null

null
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, using a notion found in proof theory [Howard, 1986]. Normal forms are described by typed terms, and the language of tactics and tactics as types with the type of each proof being specified by a type of the term. In Section 2, we describe the tactics of the LCF approach, which represent proofs as typed terms, for which the tactics are looking.

In Section 3, we show how the LCF notations of tactics and tactics as types can be used to specify an interactive theorem prover, thus giving a representation of natural deduction proofs as typed terms, for which the tactics are looking.

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 proofs. Also in Section 5, we present a program in the language of tactics, which is able to automatically convert a resolution refutation to a natural deduction proof.

2. Natural Deduction Proofs

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 [Genzien, 1993], but without the cut rule. More modern presentations of similar systems can be found in [Galier, 1993] and [Prawitz, 1965]. Proofs in the LK system are finite, ordered trees in which nodes are labeled with sequents. A sequent, written as \( \Gamma \rightarrow \Delta \), will represent the proposition connected to the sequent. A sequent \( A \rightarrow \Delta \) is trivially true. Sequents of this kind are called axioms. The non-terminal nodes of an LK proof are called inference rules and are listed below.

\[
\begin{align*}
\Gamma & \rightarrow \theta, A \rightarrow \Delta \\
\Gamma & \rightarrow \theta, A \lor C \\
\Gamma & \rightarrow \theta, A \rightarrow \Delta, \lor \neg C \\
A \lor C & \rightarrow \theta, \lor \neg C \\
\neg A & \rightarrow \theta, \lor \neg C \\
\lor \neg A & \rightarrow \theta, \lor \neg C \\
A & \rightarrow \theta, \lor \neg C \\
A & \rightarrow \theta, A \\
\end{align*}
\]
simplify the presentation of examples.

The quantifier introduction rules must contain the substitution term used to instantiate the quantifiers. Although such information is necessarily we avoid presenting it in this paper to infer those terms. For example, a term representing a proof which contains any of the quantified introduction rules can be represented by function symbols for quantifiers. For example,

\[ \Theta \leftarrow B \land \forall x \; L \]

\[ \Theta \leftarrow B \land \forall x \; L \]

where \( L \) and \( L \) are the terms representing the proofs \( L \) and \( L \), respectively.

\( \exists x \).

\( \exists x \).

\[ (x)b \; \exists E \subset [(x)b \subset (x)d] \; \exists A \lor [(q)b \land (v)d] \]

\[ (x)b \; \exists E \subset [(x)b \subset (x)d] \; \exists A \lor [(q)b \land (v)d] \]

or \( \exists x \).

or \( \exists x \).

\[ (x)b \; \exists E \subset [(x)b \subset (x)d] \; \exists A \lor [(q)b \land (v)d] \]

\[ (x)b \; \exists E \subset [(x)b \subset (x)d] \; \exists A \lor [(q)b \land (v)d] \]

These proof trees can be represented more meaningfully as term structures. For example,

\[ (x)b \; \exists E \subset [(x)b \subset (x)d] \; \exists A \lor [(q)b \land (v)d] \]

\[ (x)b \; \exists E \subset [(x)b \subset (x)d] \; \exists A \lor [(q)b \land (v)d] \]

Figure 1.

\[ (z)b \; \exists E \subset [(z)b \subset (z)d] \; \exists A \lor [(q)b \land (v)d] \]

\[ (z)b \; \exists E \subset [(z)b \subset (z)d] \; \exists A \lor [(q)b \land (v)d] \]

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

\[ (z)b \; \exists E \subset [(z)b \subset (z)d] \; \exists A \lor [(q)b \land (v)d] \]

\[ (z)b \; \exists E \subset [(z)b \subset (z)d] \; \exists A \lor [(q)b \land (v)d] \]

Exams.

\[ (z)b \; \exists E \subset [(z)b \subset (z)d] \; \exists A \lor [(q)b \land (v)d] \]

\[ (z)b \; \exists E \subset [(z)b \subset (z)d] \; \exists A \lor [(q)b \land (v)d] \]

But the last two are introduction rules and are responsible for introducing the
of $x$ is the form $a$. Thus, restriction thus enforces the restriction of combining partials

Remember that the typed $\wedge$-calculus has the following restriction on application: a term $a$ can be applied to a term $b$ if and only if the type of $a$ is the form $x$ or the form $y$ and the type of $b$ is a term $z$.

\[(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land [(q)b \land (v)d] \subseteq \]
\[\subseteq [(x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]

\[\subseteq (x)b \ xz \subseteq (q)b \subseteq [(x)b \ xz \subseteq [(x)b \in (x)d] \ xz \land (v)d] \subseteq \]
\[
\begin{align*}
\lambda b. ((x) b \in (x) d) & \iff (x) b \in (x) d \quad \lambda b. (v) d \\
T-\text{I} & \quad (x) b \in (x) d \\
T-\text{IMP} & \quad (v) b \in (v) d
\end{align*}
\]

This is a more refined partial proof of \( \theta \).

Example 4. Suppose that some combination of tactics requires the following partial proof:

\[
0_0 \leftarrow u_0 \leftarrow \cdots \leftarrow \ldots \leftarrow I \leftarrow \cdot^{g \rightarrow} \cdot \leftarrow \cdot
\]

Two partial proofs into a single one of type \( \Theta \).

