one list structure alone may represent several expansion trees. For example, (EXP $(a\ Paa)$) could represent an expansion tree for $\exists x\ Pxx$, $\exists x\ Pax$, and $\exists x\ Paa$. If we keep this complication in mind, we can informally considered list structures as expansion trees.

**4.2. Example.** The expansion tree in Example 3.13 can be written as the list structure:

$$(\text{EXP}\ \ (u\ (\text{SEL}\ \ v\ (\supset Pv\ Pu)\ ))\ (v\ (\text{SEL}\ \ w\ (\supset Pw\ Pv)\ ))).$$

## 5. Natural Deductions

Beyond the fact that ET-proofs are sound and (relatively) complete for $\mathcal{T}$, they also have several other pleasing properties, for both theoretical and practical concerns. We shall illustrate this claim by showing how ET-proofs can be converted to natural deduction-style proofs. This investigation is an immediate extension of the work described by Andrews in [4]. In that paper, Andrews showed how natural deduction proofs could be constructed by processing incomplete proofs, called *outlines*, in both a top-down and bottom-up fashion. In these outlines, certain lines, called *sponsoring* lines, were not justified. To each sponsoring line is associated a (possibly empty) list of justified lines which appear earlier in the proof and which might be required for completing the

proof of the sponsoring line. These lines are called ***supporting*** lines. Proof lines which are either supporting or sponsoring are called ***active.*** Incomplete proofs built in this fashion are such that their assertions are subformulas of the original theorem. (Notice that in higher-order logic, this is stretching the usual meaning of subformulas.) Using this fact, we shall be able to attach to each active line an expansion tree (actually a list representation) *for* the assertion in that line. These expansion trees, which are essentially sub-trees of the ET-proof of the original theorem, provide the information necessary to determine how an active line should be "processed."

Beyond the fact that the conversion process describe below works for higher-order logic, this process differs in two other important ways from the process described in [4]. First, Andrews used a structure called a ***plan*** to provide the information which would indicate how to process active lines. ET-proofs, when restricted to first-order logic, contain the same kind of information as plans. Plans, however, are defined with respect to several global properties of formulas. This makes it awkward (in theory and practice) to construct new plans for new subproofs. Since subtrees or the negation of subtrees of expansion trees are themselves expansion trees, it is much easier to build new ET-proofs for new subproofs. Secondly, Andrews actually considered subproofs to be based on a sponsoring line and its hypotheses while we consider subproofs to be based on sponsoring lines and their supports. These differences allow us to give a complete analysis of this transformation process.

Below we provide formal definitions for the concepts informally discussed above. In the rest of this paper, all ET-proofs will be assumed to be grounded.

**5.1. Definition.** By a natural deduction proof we mean a Suppes-style proof structures [21]. Such systems emphasize reasoning from hypotheses instead of axioms. An ***incomplete natural deduction proof*** is a list of proof lines some of which are justified by NJ — the non-justification label. Such lines represent subproofs which must be completed. The rules of inference in this system are those listed in [4] along with a rule for $\lambda$-conversion. The rules of existential generalization and universal instantiation are examples of two rules of inference. ∎

**5.2. Example.** The following is an example of an incomplete natural deduction proof.

| | | | | |
|---|---|---|---|---|
| (1) | 1 | $\vdash$ | $\exists c\ \forall p\ .[\exists u\ .pu] \supset .p.cp$ | *Hyp* |
| (2) | 2 | $\vdash$ | $\forall x\ \exists y\ .Pxy$ | *Hyp* |
| (3) | 3 | $\vdash$ | $\forall p\ .[\exists u\ .pu] \supset .p.cp$ | *Hyp* |
| (16) | 2,3 | $\vdash$ | $\exists f\ \forall z\ .Pz.fz$ | *NJ* |
| (17) | 1,2 | $\vdash$ | $\exists f\ \forall z\ .Pz.fz$ | *RuleC* : 1, 16 |
| (18) | 1 | $\vdash$ | $[\forall x\ \exists y\ .Pxy] \supset .\exists f\ \forall z\ .Pz.fz$ | *Deduct* : 17 |
| (19) | | $\vdash$ | $[\exists c\ \forall p\ .[\exists u\ .pu] \supset .p.cp] \supset$ $[\forall x\ \exists y\ .Pxy] \supset \exists f\ \forall z\ .Pz.fz$ | *Deduct* : 18 |

Here $c$ is a variable$_{\iota(o\iota)}$, $p$ is a variable$_{o\iota}$, $P$ is a variable$_{o\iota\iota}$, $f$ is a variable$_{\iota\iota}$, and $x$, $y$, $z$, $u$ are variables$_{\iota}$. ∎

In what follows, we shall use $\perp$ to represent a false statement. It can be treated as an abbreviation for $p \wedge \sim p$. We shall also let $\perp$ stand for both the expansion tree for $\perp$ and for the list representation for this expansion tree. If $\perp$ occurs as one of the

disjuncts of a formula, we shall assume that that formula is an abbreviation for the formula which results from removing $\bot$ as a disjunct.

**5.3. Definition.** A *proof outline*, $\mathcal{O}$, is the triple, $\langle L, \rho, \{R_l\}\rangle$, where:

(1) $L$ is a list of proof lines which forms an incomplete natural deduction proof. A line with the justification $NJ$ corresponds to a subproof which must be completed. Let $L_0$ be the set of all lines labels in $L$ which have this justification. These are called the *sponsoring* lines of $\mathcal{O}$.

(2) $\rho$ is a function defined on $L_0$ such that whenever $z \in L_0$, $\rho(z) \subset L \setminus L_0$ and all the lines in $\rho(z)$ precede $z$ in the list $L$. Whenever $l \in \rho(z)$, we say that $z$ *sponsors* $l$, $l$ *supports* $z$, $z$ is a *sponsoring* line, and $l$ is a *supporting* line. A line is *active* if it is either a supporting line or a sponsoring line which does not assert $\bot$. (In the outlines we shall consider, only sponsoring lines may assert $\bot$.)

(3) $\{R_l\}$ represents a set of list structures, one for each active line, such that if $l$ is a supporting line, then $\langle \sim R_l, \sim l\rangle \in \mathcal{E}$ and if $l$ is a sponsoring line, then $\langle R_l, l\rangle \in \mathcal{E}$.

(4) If line $a$ supports line $z$ then the hypotheses of $a$ are a subset of the hypotheses of $z$.

If $L_0$ is not empty, we define the following formulas and expansion trees. For each $z \in L_0$ set $A_z := [\bigvee_{l\in\rho(z)} \sim l] \vee z$ (where line labels stand for their assertions) and let $Q_z$ be the expansion tree for $A_z$ represented by the list structure $(\vee\ (\bigvee_{l\in\rho(z)} \sim R_l)\ R_z)$. The following condition must also be satisfied by an outline.

(5) If $L_0$ is not empty, then $Q_z$ is a (grounded) ET-proof for $A_z$ for each $z \in L_0$.

It is easy to show that $\mathcal{O}$ has an active line if and only if $L_0$ is not empty. We say that $\mathcal{O}$ is an outline *for* $A$ if the last line in $\mathcal{O}$ has no hypotheses and asserts $A$. ∎

The ET-proof $Q_z$ roughly corresponds to a plan for the sponsoring line $z$ as described in [4].

**5.4. Definition.** Let $A$ be a formula and $R$ a list representation of an ET-proof for $A$. Let $z$ be the label for the proof line

$$(z) \qquad \vdash \quad A \qquad\qquad\qquad NJ,$$

and set $L := \langle z\rangle$, $\rho(z) = \emptyset$ and $R_z := R$. Then $\mathcal{O}_0 := \langle L, \rho, \{R_l\}\rangle$ is clearly an outline. We call this outline the *trivial outline for* $A$ *based on* $R$. ∎

