

Appearing in the Proceedings of the 1986 National Conference on Artificial Intelligence.

DARPA N000-14-85-K-0018.

This work has been supported by NSF grants MCS8219196-CER, MCS-82-07294, and

natural deduction is often the core of interactive theorem provers. would have removed. Thus, resolution is often the core of automatic theorem provers while unimportant features in the search space which the preprocessing done by normal forms moment is easily understood. On the other hand, natural deduction often leaves too many very easy to involve a user in the search for a proof since the state of the search at any formulas of the proposed theorem are used during the search for a proof. As a result, it is example, no normal forms are generally used and only subformulas or instances of subfor-process. On these accounts, natural deduction theorem proving is just the opposite. For remote from a user's original input, it is difficult to get the user to interact with the search hand, since the search in such a theorem prover is carried out in a space which is rather and automatic theorem provers using this paradigm are rather easy to build. On the other a result of using such normal forms, the search space of refutations is very homogeneous, theorem and putting its negation into skolem normal and conjunctive normal form. As characteristics. For example, a search for a resolution refutation starts by taking a proposed Theorem provers built on resolution and natural deduction have very different char-

1. Introduction

use in the development of user oriented explanation facilities. automatically construct a complete natural deduction proof. Such a capability can be of is possible to write a tactic which can take the type specified by a resolution refutation and several capabilities not generally found in other theorem proving systems. For example, it Explicit representation of proofs as typed values within a programming language provides used to program different interactive and automatic natural deduction theorem provers. also be extended to incorporate proofs as typed λ -terms. Such extended tacticals can be formulas-as-type notion found in proof theory. The LCF notion of tactics and tacticals can are attached to quantifiers. As such, this approach to proofs and their types extends the type structures, called *expansion trees*, are essentially formulas in which substitution terms deduction proofs as λ -terms and resolution refutations as the types of such λ -terms. These proving technologies as resolution and natural deduction. This system represents natural

Abstract: We present a high-level approach to the integration of such different theorem

Philadelphia, PA 19104

University of Pennsylvania

Computer and Information Science

Dale Miller and Amy Felty

Theorem Proving

Resolution and Natural Deduction

An Integration of

$$\begin{array}{c}
 \frac{\Gamma \rightarrow \Theta, A \quad \Gamma \rightarrow \Theta, C}{\Gamma \rightarrow \Theta, A \wedge C} \text{and-1} \quad \frac{\Gamma \rightarrow \Theta, A \quad \Gamma \rightarrow \Theta, C}{\Gamma \rightarrow \Theta, A \vee C} \text{or-1} \\
 \frac{\Gamma \rightarrow \Theta, A \quad \Gamma \rightarrow \Theta, \sim A}{\Gamma \rightarrow \Theta} \text{not-1} \quad \frac{\Gamma \rightarrow \Theta, A \quad \Gamma \rightarrow \Theta, \sim A}{\Gamma \rightarrow \Theta} \text{not-r} \\
 \frac{\Gamma \rightarrow \Theta, A \quad \Gamma \rightarrow \Theta, \Delta, \Gamma, \Delta}{\Gamma \rightarrow \Theta, A} \text{imp-1} \quad \frac{\Gamma \rightarrow \Theta, A \quad \Gamma \rightarrow \Theta, C}{\Gamma \rightarrow \Theta, A \supset C} \text{imp-r}
 \end{array}$$

Although much of what we describe here is applicable to most forms of natural deduction, the form we present in this paper is essentially the sequent system LK presented in [Gentzen, 1935] but without the cut rule. More modern presentations of similar systems can be found in [Gallier, 1986] and [Prawitz, 1965]. Proofs in the LK system are finite, ordered trees in which nodes are labeled with *sequents*. A sequent, written as $\Gamma \rightarrow \Theta$, will represent the proposition "from all the formulas in the set Γ , some formula in the set Θ can be proved." Notice that the proposition connected to the sequent $A \rightarrow A$ is trivially true. Sequents of this simple kind are called *axioms*. The non-terminal nodes of an LK proof are called inference rules and are listed below.

2. Natural Deduction Proofs

Clearly it is desirable to find some way to smoothly integrate these two very different paradigms. In this paper, we propose just such an integration. This integration is not a merging of the two different search spaces. It is, instead, an integration of the two kinds of proofs. We shall present a system which explicitly represents proofs in both systems and is capable of translating between them. In order to achieve this goal, we have designed a *programming language* which permits proof structures as values and types. This approach builds on and extends the LCF approach to natural deduction theorem provers by replacing the LCF notion of a *validation* with explicit term representation of proofs. The terms which represent proofs are given types which generalize the formulas-as-type notion found in proof theory [Howard, 1969]. Resolution refutations are seen as specifying the *type* of a natural deduction proofs. This high level view of proofs as typed terms can be easily combined with more standard aspects of LCF to yield the integration for which we are looking.

