Conflict Detection for Integration of Taxonomic Data Sources

Suzanne M. Embury, Andrew C. Jones,
Iain Sutherland and W. Alex Gray
Department of Computer Science,
Cardiff University,
Cardiff, CF2 3XF, U.K.
{S.M.Embury|A.C.Jones}@cs.cf.ac.uk
{I.Sutherland|W.A.Gray}@cs.cf.ac.uk

Richard J. White and John S. Robinson
Biodiversity & Ecology Research Division,
School of Biological Sciences,
University of Southampton,
Southampton, SO16 7PX
UK
{R.J.White|J.S.Robinson}@soton.ac.uk

Frank A. Bisby and Sue M. Brandt
Biodiversity Informatics Laboratory,
Centre for Plant Diversity & Systematics,
The University of Reading,
Reading RG6 6AS, U.K.
{F.A.Bisby|S.M.Brandt}@reading.ac.uk

Abstract

Over recent years, international initiatives such as the 1993 U.N. Convention on Biological Diversity have highlighted the need for information about species diversity on a global scale. However, attempts to build global information systems by integrating smaller, independently created biodiversity databases have been hampered by differences in the sets of species names used. Some databases use different names to refer to the same species, while in other cases the same name can be applied to differing definitions of a species, or even entirely different species.

The LITCHI project aims to assist biologists in the integration of databases by searching for conflicts within taxonomic checklists (i.e. lists of the species names used in a database and the relationships between them). In order to detect such conflicts, we have created a formal model of taxonomic practice, which describes (amongst other things) what it means for a checklist to be consistent and well-specified. This model has been used as the basis for a prototype tool that uses Prolog to search for naming conflicts within a relational database of checklists. In this paper, we describe the background to our formal model and show how it has been used to implement the LITCHI system. Our prototype tool is already proving its worth by detecting conflicts and errors within real taxonomic checklists.

1. Introduction

Over recent years, international initiatives such as the 1993 U.N. Convention on Biological Diversity have highlighted the need for information about species diversity on a global scale. However, biologists have tended to concentrate their efforts on creating smaller, more focussed databases, perhaps containing information on the species found in a particular geographical area[1] or on a group of related species[2]. Despite the success of these databases, the increasing demands for biodiversity data covering a wider international scope is creating a corresponding pressure for these databases to be integrated to form larger information sources, and to allow a single query to be evaluated across many databases at once.

The problems inherent in the integration of independently developed databases have been the subject of much study within the database research community (e.g. [1, 9, 5]). However, the majority of this work has concentrated on the problems of masking syntactic differences between query languages, and structural and semantic differences between database schemas. In fact, a third problem must also be considered — that of dealing with the semantic differences between the *data*

[1] For example, the Legumes of Northern Eurasia database [12] or the Legumes of West Asia collection [6].

[2] Examples include the ILDIS World Database of Legumes [13] and the FishBase global information system on fishes [2].

values used by the individual databases. For example, once we have determined that attribute `weekDay` in one database corresponds to attribute `dayOfWeek` in another, we must also ensure that the data values stored within these attributes correspond to the same real world entities. We must know that the data value `Monday` in one database refers to the same day of the week as the data value `mon` in the other. This third problem has so far received much less attention than the other aspects of database integration.

Unfortunately, this is a real issue for the creators of integrated databases. Some differences in value semantics can be dealt with by keeping track of the units used by attributes, and by providing mapping functions to convert between different units. Data types such as temperatures, time values and currencies are all amenable to this kind of handling [8]. But for many data types, particularly those represented as string values, there is no simple mapping function that can be used to convert between different semantic viewpoints. It is possible, of course, to create a table of mappings between equivalent values (Singh, for example, suggests the use of rules to model such a table [10]) but the sheer volume of possible data values makes this approach impractical in real applications.