**5.5. Example.** An example of a proof outline is given by setting $L = \langle 1, 2, 3, 16, 17, 18, 19\rangle$, $\rho(16) = \{2, 3\}$ and

$$\begin{aligned} R_2 &= (\text{EXP}\ \ (z\ (\text{SEL}\ \ y\ \ Pzy))) \\ R_3 &= (\text{EXP}\ \ (Pz\ (\supset\ (\text{EXP}\ \ (y\ Pzy))\ Pz.c.Pz))) \\ R_{16} &= (\text{EXP}\ \ ([\lambda v.c.Pv]\ \ (\text{SEL}\ \ z\ \ Pz.c.Pz))) \end{aligned}$$

where the lines in $L$ are those listed in Example 5.2. It is easy to verify that $(\vee\ \sim R_2\ (\vee\ \sim R_3\ R_{16}))$ represents an ET-proof of $\sim 2 \vee \sim 3 \vee 16$ and that $\langle L, \rho, \{R_2, R_3, R_{16}\}\rangle$ is an outline. ∎

**5.6. Definition.** A formula $t$ is *admissible in* $\mathcal{O}$ if no free variable in $t$ is selected in $R_l$ for any active line $l$. ∎

The D- and P- (deducing and planning) transformations described in [4] can now be used in this setting if we describe how each such transformation attributes expansion trees to each new active line. We illustrate how this is done with the P-Conj and D-All transformations.

If some sponsoring line $z$ in an outline $\mathcal{O} = \langle L, \rho, \{R_l\}\rangle$ is of the form

$$(z) \quad \mathcal{H} \quad \vdash \quad A_1 \wedge A_2 \qquad\qquad NJ$$

then $R_z$ is of the form $(\wedge\ R_1\ R_2)$. Applying P-Conj to line $z$ will result in an outline $\mathcal{O}' = \langle L', \rho', \{R'_l\}\rangle$, where $L'$ contains the new sponsoring lines

$$(x) \quad \mathcal{H} \quad \vdash \quad A_1 \qquad\qquad NJ$$
$$(y) \quad \mathcal{H} \quad \vdash \quad A_2 \qquad\qquad NJ$$

and line $z$ has its justification changed to RuleP: $x, y$. Also, $\rho'(x)$ and $\rho'(y)$ are set equal to $\rho(z)$, and $R'_x := R_1$, $R'_y := R_2$. $\rho'$ agrees on all other sponsoring lines of $\mathcal{O}'$, and $R'_l := R_l$ for all active lines of $\mathcal{O}'$ other than $x$ and $y$. This application of P-Conj has reduced the subproof based on line $z$ to the two subproofs based on lines $x$ and $y$.

If the outline $\mathcal{O}$ contains a supporting line $a$ of the form

$$(a) \quad \mathcal{H} \quad \vdash \quad \forall x\, P \qquad\qquad RuleX$$

for some justification RuleX (other than NJ), then $R_a$ has the form $(\text{EXP}\ \ (t_1\ R_1)\ldots (t_n\ R_n))$. If any one of the terms $t_1, \ldots, t_n$ is admissible within $\mathcal{O}$, say $t_i$, then D-All can be applied to line $a$ by doing a universal instantiation of it with $t_i$. $L'$ is then equal to $L$ with the line $b$, shown below, inserted after line $a$.

$$(b) \quad \mathcal{H} \quad \vdash \quad \mathbf{S}^x_{t_i} P \qquad\qquad \forall I : a.$$

Here it is assumed that in this substitution, bound variables are systematically renamed to avoid variable capture. Also, $R'_b := R_i$. If $n \geq 2$ then line $a$ must remain active, so

$$R'_a := (\text{EXP}\ \ (t_1\ R_1)\ldots(t_{i-1}\ R_{i-1})\ (t_{i+1}\ R_{i+1})\ldots(t_n\ R_n))$$

and for each sponsoring line $z$ such that $a \in \rho(z)$, set $\rho'(z) := \rho(z) \cup \{b\}$ (*i.e.* $b$ is a cosupport with $a$). If $n = 1$, then line $a$ is no longer active so $b$ replaces $a$ as a support — that is, for each sponsoring line $z$ such that $a \in \rho(z)$, set $\rho'(z) := \rho(z) \setminus \{a\} \cup \{b\}$. In either case, $\rho'(z) := \rho(z)$ for all other sponsoring lines of $\mathcal{O}$ and $R'_l := R_l$ for all active lines $l \neq a$ of $\mathcal{O}$. It is straightforward to verify that $\mathcal{O}' = \langle L', \rho', \{R'_l\}\rangle$ is an outline.