In Section 2 we describe a representation of natural deduction proofs as λ -terms, and in Section 3 we show how the LCF notion of tactics and tacticals can be used to specify an interactive theorem prover based on such a term representation of natural deduction proofs. In Section 4 we describe how resolution refutations can be converted to generalized type structures called *expansion trees*. In Section 5 we show how tactics can make use of the information stored in these generalized types. Also in Section 5, we present a program in the language of tactics which is capable of automatically converting a resolution refutation to a natural deduction proof.

where T'_1 and T'_2 are the terms representing the proofs T_1 and T_2 , respectively. Many inference rules require more information than just subproofs in order to put those subproofs together into larger proofs. For example, a term representing a proof which contains any of the quantifier introduction rules must contain the substitution term used to instantiate the quantifiers. Although such information is necessary, we avoid presenting it in this paper to simplify the presentation of examples.

$$\frac{T_1 \quad T_2}{\Gamma, A \vee B \rightarrow \Theta} \text{or-1}$$

These proof trees can be represented more manageably as term structures. For example, let $\text{axiom}(A)$ represent the proof tree which contains just the sequent $A \rightarrow A$. The inference rules can be represented by function symbols of 1 or 2 arguments. For example, if T_1 and T_2 are proofs of $\Gamma, A \rightarrow \Theta$ and $\Gamma, B \rightarrow \Theta$, respectively, we would write $\text{or-1}(T'_1, T'_2)$ to represent the proof

$$\frac{\frac{\frac{d(a) \rightarrow d(a)}{\text{some-r}} \quad \frac{d(a) \rightarrow \exists x b(x)}{\text{imp-1}}}{d(a), d(a) \rightarrow \exists x b(x)} \quad \frac{\frac{d(a), d(a) \rightarrow \exists x b(x)}{\text{all-1}} \quad \frac{d(a), d(a) \rightarrow \exists x b(x)}{\text{some-r}}}{d(a), \exists x b(x) \rightarrow \exists x b(x)} \quad \frac{\frac{d(a), \exists x b(x) \rightarrow \exists x b(x)}{\text{thin-1}} \quad \frac{d(a), \exists x b(x) \rightarrow \exists x b(x)}{\text{or-1}}}{d(a), \exists x b(x) \rightarrow \exists x b(x)}$$

Figure 1.

$$d(a) \vee q(b) \rightarrow [\exists x q(x) \vee \forall x d(x)]$$

Example 1. Figure 1 is an LK proof of the formula

axioms.

All but the last two rules are *introduction rules* and are responsible for introducing into sequents the various logical connectives. The proviso that the variable y is not free in any formula of the lower sequent must be added to the rules all-1 and some-1 . A derivation tree is an *LK proof* of A if the root of the tree is the sequent $\rightarrow A$ and its leaves are

$$\frac{\Gamma \rightarrow \Theta \quad \Gamma \rightarrow \Theta}{\Gamma \rightarrow \Theta} \text{thin-1} \quad \frac{\Gamma \rightarrow \Theta, A \quad \Gamma \rightarrow \Theta, A}{\Gamma \rightarrow \Theta, A} \text{thin-r}$$

$$\frac{\Gamma \rightarrow \Theta, \exists x P \quad \Gamma \rightarrow \Theta, \exists x P}{\Gamma \rightarrow \Theta, \exists x P} \text{some-1} \quad \frac{\Gamma \rightarrow \Theta, \exists x P \quad \Gamma \rightarrow \Theta, \exists x P}{\Gamma \rightarrow \Theta, \exists x P} \text{some-r}$$

$$\frac{\Gamma \rightarrow \Theta, \forall x P \quad \Gamma \rightarrow \Theta, \forall x P}{\Gamma \rightarrow \Theta, \forall x P} \text{all-1} \quad \frac{\Gamma \rightarrow \Theta, \forall x P \quad \Gamma \rightarrow \Theta, \forall x P}{\Gamma \rightarrow \Theta, \forall x P} \text{all-r}$$

Remember that the typed λ -calculus has the following restriction on application: a term g can be applied to a term h if and only if the type of g is of the form $\alpha \rightarrow \beta$ and the type of h is of the form α . This restriction thus enforces the restriction of combining partial

$$\frac{d(a), \forall x [d(x) \supset q(x)] \rightarrow \exists x q(x) \quad \exists x [d(x) \supset q(x)] \rightarrow \exists x q(x)}{d(a), \forall x [d(x) \supset q(x)] \rightarrow \exists x q(x)}$$

In order for the mechanism of λ -conversion to correctly represent the operation of supplying a partial proof with a subproof, we must *type* these λ -terms. For example, $\lambda x \lambda y T(x, y)$ represents a partial proof of some sequent in which two subproofs must be supplied. However, before this term can be applied to some actual proof, say S , one must check that the abstracted variable x is a place holder for proofs of the sequent for which S is a proof. Thus, we should make sequents and functions among sequents be the types of λ -terms. For example, if x and y are place holders for proofs of the sequents σ_1 and σ_2 , respectively, and if $\lambda x \lambda y T(x, y)$ is a partial proof of the sequent σ , then we attach to this λ -term the type $\sigma_1 \rightarrow \sigma_2 \rightarrow \sigma$. The type of the λ -term representing the partial proof in Figure 2 is, therefore,

Figure 2.

$$\frac{\frac{\frac{d(a) \vee q(b), \forall x [d(x) \supset q(x)] \rightarrow \exists x q(x)}{\text{and-1}} \quad \frac{d(a) \vee q(b)}{\text{or-1}}}{\text{imp-r}}}{\text{thin-1}} \quad \frac{d(a), \forall x [d(x) \supset q(x)] \rightarrow \exists x q(x)}{\text{thin-1}}$$

λ lambda Y. imp-r (and-1 (or-1 (X, thin-1 (Y))))

term
Example 3. A partial proof of the formula in Example 1 is given in Figure 2 and by the subproofs to a completed proof.

To build interactive proof systems, it is important to represent not only completed proofs but also incomplete or partial proofs. We represent these by introducing into proof terms free variables which act as place holders for the actual subproofs. These free variables are also abstracted with λ -bindings. Thus a partial proof is represented as a function from

$$\text{some-r (axiom (q(a)))},$$

$$\text{imp-r (and-1 (all-1 (imp-1 (axiom (p(a))),$$

Example 2. The (simplified) term which represents the proof in Example 1 is written as:

