A Framework for Interactive Sharing and Deductive Searching in Distributed Heterogeneous Collections of Formalized Mathematics

James L. Caldwell[1] * and Christoph Jechlitschek[2] **

[1] Department of Computer Science, University of Wyoming, Laramie, WY
[2] Department of Computer Science, Washington University, St. Louis, MO.

Abstract. Peer-to-peer technology implemented in systems like Napster allowed sharing of digitized music across the web in an incredibly easy to use system. This paper describes a prototype peer-to-peer system for networking distributed and heterogeneous databases of formalized mathematics. We also propose a general framework for deductive search in heterogeneous libraries of formal content. As participants in this conference well know, a significant body of mathematics has been formalized in theorem provers. We believe that a truly distributed mechanism for sharing formal content will multiply efforts of individual users of theorem proving systems, will invigorate ongoing formalization efforts, and will spur new research in deductive search and content-based addressing. Interactive sharing has the potential to be a significant new methodology for theorem proving. A basic tenet of our approach is that users of the system must be able to account for results and methods for accountability are incorporated into the proposed methods.

1 Introduction

We imagine a future in which the web plays an integral role in theorem proving efforts. Where theorems and proofs of diverse systems are interactively searched by developers across the web and where sharing is used to discharge significant numbers of proof obligations.

There is a diverse array of theorem proving systems representing many hundreds of man-years of effort, they range from those that completely automate the proof search to interactive proof checkers. A list of these systems would includes ACL2, Coq, Elf, HOL, Isabelle, MetaPrl, Mizar, Nuprl, PVS and others. The number of extant theorems that have been proved in these systems is astounding. (The reader is invited to make his own estimate). Currently, the costs

* The first authors work was partially supported by NSF grant CCR-9985239.
** This work was performed at the University of Wyoming and was supported by a DoD Multidisciplinary University Research Initiative (MURI) program administered by the Office of Naval Research under grant N00014-01-1-0765.

of sharing formal material are so high that little sharing takes place, sometimes even within communities of users of the same tool. Much of the sharing that does take place requires personal communication between the parties involved.

In this paper we describe a vision of the future and argue that it need not merely be a fantasy. Toward this end we describe the implementation of a prototype peer-to-peer framework for connecting distributed libraries of mathematics [17]. The prototype implementation also supports a query language for content and name based searching. We go further in proposing a general framework for deductive search methods in a collection of logically heterogeneous databases[1]. Our approach is guided by our experiences in the Formal Digital Libraries project [8], a joint project between Cornell, Caltech and Wyoming.

1.1 Vision

We imagine a system in which proof obligations may be discharged by existing results, that have perhaps been verified in different logics, and recorded in a distributed and heterogeneous database. Consider the following scenario.

> A user, sitting in Laramie Wyoming on a blustery winter evening, is constructing a proof in the Nuprl system[2]. At various points in the process, suspecting that surely someone else has already proved the result required to complete her proof, she initiates an interactive query. Moments before the Wyoming user issued her query, an early rising HOL user in the warm summer morning of Canberra Australia has just proved a lemma having the required semantic import. Upon completing his proof it was automatically committed to his local online database. Our Wyoming users query returns the HOL result together with a procedure for translating HOL terms into Nuprl terms and includes evidence that the proof actually was completed in HOL. This information is incorporated into her local database and used to complete her proof. Once completed, her new result is recorded into her data-base thereby making it immediately available to other proof efforts distributed across the web.

In this paper we argue that this scenario is both theoretically feasible and practically realizable with existing technologies (circa 2004). We present evidence for this argument by describing a prototype implementation of a peer-to-peer network for interconnecting databases of formal mathematical content. We continue by outlining a general theoretical framework for deductive searching in distributed networks libraries of heterogeneous formalized mathematics.

We reckon that the following are the necessary components for such a system.

[1] Throughout the paper, the words "library" and "database" should be considered synonymous, though perhaps the word "database" emphasizes implementation.

[2] By inclination, our hypothetical user is interested in extracting programs from proofs but has no philosophical objections to incorporating classical results into her proofs if it does not impinge on the constructive content. For a discussion of just such a methodology for incorporating classical results into constructive proofs see [5].

i.) Individual databases of formalized mathematics.
ii.) A framework for connecting individual databases into a distributed network including methods for finding databases of formal material and a protocol for communicating between them.
iii.) Methods of translating between logical theories.
iv.) Methods of searching across the distributed network.

There are independent research efforts underway on all these topics. What does not currently exist is an effort to pull these technologies together into an unified approach. In this paper we address items (ii), (iii) and (iv). We do not propose to constrain (i) other than to require that databases participating in the network implement the protocol described as part of (ii). We believe that a successful implementation of items (i), (ii) and (iv) will create a *market* that will further stimulate the development of translations (iii).

We remark that, perhaps surprisingly, scientific communities other than computer science seem to be better at sharing results in a significant way; by which we mean that information is shared so that it can be used directly in establishing new results, not simply in a secondary form. For example, the databases of genetic structures are remotely accessible and remote access forms a crucial part of the methodology used by researchers working in that field. On the homepage for the National Center for Biotechnology Information [22] it says: "Most journals now expect that DNA and amino acid sequences that appear in articles will be submitted to a sequence database before publication." As a community, we could take a lesson here.

1.2 Relating theories

The ability to soundly combine theorems proved in different logics within the same framework is a deep mathematical problem. Institutions [10, 11] provide a category theoretic framework in which the formal relations between different theories can be established. Although institutions provide a mathematical framework within which relations between logics can be understood, they have been little used in practice. The hard part of relating theories is establishing the semantic map. Howe [15, 14] has provided the semantic foundations for a map between HOL and a classical variant of Nuprl. An implementation of the translation is described in [21]. Naumov [20] has related Isabelle and classical Nuprl and a semantic justification for translating PVS results into classical Nuprl has recently been completed [19]. Staples has related ACL2 and HOL [25] providing a mechanism to incorporate results of ACL2 into HOL proofs. Applications for sharing results (even the use of classical results from PVS in constructive Nuprl proofs) are discussed in [5]. In each case, a semantically justified translation from the language of one logic into the language of another is required.

Applying an economic model, we note that translations between theories are implemented by individuals who value the incorporation of results from one theory into their own highly enough to do the required work. Part of the calculation of the worth of such an effort is based on the amount of and type of material that will be made accessible by a translation and the ease with which it will be used by the developer and others. The framework proposed here lessens the effort required to apply such translations, *i.e.* it provides a market for such tools. By providing a such a market, we believe that a framework such as the one described here for integrating multiple provers will motivate further developments.

1.3 Accountability

A guiding principle of our framework and of the Formal Digital Library [8] (FDL) is one of accountability. Consumers of theorems and other formal objects have a right to know what assumptions, facts and methods an object depends on; this problem has seen previous study [12]. Only with this knowledge can users make epistemic judgments whether to accept results and to incorporate them into their own work. As part of the FDL effort, Allen [2] has designed a novel mechanism to certify facts about objects in a database of terms. These certifications carry epistemic weight in that: users may create new *certificate kinds*, they may request than an existing kind of certification be run on a particular object, or they may examine existing certificates. Users may not create certification objects, only the system can do so by evaluating the computational part of a certificate kind. Users can determine exactly what has been certified by examining the code used to create a certificate. In the scheme of the FDL, there are a plethora of certificates generated by many users; some may be as simple as a claim that some individual created the certified object, others may certify that a proof has been accepted by some formal tool or that some object originated from a particular database. This certification mechanism can be used to build sets of dependencies and properties of objects and to track them. Users can inspect certificates and, by evaluating the methods used to generate a particular certificate kind, can determine the epistemic weight they accord to the certified object.

Accounting for the correctness of a formal object (let's say a proof) depends on a complex set of facts that at least include which tools (and version) were used to produce the proof; the lemmas, tactics, and methods of proof the theorem itself depends on; global settings in the environment when the proof was done; and perhaps other facts. This list must be open-ended since the evidence required for an individual to accept a result ultimately depends on that individuals personal criteria. The criteria for believing something can vary from individual to individual and thus, the threshold of evidence may be higher or lower, depending on the individual. In an extreme case, users may accept results based on authority *e.g.* *'Caldwell said "Constable said ϕ is a theorem.".'* But even this form of evidence[3] may carry epistemic weight with users and our goal is to include even this kind of evidence. Every kind of formal object potentially requires some form of evidence (formal or informal) to justify its use in certain contexts.

[3] Evidence like this may actually be easy to account for using certificates based on digital signatures.

1.4 Searching

We intend to search in heterogeneous databases, *i.e.* databases containing results from a number of logical systems. The effectiveness of existing search technologies would seem to be the principal technical obstacle to true integration of these ideas into proof engines.

There are two aspects to the search problem. The first is to find the available databases of formal content on the web that are open to pubic search; the second is to search those databases for formal content (definitions, theorems, proofs, translations, tactics, *etc.*) in a semantically robust way. The first problem is addressed (and solved) by our prototype peer-to-peer network. The second problem is theoretically challenging and open ended in that we expect new search methods will constantly be developed. Below we describe a framework for deductive search within which we believe new methods can be couched.

Formalized mathematical proofs and theories are fragile objects[4] and although the semantic import of a theorem may well match or subsume a lemma being searched for, the shape of the theorem may not trivially match the search pattern. Trivial syntactic differences in theorems having little or no semantic content (*e.g.* $\forall x.\phi \wedge \psi$ vs. $\forall y.\psi \wedge \phi$) can make naively implemented search fail, users would be disappointed with such failures. Also, the equivalence of formulas in different logics differ, *e.g.* classically, $\phi \Rightarrow \psi$ and $\neg\phi \vee \psi$ are equivalent while they are not equivalent in the constructive setting; this must be taken into account in a heterogeneous setting by specifying the logic to use for deduction in search.

Methods for searching formal content might be based on unification[5] [7], but other strategies are possible as well. Of course, the problem of determining if a previously verified lemma (or collection of lemmas) subsumes a query target is undecidable in general.

2 A Peer-to-peer framework

The second author has built a prototype peer-to-peer network for sharing information between FDL's. The details of the architecture and of the implementation are described in [17]. Figure 1 gives an overview of the architecture. Peer-to-peer applications are becoming ubiquitous; they are used to share files, CPU cycles, and other resources. A principal advantage of the technology is its inherent fault-tolerance, there is no centralized component to fail. Peer-to-peer networks also

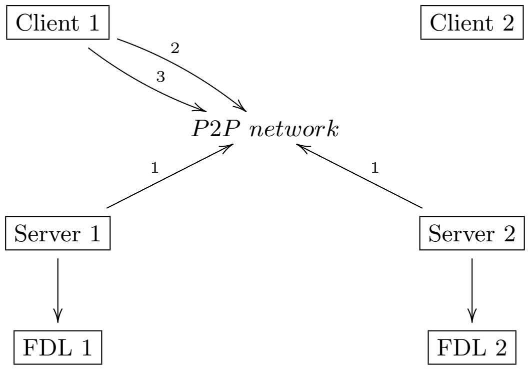
[4] Even tyros have first hand experience of this fragility. Small changes, *e.g.* adding an antecedent to the statement of a putative theorem, will often break a partial derivation that may have already been constructed. In fact, the most experienced users of such tools distinguish themselves from novices in that they build proofs in such a way as to avoid failure under minor perturbations to the statement being verified.

[5] Higher-order unification is undecidable but unification based methods can still be used since a user only needs a non-empty approximation to the complete search to satisfy a query.

support distributed discovery mechanisms. Sun has developed an open source peer-to-peer framework called JXTA [18] that is platform and programming language independent. Our system is built on JXTA.

2.1 A Prototype Implementation

The prototype is implemented in Java. It consists of about 6000 lines of code and includes a name and content based search engine for FDLs. The JXTA framework is used to provide the peer-to-peer network functionality.



Server 1 and Server 2 advertise the existence of the libraries FDL 1 and FDL 2 to the P2P network (step1). An interested client discovers these libraries (step 2). The client then queries the servers over the P2P network (step 3).

Fig. 1. P2P architecture

The FDL provides a TCP/IP based mechanism that allows clients to connect to the library and to issue simple search requests. The current interface to the library is limited to a search by name request and a search by content request. The search by name request returns a set of all theorems that contain a given string as a substring of their names. The search by content request returns the set of all theorems whose statements contain operators specified in the search. We developed a Boolean query language using the logical operations 'and', 'or', and 'not' to create more powerful expressions. While the query language is very simple it is surprisingly useful and serves to prove the mechanisms for searching remote libraries work. Indeed we found many new theorems in the Cornell libraries while testing our tool. Within the prototype, the search engine is implemented modularly and can easily be replaced if extensions are required.

To share theorems between groups we not only need to be able to search libraries but we also we need to discover the libraries themselves. In peer-to-peer networks, servers and clients have equal rights and responsibilities and are connected in a mesh topology. Peer-to-peer networks support mechanisms for discovering other peers and exchanging information between them. The JXTA framework provides a high level abstraction of these mechanisms. In our prototype, the libraries offering search services advertise it in the network. Clients can use those advertisements to invoke the services. All communication between peers is done within the peer-to-peer network having the advantage that problems with firewalls can be avoided (see [17] for details.) Since databases are not designed to participate directly in the peer-to-peer network a small server application was developed which is deployed in front of each library. Not only does this provide the interface for application libraries to join the peer-to-peer network, the server could also be used to provide additional functionality like authentication, authorization, and accounting.

3 A Framework for Deductive Search

In this section we describe the framework within which we address the problem of searching in distributed heterogeneous databases of formalized mathematics. The proposed framework is intended to be independent of the underlying individual databases; although we have in mind the FDL. The proposed framework does not make assumptions about the underlying databases but assumes that they provide a uniform interface; we (partially) specify that interface here.

The framework consists of the following components.

i.) A term structure used to communicate information across the network, the class of terms is denoted $Term_I$.[6]

ii.) An application programmers interface (API) supported by databases in the network.

iii.) A peer-to-peer architecture for the interconnection of the databases supporting functions for dynamically integrating new databases into the network and the protocol for communication between them.

iv.) A logical framework imposed on terms for describing the methods of deductive search. This is based on a concept of formal languages as decidable subclasses of terms in $Term_I$. These languages include the languages of the various logics together with the other extra-logical languages; *e.g.* representations of executable code (*e.g.* tactics, translations and others) together with all the other components necessary for the representation and manipulation of formalized mathematics. We also intend that informal content will be representable in the database as well.

The communication between systems is facilitated by a uniform and extensible term structure. This is the same term structure used internally by the FDL to represent formalized mathematics though we do not assume it is the internal representation used by all databases connected in the network; simply that they can translated their internal representations into the specified form. We also use the term structure in the description of the framework for deductive searching.

[6] The term structure described here is based on Nuprl's term structure [1] and is the one used internally within the FDL; we use it as an interface language.

3.1 Terms and Languages

The issues related to the representation of formalized mathematics are extraordinarily complex, especially as related to binding structures[7]. In this section we present the term structure used in the FDL which offers some generality in binding.

$Term_I$ is the class of recursive tree structures of the form

$$opid\{parameters\}(bterms)$$

where *opid* is the operator name, *parameters* is a list of value-type pairs and *bterms* is a list pairs consisting of a list of variables and a term. The parameter I is the class of abstract atomic identifiers allowing terms to refer to other terms in $Term_I$.

$$\begin{array}{ll}
\langle Term_I \rangle & ::= D \mid \langle opid \rangle \{ \langle parameter \rangle^* \} (\langle bterm_I \rangle^*) \\
\langle opid \rangle & ::= \langle C \rangle \langle C \rangle^* \\
\langle bterm_I \rangle & ::= \langle vars \rangle^* . \langle term_I \rangle \\
\langle parameter \rangle & ::= \langle value \rangle , \langle type \rangle \\
\langle C \rangle & ::= \textit{any character}
\end{array}$$

Parameters are constants or other arguments not constituent in the subterms *e.g.* within the FDL representation of Nuprl and PVS terms, the number 1 is a constant term of the form `natural{1:num}()`, the string "`xyzzy`" is represented by the term `string{"xyzzy":string}()`. The class of parameters is not fixed and can be extended to accommodate new languages and logics.

The *bterms* are the subterms of a term, possibly containing bound variables. A bterm consists of a list of variables (the bound variables) and a term (the body). Occurrences of variables included in the list of bound variables are bound in the body of the bterm. The use of bterms to encode binding operators can be seen by considering the encoding of the lambda abstraction in this structure. The term $\lambda x.M$ is encoded as `lambda{}(x.`M`)`. The opid of this term is `lambda`, it has no parameters, and it has one subterm M in which occurrences of the variable `x` are bound. The universal quantifier $\forall x{:}T.P$ is encoded as `all{}(`T`;x.`P`)`. The operator id is `all`, there are no parameters, and the operator has two subterms, T (the domain from which the bound variable `x` is chosen, and the bound term `x.`P where P is a term possibly containing free occurrences of the variable `x`. The fact that there may be multiple variables bound simultaneously in a subterm allows for the specification of an operator like Nuprl's *spread* operator, a generalized destructor for pairs; it is defined as `spread{}(`p`;`x`,`y`.`t`)`. The computation rule for spread makes clear how the simultaneous binding is used when the subterm p is a pair.

$$\texttt{spread(<}a\texttt{,}b\texttt{>;}\ x\texttt{,}y\texttt{.}t\texttt{)} \rightarrow t\texttt{[}a,b/x,y\texttt{]}$$

[7] For an interesting discussion of alternative binding structures see [13] and references therein.

i.e. if the first argument to `spread` is a pair of the form $\langle a, b\rangle$, spread simultaneously substitutes the first element of the pair for x in t and the the second element of the pair for y in t.