It is possible to show that at least one expansion term associated with such active lines in $\mathcal{O}$ must be admissible, so requiring that the terms introduced in a universal instantiation (or introduced in a bottom-up fashion by P-Exists) be admissible is always possible to meet. This restriction to admissible terms is necessary to guarantee that

when variables are selected in the P-All and P-Choose transformations, they do not already have a free occurrence in the current proof outline.

A simple, naive process of transforming an ET-proof, represented by the list structure $R$, for the theorem $A$, would then start by successively applying either D- or P-transformations to the trivial outline for $A$ based on $R$ and finish when all the subproofs generated can be recognized as instances of the RuleP transformation.

## 6. Focused Construction of Proof Outlines

The proof outlines produced by the naive method described above will often turn out to be very inelegant for at last two reasons, which we will examine here. An implementation of this naive algorithm was made in the computer program TPS (see [15]) and it was frequently found that many of the supporting lines for a given sponsoring line were not really needed to prove that sponsoring line. The naive algorithm contained no way of checking for this since it was provided with no ability to "look ahead." Hence, may applications of D- and P- rules were not necessary and the resulting, completed natural deduction proofs were much longer and redundant than necessary. The naive algorithm was also not equipped to recognize when it could backchain on a supporting line which asserted an implication, since backchaining also requires looking ahead to see it if can actually be applied. Hence, the naive procedure always treated such implicational lines in the most general possible way — by using its equivalent disjunctive form in the form of an argument from cases. Implicational support lines were always used in a very unnatural fashion.

The information which would supply a transformation process with the necessary ability to look ahead is contained in a *mating* which is present in the tautology encoded in the ET-proofs of each subproof of a given outline. We now need several definitions.

**6.1. Definition.** If $\mathcal{A}_1$ and $\mathcal{A}_2$ are sets, define $\mathcal{A}_1 \Cup \mathcal{A}_2 := \{\xi_1 \cup \xi_2 \mid \xi_1 \in \mathcal{A}_1, \xi_2 \in \mathcal{A}_2\}$. Let $D$ be a $\lambda$-normal formula$_o$. We shall define two sets, $\mathcal{C}_D$ and $\mathcal{V}_D$, which are both sets of sets of b-atom subformula occurrences in $D$, by joint induction on the boolean structure of $D$. $\mathcal{C}_D$ is the set of clauses in $D$ while $\mathcal{V}_D$ is the set of "dual" clauses in $D$. Dual clauses have been called vertical paths by Andrews (see [5]).

(1) If $D$ is a b-atom, then $\mathcal{C}_D := \{\{D\}\}$ and $\mathcal{V}_D := \{\{D\}\}$.

(2) If $D = \sim D_1$ then $\mathcal{C}_D := \mathcal{V}_{D_1}$ and $\mathcal{V}_D := \mathcal{C}_{D_1}$.

(3) If $D = D_1 \vee D_2$ then $\mathcal{C}_D := \mathcal{C}_{D_1} \Cup \mathcal{C}_{D_2}$ and $\mathcal{V}_D := \mathcal{V}_{D_1} \cup \mathcal{V}_{D_2}$.

(4) If $D = D_1 \wedge D_2$ then $\mathcal{C}_D := \mathcal{C}_{D_1} \cup \mathcal{C}_{D_2}$ and $\mathcal{V}_D := \mathcal{V}_{D_1} \Cup \mathcal{V}_{D_2}$.

(5) If $D = D_1 \supset D_2$ then $\mathcal{C}_D := \mathcal{V}_{D_1} \Cup \mathcal{C}_{D_2}$ and $\mathcal{V}_D := \mathcal{C}_{D_1} \cup \mathcal{V}_{D_2}$. ∎