Species names in biological databases are an example of this kind of troublesome data type. The association of a name with a particular species, or even the decision as to which group of organisms actually comprises a single species, involves an element of judgement and subjectivity, and can change over time and between scientific communities. This can mean that the species referred to as "*Astragalus aboriginum* Sprengel" in one database is known as "*Astragalus forwoodii* S. Watson" in another. If a query involving the first name is to be evaluated over both these databases, then it must be reformulated to use the second name when executed against the latter database if a complete set of results is to be returned.

This kind of problem occurs all too frequently, because of the large number of names that are in circulation relative to the number of known species. For example, the ILDIS World Database of Legumes contains information on 19,043 taxa[3] but recognises a total of 37,394 names which have been applied to those taxa. If ILDIS is representative of taxonomy as a whole, we can expect to encounter an average of 2 names for each taxon in a biological database. It is therefore reasonable to suppose that semantic value clashes will occur frequently when evaluating queries over integrated biological information sources, and that they merit serious attention.

To add to this problem, species names are of particular importance in the integration of biological databases since they often act as the *join attribute* in global queries which indicates when data in one database refers to the same species as data stored in another database. Before a collection of biological databases can be integrated, therefore, it is necessary to identify and understand any conflicts in the way species names are used by the component databases, so that the appropriate query conversions can be carried out.

The aim of the LITCHI project[4] is to provide a tool for use by skilled taxonomic editors, which will assist in the process of identifying potential conflicts within *taxonomic checklists*. A checklist is a representation of the names of biological taxa that are used within a particular biological database, and the relationships between those names. The approach we have taken is to construct a formal model (in first order logic) of the way scientific names are used to denote particular species in common taxonomic practice. In order to keep the complexity of this task within manageable bounds, we have concentrated on formalising current practice within botanical nomenclature, rather than trying to combine these rules with the subtly different conventions of zoological and bacterial nomenclature. We have then used this model to derive queries which will search for potential naming conflicts within or between checklists of botanical names.

In this paper, we describe how we are making use of this formal model of botanical nomenclature to detect data-level semantic conflicts prior to the integration of taxonomic databases. In Section 2 we outline the structure of taxonomic checklists, while in Section 3 we describe the internal structure of scientific names, and how this information is modelled for use within the LITCHI project. In Section 4 we present some examples of the rules we have developed to model observed taxonomic practice, after which, in Section 5, we discuss our approach to the representation of conflicts, once they have been detected. Section 6 describes the implementation of the LITCHI tool. Finally, we present some of the results that have been obtained by applying our software to data extracted from real checklists, and give some directions for future work.

[3] A taxon is a group (such as a species) that arises out of the process of establishing and defining systematic groups of organisms. The figure given here for taxa in ILDIS consists principally of species, but also includes some subspecies and varieties. For conciseness, we will use the words "species" and "taxon" interchangeably throughout this paper.

[4] LITCHI (Logic-based Integration of Taxonomic Conflicts in Heterogeneous Information systems) is funded by a grant from the BBSRC/EPSRC Bioinformatics Initiative.

```
Astragalus abnormalis Rech. f.  (accepted name)

Astragalus accumbens Sheldon  (accepted name)
  Astragalus procumbens S. Watson  (synonym)
  Batidophaca accumbens (Sheldon) Rydb.  (synonym)

Astragalus adpressipilosus Gontsch.  (accepted name)
  Astragalus adpresse-pilosus Gontsch.  (orthographic variant)

Astragalus bahrakianus Grey-Wilson  (accepted name)
  Astragalus dictamnoides sensu Podl.  (misapplied name)

Astragalus aboriginum Sprengel  (accepted name)
  Astragalus aboriginum Sprengel var. fastigiorum M.E. Jones  (synonym)
  Astragalus aboriginum Sprengel var. glabriuscula (Hook.) Kuntze  (synonym)
  Astragalus aboriginum Sprengel var. muriei Hulten  (synonym)
  Astragalus forwoodii S. Watson  (synonym)
  Astragalus forwoodii S. Watson var. wallowensis (Rydb.) M. Peck  (synonym)
  Astragalus glabriusculus (Hook.) A. Gray  (synonym)
  Astragalus glabriusculus (Hook.) A. Gray var. major A. Gray  (synonym)
  Astragalus glabriusculus (Hook.) A. Gray var. patiosus Sheldon  (synonym)
  Atelophragma aboriginorum (Richardson) Rydb.  (synonym)
  Atelophragma glabriuscula (Hook.) Rydb.  (synonym)
  Atelophragma herriotii Rydb.  (synonym)
  ...
```

Figure 1. A Fragment of the ILDIS Checklist

2. Taxonomic Checklists

The basic data sets on which the LITCHI tool must operate are taxonomic checklists — lists which contain details of a (hopefully consistent) set of scientific names and the relationships between them. Figure 1 illustrates this and shows a fragment of the checklist retrieved from the ILDIS database. Each group of names refers to a single taxon, and in each case the first of the names is labelled as the *accepted name*. This is the name that has been chosen by the group of taxonomists who contributed to the checklist or database as the most appropriate one for that taxon.

It is common, however, for several other names to have been used to refer to a taxon — perhaps because a species was discovered and named by two biologists independently, or because it was once thought to be part of another taxon. It is impractical to remove all references to incorrectly or inappropriately used names[5]. Moreover, genuine differences of opinion can exist between taxonomic communities; differences which must be accommodated, not eradicated. Because of this, such names are listed within the checklist as recognised synonyms for the chosen accepted name. In Figure 1, synonyms are indented and listed beneath their respective accepted name.

The third and fourth entries in the checklist fragment illustrate two different kinds of synonym. The synonym given in the third entry (*Astragalus adpresse-pilosus* Gontsch.) is labelled as an "orthographic variant" of its accepted name. This indicates that it is a common alternative or mistaken spelling of some other taxon name. In the fourth entry, the synonym (*Astragalus dictamnoides* sensu Podl.) has been given a "misapplied name" tag. This indicates that the name of another taxon has been mistakenly used by some author (in this case "Podl.") to refer to the current taxon (in this case, *Astragalus bahrakianus* Grey-Wilson). There are a number of other kinds of synonym, and the interested reader is referred to standard texts on biological nomenclature (e.g. Jeffrey [4]) for further details.

The final set of names in Figure 1 shows just some of the synonyms of *Astragalus aboriginum* Sprengel, and illustrates the complexity of the set of names for some taxa. As we shall see in Section 4, we can make use

[5]This could require many thousands of books to be burned and many hundreds of thousands of labels on specimens to be altered.

of the synonyms attached to a pair of accepted names from different checklists to detect potential taxonomic conflicts. Alternatively, agreement between the synonyms for two different taxon entries can suggest that they in fact refer to the same taxon. We can also make use of formal descriptions of the situations in which certain synonym labels should be present to determine whether a given checklist has been correctly annotated.

3. The Structure of Scientific Names

In addition to exploiting the relationships between accepted names and synonyms to detect conflicts, we can also make use of the internal components of scientific names of plants. To a large extent, the structure of scientific names is laid down by the internationally agreed Code of Botanical Nomenclature [3]. This code describes a series of rules, supplemented in some cases by additional recommendations, which specify the conventions to be followed in the creation of new botanical names. The taxa into which organisms can be classified are arranged in a hierarchical system of taxonomic categories. These include: *Familia* (family), *Genus*, *Species*, *Subspecies* and *Varietas* (variety). The most fundamental of these for our purposes is the species category. The name of a species consists of a two part Latin name (a *binomial*) and a description of the publishing authority. For example, consider the name:

Astragalus alpinus L.

Here, *Astragalus* is the genus to which the taxon belongs. The second part of the name (*alpinus*) identifies the species within that genus and is known as the *specific epithet*. The final part of the name, "L.", indicates that Linnaeus was the first person to validly publish the name *Astragalus alpinus*.