The index set I is not necessary for representing individual terms of a logic. By providing a means for terms to refer to other terms, the identifiers in the set I allow arbitrarily complex structures to be embedded within a collection of terms. Formal libraries are such structured collections. Allen has argued in detail elsewhere [2] that the references between terms should be abstract and atomic, thus the identifiers in I have no discernible structure and simply serve to refer. Indeed, within the conception of the FDL, the only significant property of the indices in a structured collection of terms is the topology of the constituent components induced by the references between the terms. More precisely, if I is the set of abstract identifiers used in $Term_I$ and I' is a set of abstract identifiers of equal or greater cardinality, then the process of uniformly replacing the abstract identifiers in a database of terms in $Term_I$ (under any injective map from I to I') results in a database of terms in $Term_{I'}$ which carries the same semantic import as the original.

This term structure has been used to represent both Nuprl, HOL and PVS terms in the FDL [8, 3].

A *language* is a decidable subset of terms *i.e.* $\mathcal{L}$ is a language if $\mathcal{L} \subseteq Term_I$ and for every $t \in Term_I$, we can decide if $t \in \mathcal{L}$. We assume interesting languages have names and abuse our own notion by identifying $\mathcal{L}$ both with the set of terms in the language and as a name of the set of terms. If $\mathcal{L}$ is a language we also use the name $\mathcal{L}$ to denote the property of membership in $\mathcal{L}$, thus if $\mathcal{L}$ occurs as a property it denotes the property $(\lambda t.\, t \in \mathcal{L})$. We note here that many of the languages we are interested in will be the terms of some logic, though not all interesting languages are logical.

3.2 Databases and Filters

In our model, libraries are collections of terms that refer to one another via the abstract atomic identifiers together with collections of certificates making claims about the stored terms.

Every individually stored term has an index $i \in I$ and terms may contain indexes to other terms. There is no requirement that every subterm of a term be indexed, though it is possible to build terms by storing subterms individually and referring to them by their abstract identifiers.

The evidence associated with a term is carried in the certificates for the term. We use the Greek 'ϵ', possibly decorated, as meta-variables denoting evidence. Terms retrieved from databases are packaged with the evidence associated with them and we call such packages *eterms*. We denote the type of evidentiary terms $Term_{I\mathcal{E}}$ and write t_ϵ to denote elements of this type. Evidence can be erased from an evidentiary term, $\lceil t_\epsilon \rceil = t$, *i.e.* $\lceil\cdot\rceil : Term_{I\mathcal{E}} \to Term_I$ and similarly, evidence can be garnered from an eterm $\lfloor t_\epsilon \rfloor = \epsilon$. No mechanism is provided for evidentiary terms to be composed (except by the database itself) and we expect to apply encryption mechanisms to enforce the constraint that only the database can deliver an eterm.

Once exported from a database, every term has at least one piece of evidence associated with it which an identifier of the database it originated in. Of particular practical interest and current research is the problem of how evidence in the form of certificates can be transferred from one database to another without forcing the re-verification of the certificates. We expect that methods based on digital signatures, like that described in [12], can be applied to this problem.

Within a database, term indexes (either stored as data or computed as needed) are used to select objects satisfying some property *e.g.* the terms of the PVS logic, or the Nuprl tactics. Term indexing is a tool to pair down the search space before the computationally expensive part of the search is performed by filtering objects that are unlikely to match. See [24] for efficient data-structures and algorithms for term indexing of first-order terms. We imagine many such indexing operations will be defined and provide the framework for specifying them here.

Given a database $\mathcal{D}$ of terms and a property $(\varphi : Term_{I\mathcal{E}} \rightarrow \mathbb{B})$ of terms, $\mathcal{D}\!\downarrow\!\varphi$ is the set of eterms in $\mathcal{D}$ satisfying φ:

$$\mathcal{D} \downarrow \varphi \overset{def}{=} \{t_\epsilon \in \mathcal{D} \mid \varphi[t_\epsilon]\,\}$$

If $P = \{\varphi_1, \cdots, \varphi_n\}$ is a set of properties of eterms, we write $\mathcal{D} \downarrow P$ to denote the set of terms satisfying at least one of the properties in P *i.e.* $\{\varphi_1, \cdots, \varphi_n\}$ is a notation for the property $(\lambda t_\epsilon.\, \varphi_1[t_\epsilon] \vee \cdots \vee \varphi_n[t_\epsilon])$. Note that the fact that properties are defined on eterms means we can filter databases by syntactic properties of the terms and/or by the evidence the terms carry.

If $\mathbf{D}$ is any set of databases and if φ is any property of terms, then:

$$\mathbf{D} \downarrow \varphi \overset{def}{=} \bigcup_{\mathcal{D} \in \mathbf{D}} \mathcal{D} \downarrow \varphi$$

Individual databases may vary in their underlying implementations though they must all support translations from their internal representations into the term representation that serves as the medium of communication between systems. A framework like the one proposed here, characterized by operations on terms, allows for specification of search methods in terms of the interface language.

3.3 Translations

Our methodology for sharing results rests on the idea that there may be effective translations between logics. In [26], an application similar to the one here is given which accounts for the use of multiple logics within a single specification.