**6.2. Definition.** Let $D$ be a $\lambda$-normal formula$_o$. Let $\mathcal{M}$ be a set of unordered pairs, such that if $\{H, K\} \in \mathcal{M}$ and $H$ and $K$ are b-atom subformula occurrences in $D$, then $H$ and $K$ are contained in a common clause in $D$, $H$ conv-$I$ $K$, and either $H$ occurs positively and $K$ occurs negatively in $D$, or $H$ occurs negatively and $K$ occurs positively in $D$. Such a set $\mathcal{M}$ is called a *mating* for $D$. If $\{H, K\} \in \mathcal{M}$ we say that $H$ and $K$ are $\mathcal{M}$-*mated*, or simply *mated* if the mating can be determined from context. If it is also the case that for all $\xi \in \mathcal{C}_D$ there is a $\{H, K\} \in \mathcal{M}$ such that $\{H, K\} \subset \xi$, then we say

that $\mathcal{M}$ is a *clause-spanning mating* (cs-mating, for short) for $D$. In this case, we shall also say that $\mathcal{M}$ *spans* $D$. If $\mathcal{D}$ is a set of $\lambda$-normal formulas$_o$, we say that $\mathcal{M}$ is a mating (cs-mating) for $\mathcal{D}$ if $\mathcal{M}$ is a mating (cs-mating) for $\vee\mathcal{D}$. Here, the order by which the disjunction $\vee\mathcal{D}$ is constructed is taken to be arbitrary but fixed. ∎

The notion of a mating used by Andrews in [5] is a bit more general than the one we have defined here. In that paper, a mating, $\mathcal{M}$, is a set of ordered pairs, $\langle H, K\rangle$, such that there is a substitution $\theta$ which makes all such pairs complementary, *i.e.* $\theta K = \sim\theta H$. Except for this difference, the notion of a cs-mating corresponds very closely to his notion of a p-acceptable proof*-mating. Bibel in [7] also exploits matings for various theorem proving and metatheoretical application.

**6.3. Proposition.** *Let $D$ be in $\lambda$-normal form. $D$ is tautologous if and only if $D$ has a cs-mating.*

**6.4. Definition.** Let $\mathcal{D}$ be a finite, nonempty set of formulas$_o$, and let $\mathcal{M}$ be a mating for $\mathcal{D}$. With respect to $\mathcal{D}$ and $\mathcal{M}$, define $\approx^0$ to be the binary relation on $\mathcal{D}$ such that when $D_1, D_2 \in \mathcal{D}$, $D_1 \approx^0 D_2$ if $D_1$ contains a b-atom subformula occurrence $H$ and $D_2$ contains a b-atom subformula occurrence $K$ such that $\{H, K\} \in \mathcal{M}$. Let $\approx$ be the reflexive, transitive closure of $\approx^0$. Clearly $\approx$ is an equivalence relation on $\mathcal{D}$. If $D \in \mathcal{D}$, we shall write $[D]_\approx$ to denote the equivalence class (partition) of $\mathcal{D}$ which contains $D$. The following proposition is easily proved. ∎

**6.5. Proposition.** *Let $\mathcal{D}$ be a finite, nonempty set of formulas$_o$. If $\mathcal{M}$ is a cs-mating for $\mathcal{D}$ then $\mathcal{M}$ spans at least one of the $\approx$-partitions of $\mathcal{D}$. The converse is trivially true.*

**6.6. Definition.** Let $\mathcal{O} = \langle L, z, \{R_l\}\rangle$ be an outline. Let $D_l$ be the formula $Dp(Q_l)$ if $l$ is a sponsoring line or $Dp(\sim Q_l)$ if $l$ is a supporting line. Now define $\mathcal{D}_z := \{D_z\} \cup \{D_l \mid l \in \rho(z)\}$ if $z$ does not assert $\perp$ and $\mathcal{D}_z := \{D_l \mid l \in \rho(z)\}$, otherwise. Notice that for each $z \in L_0$, $Dp(Q_z) = \bigvee \mathcal{D}_z$. Now let $\mathcal{M}_z$ be a cs-mating for $Dp(Q_z)$ for each $z \in L_0$ and set $\mathcal{M} := \bigcup_{z \in L_0} \mathcal{M}_z$. $\mathcal{M}$ is called a *cs-mating for* $\mathcal{O}$. (Notice that $\mathcal{M}$ is also a cs-mating for each $Dp(Q_z)$.) We say that $\mathcal{O}$ is $\mathcal{M}$*-focused* if for each $z \in L_0$, $\mathcal{D}_z$ is composed of exactly one $\approx$-partition. ∎

**6.7. Example.** If $\mathcal{O}$ is the outline in Example 5.5, then

$$D_2 = \sim Pzy$$
$$D_3 = \sim[Pzy \supset Pz.c.Pz]$$
$$D_{16} = Pz.c.Pz$$
$$\vee\mathcal{D}_{16} = \sim Pzy \vee \sim[Pzy \supset Pz.c.Pz] \vee Pz.c.Pz]$$

Notice that $\mathcal{D}_{16}$ is tautologous. If we let $A1, A2, A3, A4$ represent the four b-atom occurrences in $\mathcal{D}_{16}$ then a cs-mating for $\mathcal{D}_{16}$ would be $\{(A1, A2), (A3, A4)\}$. ∎

Let $\mathcal{O} = \langle L, z, \{R_l\}\rangle$ be an outline and let $\mathcal{M}$ be a cs-mating for $\mathcal{O}$. If $\mathcal{O}$ is not $\mathcal{M}$-focused, then there must be a $z \in L_0$ such that $\mathcal{D}_z$ has too many members, *i.e.* there are at least two $\approx$-partitions of $\mathcal{D}_z$. What we need is a thinning outline transformation which will permit us to deactivate lines in $\mathcal{O}$, there by removing elements from $\mathcal{D}_z$. As long as the resulting $\mathcal{D}_z$ is still spanned by $\mathcal{M}$, the result of the thinning transformation will satisify the requirements of being an outline.

The thinning transformation works as follows. Let outline $\mathcal{O}$ and a cs-mating $\mathcal{M}$ for $\mathcal{O}$ be such that $\mathcal{O}$ is not $\mathcal{M}$-focused. Let $z$ be sponsoring line such that $\mathcal{D}_z$ contains

more than one $\approx$-partition. By Proposition 6.5, there is at least one $\approx$-partition $\mathcal{P} \subset \mathcal{D}_z$ such that $\mathcal{M}$ spans $\mathcal{P}$. Set $\mathcal{P}' := \mathcal{P} \setminus \mathcal{D}_z$. For each supporting line $l$ of $z$ such that $l \in \mathcal{P}'$, the thinning transformation modifies the value of $\rho(z)$ by removing $l$ from it. If it is the case that $D_z \in \mathcal{P}'$, then the supporting lines in $\mathcal{P}$ are strong enough to prove $\perp$, from which the assertion in line $z$ follows immediately. In this case, the thinning transformation must add the new sponsoring line

$$(y) \quad \mathcal{H} \vdash \perp \qquad\qquad NJ,$$

where $\mathcal{H}$ is the set of hypotheses for line $z$. The justification for line $z$ is changed to RuleP: $y$. The supports for line $y$ are those lines which were supporting line $z$ and were not thinned out as described above.

## 7. Backchaining

Using the mating information contained in the $Dp$-values of the expansion trees associated with each active line of an outline provides the outline transformation process with enough information to look ahead and identify unnecessary supporting and sponsoring lines. This same look ahead will help us determine when we should backchain on an implicational support line.

Consider the outline fragment

$$\begin{array}{llll} (a) & \mathcal{H} \vdash & \sim A_1 \vee A_2 & RuleX \\ (z) & \mathcal{H} \vdash & B & NJ \end{array} \qquad (\sigma)$$

where we have already determined that line $a$ is a necessary support of line $z$, and RuleX is the justification for line $a$. One way to use line $a$ in proving line $z$ is to apply P-Cases (see [4]) to the lines in $(\sigma)$, which would then yield the following lines.

$$\begin{array}{llll} (a) & \mathcal{H} \vdash & \sim A_1 \vee A_2 & RuleX \\ (b) & b \vdash & \sim A_1 & Hyp \\ (m) & \mathcal{H}, b \vdash & B & NJ \\ (n) & n \vdash & A_2 & Hyp \\ (y) & \mathcal{H}, n \vdash & B & NJ \\ (z) & \mathcal{H} \vdash & B & Cases : a, m, y \end{array} \qquad (\tau_1)$$

It may turn out that this new outline is no longer focused for at least two reasons. First, line $m$ may be proved indirectly from its sponsors, which now includes line $b$. In other words, $\mathcal{D}_m$ may contain a partition $\mathcal{P}$ such that $D_b \in \mathcal{P}$ but $D_m \notin \mathcal{P}$. Hence, $\sim A_1$ is used to prove $\perp$. The proof could, therefore, be reorganized so that we instead try to prove $A_1$ directly. In this case, we should apply the new D-ModusPonens transformation to the lines in $(\sigma)$ to yield the following lines.

$$\begin{array}{llll} (a) & \mathcal{H} \vdash & \sim A_1 \vee A_2 & RuleX \\ (m) & \mathcal{H} \vdash & A_1 & NJ \\ (n) & \mathcal{H} \vdash & A_2 & RuleP : a, m \\ (y) & \mathcal{H} \vdash & A_2 \supset B & NJ \\ (z) & \mathcal{H} \vdash & B & RuleP : y, n \end{array} \qquad (\tau_2)$$

Lines $m$ and $y$ are new sponsoring lines and they share the supports which $z$ had, less line $a$. Notice that $R_a$ has the form $(\vee \sim R_1\ R_2)$ for some list structures $R_1$ and $R_2$. In the new outline, we set $R'_m := R_1$ and $R'_y := (\supset R_2\ R_z)$. The new outline will be focused.

Another way the outline containing the lines in $(\tau_1)$ may not be focused is that line $y$ is proved indirectly from its supports. In this case, we need to backchain on the contrapositive form of line $a$, *i.e.* we should apply the new D-ModusTollens on the lines in $(\sigma)$ to yield the following lines.

$$\begin{array}{llll} (a) & \not\parallel & \vdash & \sim A_1 \vee A_2 \quad\quad RuleX \\ (m) & \not\parallel & \vdash & \sim A_2 \quad\quad NJ \\ (n) & \not\parallel & \vdash & \sim A_1 \quad\quad RuleP : a, m \\ (y) & \not\parallel & \vdash & \sim A_1 \supset B \quad\quad NJ \\ (z) & \not\parallel & \vdash & B \quad\quad RuleP : y, n \end{array} \qquad (\tau_3)$$

If in fact the outline containing the lines in $(\tau_1)$ was focused, then neither D-ModusPonens or D-ModusTollens could not be used on line $a$, and we actually needed to treat line $a$ as a disjunction by applying P-Cases. Of course, all these comments apply equally well when line $a$ asserts a formula of the form $A_1 \supset A_2$.

## 8. Other Forms of Natural Deduction

There are several different formats of proofs which have been called natural deduction, and, at first glance, the problems encountered in converting ET-proofs to these other proof formats might appear to be quite different than the problems encountered in building the Suppes-style proofs of the previous sections. This is generally not the case. For example, the transformation process already described produces, in a sense, proofs in Gentzen's LK format [13]. For each sponsoring line $z$ in a given outline, consider the sequent, $\rho(z) \rightarrow z$, where line labels are used to refer to their assertions. Hence, to each outline there corresponds a set of sequents which represent the unfinished subproofs of that outline. The D- and P- transformations can then be seen as ways of taking the sequents of one outline and replacing some of them with logicially simpler sequents. These simpler sequents can then be joined using derived rules of the LK-calculus to yield the sequents they replace. In this fashion, an entire LK derivation can be built. Of course, for this to work in higher-order logic, we would need to add an inference rule for $\lambda$-conversion, but this is the only essential addition needed for this accommodation. LK derivations built in this fashion will contain no instances of the cut inference rule. Thus, by using our relative completeness result for ET-proofs, if $A$ is a theorem of $\mathcal{T}$, $A$ has an ET-proof which can be converted to a cut-free LK derivation. Via the transformation process, our version of Herbrand's theorem can thus be used to prove Gentzen's *Hauptsatz*. See [16] for a complete account of how ET-proofs can be converted to LK deriviations.

## 9. Acknowledgements

I would like to thank Peter Andrews and Frank Pfenning for many valuable comments concerning this paper and the work reported in it.