Taxa occurring at lower levels in the taxonomic hierarchy, such as subspecies and varieties, are named by adding additional components to the higher level name. For example, Hultén has published the following subspecies of *Astragalus alpinus* L.

Astragalus alpinus L. subsp. *alaskanus* Hultén

Thus, subspecies names contain an additional component (the *subspecific epithet*) which identifies the subspecies within the given species. Varieties are named in a similar way. E.g.

Astragalus alpinus L. var. *brunetianus* Fern.

The structure of the authority string within a taxonomic name can also provide useful information when checking for conflicts. Sometimes, as a result of changing taxonomic opinion, a taxon may be moved to another position, or even another rank, in the taxonomic hierarchy. This results in a new name for the taxon and a more complicated authority string. For example, when Popov transferred the taxon *Caragana acaulis* Baker to the genus *Chesneya*, its name became:

Chesneya acaulis (Baker) Popov

The new species name will typically retain the specific epithet from the original name (which is called the *basionym* or *base name*). The authority of the new name is given by prefixing the names of the revising authors with the names of the original authors (the *basionym authority*). Thus, something of the history of the name is preserved in its revised form. This information can help us to match species names correctly when a database using this new name is integrated with a database containing the original name.

3.1. Modelling the Structure of Scientific Names

In order to allow us to query the internal structure of botanical names when searching for conflicts, we store the checklists which are to be integrated within a DBMS. The schema of this database (shown as an ER-model in Figure 2) must therefore model the structure of both checklists and scientific names.

The heart of this model is the inheritance hierarchy rooted at the `Name` class. This hierarchy represents the fact that there may be a number of different kinds of name present in a checklist. However, the model also shows that the different kinds of name also exist within a part-component hierarchy, represented by the `HasGenus`, `HasSpecies`, `HasSubspecies` and `HasAuthority` relationships. For example, the subspecies name *Astragalus alpinus* L. subsp. *alaskanus* Hultén is represented as a composite of three sub-parts: a species name, a subspecific epithet and an authority. The species name is itself a composite of three further sub-parts: a genus, a specific epithet and an authority. The genus is also a complex object, and consists of a genus name and (where present) a genus authority.

Authorities are also composite objects, consisting of a basionym authority and an optional derived authority. Every level of a scientific name has an authority associated with it, but it is not always necessary for the authority to appear explicitly within the name. Genus authorities, for example, are commonly omitted, and authorities at some levels in a name are omitted where they can be inferred from information given at other levels.