If there is a partial function f mapping terms to terms $(f : Term_I \rightarrow Term_I)$ such that the domain of f is $\mathcal{L}'$ and the codomain of f is $\mathcal{L}$ we call $\langle f, \mathcal{L}', \mathcal{L}\rangle$ a *translation.* Note that since the domains of translations may intersect, we explicitly carry the domain and codomain with the translation[8]. If f is a function from $\mathcal{L}'$ to $\mathcal{L}$ and $t \in \mathcal{L}'$ then $f(t)$ evidently denotes the translation of $t \in \mathcal{L}'$ into a term in the language $\mathcal{L}$.

We are typically interested in translations that make some kind of guarantee, *e.g.* that some property is preserved by the translation. The evidence for guarantees are represented in certificates[9] and so, a translation which generates evidence for its own correctness must generate certificates. Only the database can generate certificates and so evidentiary translations must carry references to certificate kinds (a certificate generator) and make requests to the database to execute them. A translation certificate kind (of type $\mathcal{CK}$) takes an eterm t_ϵ whose term part is of type $\mathcal{L}'$ and returns a new eterm $t'_{\epsilon'}$ where ϵ' is the new evidence for the translated term t'. As a side effect it adds the new term ($t' = f(\ulcorner t_\epsilon \urcorner)$) to the database and creates new certificates for t' both preserving the old certificates $\llcorner t_\epsilon \lrcorner$ (noting that they belonged to the untranslated term t_ϵ) and generating a new certificate certifying that t' was indeed generated by the translation of t_ϵ. If $\mathcal{C}$ is a reference to a certificate kind we write $\mathcal{C}^*(t_\epsilon)$ to denote the result of a request for the database to apply certificate kind $\mathcal{C}$ to t_ϵ.

Thus,the type of evidentiary translations is defined to be the four-tuple:

$$\mathbf{Tr} \overset{def}{=} (Term_I \to Term_I) \times (Term_I\,\mathbf{Set}) \times (Term_I\,\mathbf{Set}) \times \mathcal{CK}$$

If $\tau = \langle f, \mathcal{L}, \mathcal{L}', C\rangle$ is in $\mathbf{Tr}$ then:

$$\tau(t_\epsilon) \overset{def}{=} C^*(t_\epsilon) \qquad dom(\tau) \overset{def}{=} \mathcal{L} \qquad codom(\tau) \overset{def}{=} \mathcal{L}'$$

We define the composition of evidentiary translations $(\tau \circ \hat{\tau})$ as follows. If $\tau = \langle f, \mathcal{L}', \hat{\mathcal{L}}, \mathcal{C}\rangle$ and $\hat{\tau} = \langle g, \hat{\mathcal{L}}, \mathcal{L}, \hat{\mathcal{C}}\rangle$ are compatible translations (*i.e.* if $codom(\tau) = dom(\hat{\tau})$) then:

$$\tau \circ \hat{\tau} = \langle f \circ g,\, \mathcal{L}',\, \mathcal{L},\, \mathcal{C} \circ \hat{\mathcal{C}}\rangle$$

The notation $(f \circ g)$ denotes ordinary function composition defined as $(f \circ g)(x) = g(f(x))$.

The identity translation on a language $\mathcal{L}$ is defined as $Id_{\mathcal{L}} = \langle \lambda x.x, \mathcal{L}, \mathcal{L}, \mathcal{C}_{Id}\rangle$, where $\mathcal{C}_{Id}$ is the certificate kind that has no side effects and $\mathcal{C}^*_{Id}(t_\epsilon)$ simply returns the value t_ϵ.

In practice the syntactic translations between the formal languages may be straightforward, the hard part for nontrivial translations between logics is the justification that the translation preserves intended meanings. The justification that the intended meaning is preserved by a translation may be informal or

[8] This is consistent with formalizations of category theory [16, 6] in which each arrow has a *dom* and *codom* function and so arrows in non-trivial categories are triples. With this in mind, we see that languages are the objects of the category, translations are the arrows and composition is defined as below.

[9] Certificates justifying a translation may refer to an informal argument (a paper)or they may refer to other formal content.

formal. To the extent that a user believes the justification for a translation, he will include it (or not) in the set of translations he wants considered when calculating a set of candidates for a search.

3.4 Stratification of Languages by Translations

To consider the relationships between objects in different languages in a heterogeneous database, we stratify terms relative to a fixed language $\mathcal{L}$ by their distance from that language via some sequence of translations in a specified[10] set $\mathbf{T}$. For the purposes of search, we are ultimately interested terms that can be effectively translated from one language (logic) to another. Based on this idea, we provide the following definition of the n-closure of a translation set $\mathbf{T}$ relative to a language $\mathcal{L}$.

$$\begin{aligned} \mathbf{T}^0_{\mathcal{L}} &\overset{def}{=} \{\tau : \mathbf{Tr} \mid \tau = Id_{\mathcal{L}}\} \\ \mathbf{T}^{n+1}_{\mathcal{L}} &\overset{def}{=} \{\tau : \mathbf{Tr} \mid \exists \tau' \in \mathbf{T}.\, \exists \hat{\tau} \in \mathbf{T}^n_{\mathcal{L}}.\, codom(\tau') = dom(\hat{\tau}) \wedge \tau = (\tau' \; o \; \hat{\tau})\} \end{aligned}$$

The class $\mathbf{T}^n_{\mathcal{L}}$ consists of all translations mapping terms of languages $\mathcal{L}'$ to the language $\mathcal{L}$ by a sequence of n translations from the set $\mathbf{T}$.

We define the closure of the stratification to be the union of all the levels.

$$\mathbf{T}^*_{\mathcal{L}} \overset{def}{=} \bigcup_{i \in \mathbb{N}} \mathbf{T}^i_{\mathcal{L}}$$

This is the set of all terms interpretable as terms in $\mathcal{L}$ by some sequence of translations in $\mathbf{T}$.