## 10. Bibliography


[1] Peter B. Andrews, "Resolution in Type Theory," Journal of Symbolic Logic **36** (1971), 414–432.

[2] Peter B. Andrews, "Provability in Elementary Type Theory," Zeitschrift für Mathematische Logik und Grundlagen der Mathematik **20** (1974), 411–418.

[3] Peter B. Andrews and Eve Longini Cohen, "Theorem Proving in Type Theory," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* 1977, 566.

[4] Peter B. Andrews, "Transforming Matings into Natural Deduction Proofs," *Fifth Conference on Automated Deduction, Les Arcs, France*, edited by W. Bibel and R. Kowalski, Lecture Notes in Computer Science, No. 87, Springer-Verlag, 1980, 281–292.

[5] Peter B. Andrews, "Theorem Proving Via General Matings," Journal of the Association for Computing Machinery **28** (1981), 193–214.

[6] Maria Virginia Aponte, José Alberto Fernández, and Philippe Roussel, "Editing First-order Proofs: Programmed Rules vs. Derived Rules," *Proceedings of the 1984 International Symposium on Logic Programming*, 92–97.

[7] Wolfgang Bibel, "Matrices with Connections," Journal of the Association of Computing Machinery **28** (1981), 633–645.

[8] W. W. Bledsoe, "A Maximal Method for Set Variables in Automatic Theorem-proving," in *Machine Intelligence 9*, edited by J. E. Hayes, Donald Michie, and L. I. Mikulich, Ellis Horwood Ltd., 1979, 53–100.

[9] W. W. Bledsoe, "Using Examples to Generate Instantiations for Set Variables," University of Texas at Austin Technical Report ATP-67, July 1982.

[10] Alonzo Church, "A Formulation of the Simple Theory of Types," Journal of Symbolic Logic **5** (1940), 56–68.

[11] Gérard P. Huet, "A Mechanization of Type Theory," *Proceedings of the Third International Joint Conference on Artificial Intelligence* 1973, 139–146.

[12] Gérard P. Huet, "A Unification Algorithm for Typed $\lambda$-calculus," Theoretical Computer Science **1** (1975), 27–57.

[13] Gerhard Gentzen, "Investigations into Logical Deductions," in *The Collected Papers of Gerhard Gentzen*, edited by M. E. Szabo, North-Holland Publishing Co., Amsterdam, 1969, 68–131.

[14] D. C. Jensen and T. Pietrzykowski, "Mechanizing $\omega$-Order Type Theory Through Unification," Theoretical Computer Science **3** (1976), 123–171.

[15] Dale A. Miller, Eve Longini Cohen, and Peter B. Andrews, "A Look at TPS," *6th Conference on Automated Deduction, New York*, edited by Donald W. Loveland, Lecture Notes in Computere and Science, No. 138, Springer-Verlag, 1982, 50–69.

[16] Dale A. Miller, "Proofs in Higher-order Logic," Ph. D. Dissertation, Carnegie-Mellon University, August 1983. Available as Technical Report MS-CIS-83-37

from the Department of Computer and Information Science, University of Pennsylvania.

[17] Frank Pfenning, "Analytic and Non-analytic Proofs," elsewhere in these proceedings.

[18] T. Pietrzykowski and D. C. Jensen, "A complete mechanization of $\omega$-order type theory," *Proceedings of the ACM Annual Conference, Volume I*, 1972, 82–92.

[19] Tomasz Pietrzykowski, "A Complete Mechanization of Second-Order Type Theory," Journal of the Association for Computing Machinery **20** (1973), 333–364.

[20] J. A. Robinson, "Mechanizing Higher-Order Logic," *Machine Intelligence 4*, Edinburgh University Press, 1969, 151–170.

[21] Patrick Suppes, *Introduction to Logic*, D. Van Nostrand Company Ltd., Princeton, 1957.

[22] Moto-o-Takahashi, "A proof of cut-elimination theorem in simple type-theory," Journal of the Mathematical Society of Japan **19** (1967), 399–410.

[23] Alfred Tarski, "A Lattice-theoretical Fixpoint Theorem and Its Applications," Pacific Journal of Mathematics **5** (1955), 285–309.