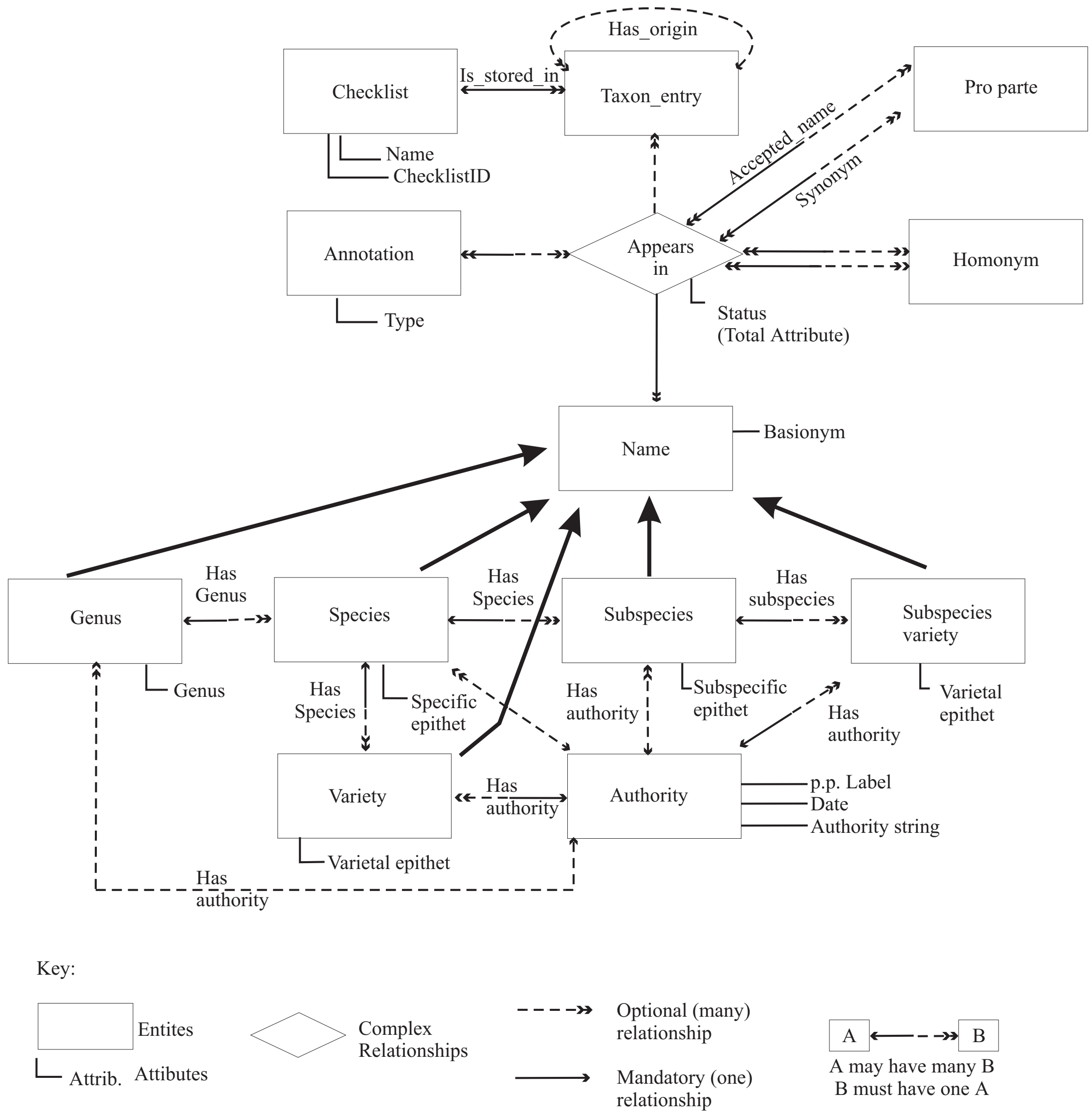

Figure 2. ER Model for the Representation of Species Checklists

The remaining part of the model represents the relationship between checklists and the names they contain. Each name can be related to one or more *entries* within any number of checklists, either as the accepted name of the taxon to which the entry refers or as a synonym of it. The role played by a name in an entry (i.e. accepted name or synonym) is given by the *Status* attribute. We also record details of any additional annotations that have been made to synonyms (e.g. marking whether a synonym is an orthographic variant or a misapplied name).

One important point to note about the ER model is that several of the cardinality and optionality constraints have been deliberately made weaker than the rules of nomenclature would otherwise have suggested. This is a deliberate strategy to allow checklists which violate the integrity constraints to exist, at least temporarily, within the database. If we were to use the DBMS's integrity constraint facilities to enforce this semantics, the user would be required to correct all violations at the time of data entry, and before the full nature of the problem could be appreciated. We prefer, therefore, to allow such violations to remain in the database until the user can examine the situation as a whole and choose the most appropriate repair action.

The ER model we have described here is not a complete representation of all the structural elements of botanical names. Several aspects are known to be missing; for example, we cannot currently store the names of forms, cultivars or hybrids in our database. Other elements of the ER model may be an inaccurate representation of current naming practice. Our aim is to evolve our current model towards a more complete and correct one, by a combination of validation against real data sets and scrutiny by botanical experts.