The languages at level k in $\mathbf{T}^k_{\mathcal{L}}$ can be retrieved by projecting them from the translations in that level.

$$\|\mathbf{T}^n_{\mathcal{L}}\| \overset{def}{=} \pi_2(\mathbf{T}^n_{\mathcal{L}})$$

where the projection functions are lifted to sets of tuples point-wise in the natural way (*i.e.* if $S \subseteq S_1 \times \cdots \times S_n$ then $\pi_i(S) = \{x_i : S_i | \langle x_1, \cdots, x_i, \cdots, x_n\rangle \in S\}$ where $0 < i \leq n\}$).

The distance of a language $\mathcal{L}'$ from $\mathcal{L}$ under the translations set $\mathbf{T}$ is defined if and only if $\mathcal{L}' \in \|\mathbf{T}^k_{\mathcal{L}}\|$ for some k and is the minimum k such that $\mathcal{L}' \in \|\mathbf{T}^k_{\mathcal{L}}\|$.

The languages included in the closure $\mathbf{T}^*_{\mathcal{L}}$ determine the potential search space (and translations to use) to satisfy a query in the language $\mathcal{L}$.

The set of terms from a collection of databases $\mathbf{D}$ under translation set T at distance k from $\mathcal{L}$ is the set $\mathbf{D} \downarrow \|\mathbf{T}^k_{\mathcal{L}}\|$. We call this set the *k-step candidate terms.* The *full set of candidate terms* are the terms in $\mathbf{D} \downarrow \|\mathbf{T}^*_{\mathcal{L}}\|$. These sets are sets of terms in the languages $\mathcal{L}'$, that can be translated into terms in $\mathcal{L}$. We are of course not only interested in the sets of terms which *can* be translated into the language $\mathcal{L}$ but are interested in their translations. The *effective candidate*

[10] We specify the set of allowable translations $\mathbf{T}$ because it is a basic tenet of our approach that users must be able to account for the results they receive.

terms of $\mathcal{L}$ from $\mathbf{D}$ under $\mathbf{T}$ is the set of terms from the languages in $\mathbf{D} \downarrow \|\mathbf{T}^*_{\mathcal{L}}\|$ paired with their translations.

The following property states that if τ is in set of translations in $\mathbf{T}^*_{\mathcal{L}}$, then every term t in $dom(\tau)$ actually is mapped to a term in $\mathcal{L}$ by τ.

$$\forall \mathbf{T} : \mathbf{Tr}\, Set.\ \forall \mathcal{L} \subseteq Term_I.\ \forall \tau \in \mathbf{T}^*_{\mathcal{L}}.\ \forall t \in dom(\tau).\ \tau(t) \in \mathcal{L}$$

The proof of this property is by induction on the level k at which τ occurs in $\mathbf{T}^*_{\mathcal{L}}$ and then follows directly from the definition and the properties of composition.

The fact that translations are not necessarily invertible determines how search is done in the languages that are one or more translation steps from $\mathcal{L}$; we apply the search methods implemented for $\mathcal{L}$ to terms in $\langle t, \tau \rangle \in \mathbf{T}^*_{\mathcal{L}}$ by searching against the translated term $\tau(t)$.

As an example of these definitions, consider the following. There are extant translations of HOL terms to classical Nuprl terms (τ_1), a translation of Isabelle terms to classical Nuprl terms (τ_2) and a translation of ACL2 terms into HOL terms (τ_3).

$$\begin{aligned}
&\|\{\}^0_{Nuprl})\| = \{Nuprl\}\\
&\|\{\tau_1, \tau_2, \tau_3\}^1_{Nuprl}\| = \{HOL, Isabelle\}\\
&\|\{\tau_1, \tau_3\}^1_{Nuprl}\| = \{HOL\}\\
&\|\{\tau_1, \tau_2, \tau_3\}^2_{Nuprl}\| = \{ACL2\}\\
&\|\{\tau_1, \tau_2\}^2_{Nuprl}\| = \{\}
\end{aligned}$$

Note that the levels as specified here are not cumulative; *e.g.* $Nuprl \in \|\{\tau_1, \tau_3\}^0_{Nuprl}\|$ but $Nuprl \notin \|\{\tau_1, \tau_3\}^1_{Nuprl}\|$. Thus a user interested in searching HOL theorems but excluding theorems of Nuprl to satisfy a Nuprl proof can specify the domain of search as $\|\{\tau_1, \tau_3\}^1_{Nuprl}\|$.

Note that the translations between these different logical theories preserve validity of theorems but do not necessarily translate proofs. Translations are justified somehow, formally or informally. But such justifications may be based on semantic arguments and the translation of proofs is unknown.

3.5 Deductive Searching

Based on these ideas we propose the following general framework for search in heterogeneous databases of theorems from multiple logics. We cast our description in terms of sequents, though it should be obvious how to recast these ideas in non-sequent based logics.

We are interested in searching the library to complete a proof of some sequent of the form $\Gamma \vdash_{\mathcal{L}} \Delta$. Should some $\phi \in \Delta$ already be proved in $\mathcal{L}$ and stored in the library, then $\Gamma, \phi \vdash_{\mathcal{L}} \Delta$ can be trivially proved in $\mathcal{L}$ by cutting in ϕ and then invoking the axiom rule. Less directly, perhaps there is some translation mapping a theorem of some other logic into the term ϕ in the language of $\mathcal{L}$.

Search will be performed using procedures that are, in most cases, incomplete. Our framework assumes that a search procedure used to find results within the context of some logic $\mathcal{L}$ can construct a "proof" in $\mathcal{L}$ when a search is successful *e.g.* a search procedure to be used in the context of an HOL theorem will return both the lemmas found in the database *and* tactic to apply them in the context of sequent being searched for.