This is a partial proof with missing subproofs. The trace that combines these

\[
\begin{align*}
\lambda b. (x) b \in (x) d & \iff (x) b \in (x) d \quad \lambda b. (v) d \\
T-\text{I} & \quad (x) b \in (x) d \\
T-\text{IMP} & \quad (v) b \in (v) d
\end{align*}
\]

Combination tactics are built from primitive tactics by using tactics. As in ITP, the

\[
\begin{align*}
\lambda b. (x) b \in (x) d & \iff (x) b \in (x) d \\
T-\text{I} & \quad (x) b \in (x) d \\
T-\text{IMP} & \quad (v) b \in (v) d
\end{align*}
\]

Lambda \( \lambda b. (x) b \in (x) d \).

Lambda \( \lambda b. (x) b \in (x) d \).

Some tactics of combination of tactics require a partial proof of the type \( \Theta \).

\[
\begin{align*}
\lambda b. (x) b \in (x) d & \iff (x) b \in (x) d \\
T-\text{I} & \quad (x) b \in (x) d \\
T-\text{IMP} & \quad (v) b \in (v) d
\end{align*}
\]

Precedence between tactics is a description of one step of the proof and represents

\[
\begin{align*}
\lambda b. (x) b \in (x) d & \iff (x) b \in (x) d \\
T-\text{I} & \quad (x) b \in (x) d \\
T-\text{IMP} & \quad (v) b \in (v) d
\end{align*}
\]

This term is typed as

\[
\begin{align*}
\lambda b. (x) b \in (x) d & \iff (x) b \in (x) d \\
T-\text{I} & \quad (x) b \in (x) d \\
T-\text{IMP} & \quad (v) b \in (v) d
\end{align*}
\]

Some of the form \( \lambda b. (x) b \in (x) d \).

In ITP, tactics are described as they are discovered.
When this term is combined with the partial proof in Example 2, the combined proof can be written as:

\[
\lambda x. \forall y. (\text{imp}(\text{and}(x,y), \text{all}(\text{imp}(\text{ax}(x,y), \text{Z})), \text{thn}(\text{th}(y))))
\]

and is of type

\[
\exists x \cdot p(x) \equiv (\exists x \cdot p(x)) \land (\forall y \cdot q(y) \lor \forall z \cdot p(z) \supset q(z))
\]

Although the number of abstracted variables (i.e., the number of subproofs) may grow in size as we combine partial proofs, the amount of work that still must be completed generally decreases because as each rule is applied, the resulting sequent(s) generally contain fewer connectives. The number of subproofs decreases when one of them is recognized as an axiom.

In general, there are many terms (proofs) of a given type (sequent). Thus, many choices can be made at each step in building a proof, and different choices can result in different proofs. These choices fall into two categories. The first choice at any given point in processing a partial proof is which abstracted variable (i.e., which subproof) should be analyzed. The second choice is which tactic to use in filling in this subproof. Tactics allow the programmer to specify the order in which tactics are attempted. For example, if we want to prove a set of theorems, we know all the rules and apply all non-branching propositional rules before continuing in interactive mode. A procedure to do this can be written as follows:

\[
\text{repeat (\text{imp-\text{neg-1-tac}}) (\text{and-\text{neg-1-tac}}) (\text{repeat (\text{imp-\text{neg-1-tac}}) (\text{neg-1-tac})})}
\]

where then, repeat, and orelse are names of high level tacticals similar to those found in LCF. By writing compound tacticals, the programmer can directly involve in how choices are made during the search for a proof. The ability to express proof search strategies as small programs allows great flexibility in customizing search functions to determine which inference rules can be applied. We, however, have not discussed what happens when a top-level quantifier is encountered. The quantifier is not given by the sequent, and so the sequent by itself does not contain enough type information to adequately specify a proof. This type information is much harder to determine, and we turn to an automatic theorem prover, such as resolution, for help.
Theorem 5. The following is a resolution of the formula in Example 1:

\[ \exists x \exists y (P(x,y) \land Q(x,y)) \]

Example 6. The following is a resolution of the formula in Example 2:

\[ \forall x \exists y (P(x,y) \land Q(x,y)) \]

Proof. To this end, we define expansion trees and the expansion trees in the following:

1. **Clause Expansion:**
   - From \( \exists x \exists y (P(x,y) \land Q(x,y)) \), we get:
     - \( \exists x \) (6)
     - \( \exists y \) (7)
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (8)
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (9)
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (10)
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (11)

2. **Universal Instantiation:**
   - From the above clause, we get:
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (12)
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (13)
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (14)
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (15)
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (16)

3. **Resolution:**
   - From the above clauses, we get:
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (17)
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (18)
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (19)
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (20)
     - \( \exists x \exists y (P(x,y) \land Q(x,y)) \) (21)

4. **Resolution Trees:**
   - The resolution process continues,...
Given a formula \( \phi \), there are, in general, many possible natural deduction proofs which could have been constructed by a compound logical form of the natural deduction proof of \( \phi \). This can naturally be converted to a compound logical form of the natural deduction proof of \( \phi \).

In Example 6, let \( \phi \) be a formula. Then \( \phi \) is both an expansion tree and dual expansion tree for \( \phi \).
6. Conclusions

Note on which mark types in this fashion.

An example of a tree with expansion rules. These can be used to store complete

Expansion trees can be more natural by compounding nodes.

Examples of trees provide the system with some capabilities not present

The search is concerned not with the existence of a proof but with the presentation of the

Types of trees, their theory, so such a conversion involves some searching.
References

7. References


8. W. A. Howland, "The Foundation of a Type of a Natural Deduction Proof."


9 (1983), 223–244.