4. The LITCHI Model of Taxonomic Practice

The ER model described above represents the structure of scientific names and the relationships that can occur between names. However, there is another more dynamic aspect to taxonomic nomenclature: namely, the rules and conventions governing the way that scientific names are used to refer to species (and other taxa) in practice. The hypothesis underlying the LITCHI project is that these rules and conventions can be used to detect inconsistencies in the usage of names between biological databases.

Unfortunately, there is as yet no agreement within the taxonomic community as to what exactly constitutes "best practice" in the use of scientific names. We have therefore taken an iterative approach to the development of our model of taxonomic practice, cycling through the following steps:

- The LITCHI biologists suggest some potential rules informally.
- The LITCHI computer scientists turn these rules into first order logic expressions, and then into Prolog rules.
- The Prolog rules are executed against test data sets, created by importing fragments of real checklists into our DBMS.
- The LITCHI biologists analyse the conflicts and suggest informal modifications to the rules.
- The cycle begins again with the new or modified rules.

It is beyond the scope of this paper to give a full description of our model. We will instead attempt to give a flavour of it by presenting two examples of our rules[6]. The Prolog primitives used in the example rules are:

- `acc_name(latin, auth, clist, taxon)` models the set of names which appear as accepted names for the taxon entries in each checklist known to the system. The `latin` argument is actually a list, in which the different parts of the Latin name are represented. For example, the name *Astragalus alpinus* subsp. *alpinus* would be represented by the list `['Astragalus', 'alpinus', 'subsp.', 'alpinus']`.

  The authority (`auth`) is likewise modelled as a list of its constituent tokens. For example, the authority "(Baker) Popov" would be represented by the list `['(', 'Baker', ')', 'Popov']`.
- `syn(latin, auth, clist, taxon)` models the set of names which appear as synonyms for the taxon entries in each checklist known to the system. As before, the `latin` and authority attributes are also modelled as lists.
- `name(latin, auth, clist, taxon)` models the set of names that appear within the checklists known to the system, without distinguishing whether they are accepted names or synonyms.
- `bas(auth1, auth2)` models the relationship between two authorities when the first authority appears as the basionym authority within the second.
- `taxon(taxon, clist)` models the set of taxon entries in a given checklist.

Each of these primitives is implemented as a query over the relational database schema, which extracts the relevant information in a form suitable for use by Prolog.

Consistency of Names One fundamental rule is that there is a conflict if a full name (that is, a Latin name and an authority) appears more than once in a checklist[7]. A full name which refers to more than one taxon is obviously of limited use as an identifier! The corresponding consistency rule must cover the three cases for comparing the two different sorts of name:

- A full name may not appear as an accepted name and as a synonym in the same checklist:

$$\neg(\exists n, a, l)\ acc\_name(n, a, l, \_) \ \wedge\ syn(n, a, l, \_)$$

- A full name should not appear as an accepted name more than once in the same checklist:

$$(\forall n, a, l, t_1, t_2)\ acc\_name(n, a, l, t_1) \\ \wedge\ acc\_name(n, a, l, t_2)\ \Rightarrow\ t_1 = t_2$$

[6] A full description of an earlier version of the model has been published elsewhere [11].

[7] In reality, this rule is complicated by the presence of certain exceptions, which for simplicity we ignore here.

- In no checklist is a full name a synonym of more than one taxon:

$$(\forall n, a, l, t_1, t_2)\ syn(n, a, l, t_1)\ \wedge\ syn(n, a, l, t_2) \Rightarrow t_1 = t_2$$

The definitions of the Prolog rules that will check for violations of these expressions can be derived trivially from their negated forms. Since checklists are merged into a single list for consistency checking, the resulting rules are parametrised by the identifier L of the merged checklist:

```
conflict_rule(1, L) :-
  acc_name(N, A, L, _), syn(N, A, L, _),
  record_conflict(1, L, [N,A]).

conflict_rule(2, L) :-
  acc_name(N, A, L, T1),
  acc_name(N, A, L, T2), T1 \== T2,
  record_conflict(2, L, [N,A,T1,T2]).

conflict_rule(3, L) :-
  syn(N, A, L, T1),
  syn(N, A, L, T2), T1 \== T2,
  record_conflict(3, L, [N,A,T1,T2]).
```