Let $\Gamma \vdash_{\mathcal{L}} \Delta$ be a sequent in the logical language $\mathcal{L}$, let $\mathbf{D}$ be a collection of databases $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_n\}$ and let $\mathcal{S}$ be a proof search procedure for $\mathcal{L}$. We define $[\![\Gamma \vdash_{\mathcal{L}} \Delta]\!]_{\mathbf{D},\mathcal{S}}$ as follows:

$$[\![\Gamma \vdash_{\mathcal{L}} \Delta]\!]_{\mathbf{D},\mathcal{S}} \overset{def}{=} \{\langle \Gamma', \rho\rangle | \Gamma' \subseteq \cup\mathbf{D} \wedge \rho \text{ proves } \Gamma, \Gamma' \vdash_{\mathcal{L}} \Delta\}$$

where $\cup\mathbf{D}$ is the set of all terms in the databases in the set $\mathbf{D}$. Thus $[\![\Gamma \vdash \Delta]\!]_{\mathbf{D},\mathcal{S}}$ is a set of pairs consisting of lists of theorems Γ' from the databases in $\mathbf{D}$, paired with a method of proof ρ which proves the sequent $\Gamma, \Gamma' \vdash_{\mathcal{L}} \Delta$. The proof ρ (together with the theorems in Γ') is the information needed by the prover for the logic $\mathcal{L}$ to complete the proof of the sequent $\Gamma \vdash_{\mathcal{L}} \Delta$. We write $[\![\Delta]\!]_{\mathbf{D},\mathcal{S}}$ for $[\![\vdash \Delta]\!]_{\mathbf{D},\mathcal{S}}$ and $[\![\phi]\!]_{\mathbf{D},\mathcal{S}}$ for $[\![\vdash \phi]\!]_{\mathbf{D},\mathcal{S}}$.

Now we discuss some consequences and applications of the definition.

Typically, the actual answer set $[\![\Gamma \vdash_{\mathcal{L}} \Delta]\!]_{\mathbf{D},\mathcal{S}}$ is infinite; to see this note that once some list of terms is enough to prove the desired result, any extension of that list will also do[11]. However, note that non-empty approximations to $[\![\Gamma \vdash_{\mathcal{L}} \Delta]\!]_{\mathbf{D},\mathcal{S}}$ are usually satisfactory answers to queries *i.e.* any answer provides a means to prove $\Gamma \vdash_{\mathcal{L}} \Delta$ from the contents of databases in $\mathbf{D}$. Indeed, although $[\![\Gamma \vdash_{\mathcal{L}} \Delta]\!]_{\mathbf{D},\mathcal{S}}$ is defined as a complete answer, only one answer is ever required to discharge the proof obligation. Multiple answers may provide the requester with options allowing them to make choices based on any number of criteria. We can imagine that one criteria might be to choose the answer that requires the minimum update to the local database. Others might be based on elegance.

To search a collection of databases $\mathbf{D}$ for an individual theorem ϕ, one searches for an approximation of $[\![\phi]\!]_{\mathbf{D},\mathcal{S}}$. Note that if any theorem ϕ of $\mathcal{L}$ is in the database, then $\langle\{\phi\}, Axiom\{\phi\}\rangle \in [\![\phi]\!]_{\mathbf{D},\mathcal{S}}$ where $Axiom\{\phi\}$ is the axiom rule for $\mathcal{L}$ *i.e.* the rule justifying sequents of the form

$$\Gamma_1, \phi, \Gamma_2 \vdash_{\mathcal{L}} \Delta_1, \phi, \Delta_2$$

Thus, we have defined a framework for deductive search in a way that users can both account for the results they receive and can apply the results in proofs.

Name and Content based Search in the Deductive Framework We note here that name and content based searches can be fit into the framework just described for deductive search. For name searches, we assume that there is a function *name* mapping library objects to user specified names (strings of characters) and returning the empty string if a name does not exist. We define a logic of names $\mathcal{L}N$, where the language of the logic of names is $Term_I$ (all

[11] Of course we are excluding various resource-bounded and substrutural logics from this consideration.

terms, including strings, are in the language of the logic of names). The logic $\mathcal{L}N$ has one proof rule.

$$\frac{}{t \vdash_{\mathcal{L}N} s}\text{Ax} \qquad \text{if } s \in \mathit{string} \wedge\ s \subseteq \mathit{name}(t)$$

Here, s is a string and $s \subseteq s'$ if and only if s is a substring of s'. Thus, a name search for all objects in some collection of databases $\mathbf{D}$ is computed as $[\![s]\!]_{\mathbf{D},\mathcal{L}N}$. To search the names of the terms of some language $\mathcal{L}$ for a particular string s can be specified as $[\![s]\!]_{D\downarrow\mathcal{L},\mathcal{L}N}$; *e.g.* to search Nuprl terms having the string "*list*" as a substring of their name is specified as $[\![\text{“}\mathit{list}\text{”}]\!]_{(\mathbf{D}\downarrow Nuprl),\mathcal{L}N}$. The result of the search would be a set of pairs $\{(t_1, Ax), \cdots, (t_n, Ax)\}$.

We can cast content-based search in the deductive framework by similarly defining a logic of content.

4 Apologia and Conclusion

In this paper we have described an implementation of a peer-to-peer framework for connecting databases of formalized mathematics [17] and the term structures used to communicating between them. We have proposed a framework for deductive searching in distributed collections of heterogeneous databases and have described how name and content based searchers can be cast into the deductive framework. We have emphasized that both *evidence* and *effective methods* of translation and proof should be included as part of the results of searches. There is obviously significant work that remains to be done, most features of the proposed framework have not been implemented. Work on representing evidence using Allen's certificate mechanism continues at Cornell and Wyoming. We have only implemented name and content based searching and intend to further explore more powerful deductive methods based on heuristic search.

In a number of ways this paper is unsatisfactory: some aspects of the proposed framework for sharing and searching have been elaborated in too much detail; while a number of aspects of the presentation are too vague. However, we believe that the proposed approach has several advantages. We do not propose to impose any particular logic or any absolute criteria for correctness on users. To us, any attempt to make such impositions will result in failure, perhaps not for technical reasons but for social ones[12]. Choice of logic and the criteria for correctness are matters for individual deliberation. Instead, we have proposed a framework within which mechanisms for translating between logics can be implemented and where mechanisms to account for results is embedded within the framework. The only imposition we reluctantly make is one of syntax, of term structure. And although we can imagine that XML or some other structured notation would work as well as the one presented here, we can not imagine how to avoid such an imposition. In any case, matters of syntax require far less commitment than

[12] One might reasonably claim that the QED project [4] ended prematurely down for precisely this reason.

matters of semantics. We believe that something very much like the system proposed here, if not this one, will eventually provide a practical means for seamless sharing formal mathematics.

Acknowledgments We would like to thank Stuart Allen and Constable at Cornell for the intellectual enjoyment gained from the time spent discussing FDL related matters. We thank Rich Eaton also at Cornell for cheerful responses to our unreasonable requests for his time and for his programming support. The first author also thanks John Paul at the University of Wyoming for acting as a sounding board for many of the ideas presented here.

References

1. Stuart Allen. *Nuprl Basics.* Cornell University, 2001. `http://www.cs.cornell.edu/Info/People/sfa/Nuprl/NuprlPrimitives/`.
2. Stuart Allen. Abstract identifiers, intertextual reference and a computational basis for recordkeeping. *First Monday*, 9(2), February 2004. `http://firstmonday.org/issues/issue9_2/allen/`.
3. Stuart F. Allen, Mark Bickford, Robert Constable, Richard Eaton, and Christoph. Kreitz. A Nuprl-PVS connection: Integrating libraries of formal mathematics. Technical Report TR2003-1889, Cornell University, 2003. `http://techreports.library.cornell.edu:8081/Dienst/UI/1.0/Display/cul.cis/TR2003-1889`.
4. Anonymous. QED Manifesto. `http://www-unix.mcs.anl.gov/qed/`.
5. James Caldwell and Judith Underwood. Classical tools for constructive proof search. In Didier Galmiche, editor, *Proceedings of the CADE-13 Workshop on Proof search in Type-theoretic languages.*, Rutgers N.J., July 1996.
6. James Caldwell and Tjark Weber. A formal framework for constructive category theory. `http://www.cs.uwyo.edu/~jlc/papers`, July 2003.
7. Gilles Dowek. *Higher-Order Unification and Matching*, chapter 16, pages 1009–1065. In Robinson and Voronov [23], 2001.
8. The Formal Digital Libraries Project (Homepage). `http://www.nuprl.org/html/Digital_Libraries.html`.
9. D. M. Gabbay and M. de Rijke, editors. *Frontiers of Combining Systems 2 (Proceedings of the Second International Workshop, FroCoS'98, Amsterdam, The Netherlands, October 1998)*, volume 7 of *Studies in Logic and Computation.* Research Studies Press Ltd., 2000.
10. Joseph Goguen and Rod Burstall. Introducing institutions. In Edward Clarke and Dexter Kozen, editors, *Proceedings, Logics of Programming Workshop*, volume 164 of *LNCS*, pages 221–256. Springer, 1984.
11. Joseph A. Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.
12. Jim Grundy. Trustworthy storage and exchange of theorems. Technical Report TUCS-TR-1, Turku, Finland, April 1996.
13. Jaakko Hintikka. *The Principles of Mathematics Revisited.* Cambridge University Press, 1996.

14. Douglas Howe. Toward sharing libraries of mathematics between theorem provers. In Gabbay and de Rijke [9], pages 161–176.
15. Douglas J. Howe. Importing mathematics from HOL into Nuprl. In J. von Wright, J. Grundy, and J. Harrison, editors, *Proceedings of the* 11th *International Conference on Theorem Proving in Higher Order Logics*, volume 1125, of *LNCS*, pages 267–282. Springer-Verlag, 1996.
16. G. Huet and A. Saibi. Constructive category theory. In Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner.* MIT, 1998.
17. Christoph Jechlitschek. *Distributed Sharing of Formalized Mathematics: a P2P approach.* Master's thesis, University of Wyoming, Laramie, WY, May 2004.
18. The JXTA project homepage. `http://www.jxta.org`.
19. Evan Moran. Forthcoming Cornell Ph.D. Thesis, Dept. of Computer Science.
20. Pavel Naumov. Importing Isabelle formal mathematics into Nuprl. *The 12th International Conference on Theorem Proving in Higher Order Logics, supplemental proceedings*, 1999. `http://www-sop.inria.fr/croap/TPHOLs99/ps/paper4.ps`.
21. Pavel Naumov, Mark O. Stehr, and Jose Meseguer. The HOL/NuPRL proof translator: A practical approach to interoperability. In *Proceedings of the* 14th *International Conference on Theorem Proving in Higher Order Logics*, volume 2152 of *LNCS*, pages 329 – 345. Springer, 2001.
22. National Center for Biotechnology Information (Homepage). `http://www.ncbi.nlm.nih.gov/`.
23. Alan Robinson and Andrei Voronov, editors. *Handbook of Automated Reasoning: Volume II.* MIT, North Holland, 2001.
24. R. Sekar, I. V. Ramakishnan, and Ardrei Voronkov. *Term Indexing*, chapter 26, pages 1855–1964. In Robinson and Voronov [23], 2001.
25. Mark Staples. Linking ACL2 and HOL. Technical Report 476, Cambridge University, Computer Laboratory, 1999. `http://citeseer.ist.psu.edu/staples99linking.html`.
26. Andrzej Tarlecki. Towards heterogeneous specifications. In Gabbay and de Rijke [9], pages 337–360.