Consistency of Basionyms Earlier, we saw how the name of a species can change when it is moved from one genus to another. We now present a rule that states that a checklist may not contain both a name and its basionym unless both names refer to the same taxon.[8]

$$(\forall e, a_1, a_2, l, t_1, t_2)\ name([\_, e|\_], a_1, l, t_1)\ \wedge\ name([\_, e|\_], a_2, l, t_2)\ \wedge\ t_1 \neq t_2\ \Rightarrow\ \neg bas(a_2, a_1)$$

This rule states that if any two names applying to different taxon entries (t_1 and t_2) have the same specific epithet (e), then the authority of one of them (a_2) must not be the basionym authority of the other (a_1). By negating this expression, and converting to existential quantifiers, we obtain the following Prolog rule for detecting violations of this rule:

```
conflict_rule(4, L) :-
  name([G1, E | _], A1, L, T1),
  name([G2, E | _], A2, L, T2),
  T1 \== T2, bas(A2, A1),
  record_conflict(4,L,[G1,G2,E,A1,A2,T1,T2]).
```

[8]For convenience, we have adopted a Prolog-like list notation to allow us to extract the subparts of a Latin name in FOL.

5. Representing Taxonomic Conflicts

In addition to storing details of the taxonomic checklists in a database, we have also chosen to use a relational DBMS to record details of the conflicts detected within them. Deciding how to handle some kinds of taxonomic conflict may require consultation with a taxonomic expert who specialises in the study of some particular genus or family. Since this process may involve a lengthy correspondence with a busy scientist in another part of the world, some checklists may take weeks or even months to be fully integrated. The use of a database to store the state of each session means that we do not have to implement extra facilities to support this form of long duration "transaction".

How then do we represent the conflicts which have been detected? In fact, each conflict is characterised by the variable bindings that cause the violated rule to evaluate to false. For example, if we consider the consistency rule given earlier which detects repeated synonyms:

$$(\forall n, a, l, t_1, t_2)\ syn(n, a, l, t_1) \wedge syn(n, a, l, t_2) \Rightarrow t_1 = t_2$$

In a given checklist, a conflict of this rule is characterised by a full name (n and a) and the two taxon entries which share it as a synonym (t_1 and t_2). In order to represent this conflict in the database, we must represent these variable bindings. This is illustrated in Figure 3, which shows the fragment of the ER-model corresponding to this conflict.

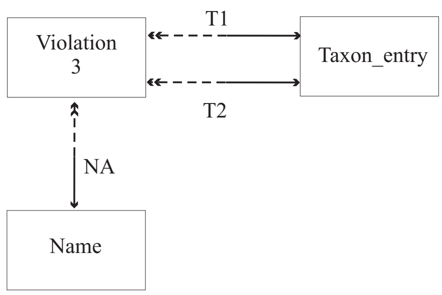

Figure 3. ER-Model for Rule Conflict

Unfortunately, since every consistency rule operates over a slightly different part of the model, each conflict table involves a slightly different set of variables. This means that our database must contain at least one conflict table for each consistency rule. For those rules which can be violated in a number of different ways we require a separate conflict table for each type of violation. The consistency rule that every taxon entry has one unique accepted name is an example of this kind of rule:

$$(\forall l, t)\ taxon(t, l) \ \Rightarrow\ (\exists! n)\ acc\_name(n, \_, l, t)$$

This rule is violated in two different situations. Firstly, it is violated whenever a taxon entry exists which does not have any accepted name; that is, whenever the following condition is satisfied in a checklist L:

$$(\exists t)\ taxon(t, L)\ \wedge\ \neg((\exists n)\ acc\_name(n, \_, L, t))$$

The variable binding which must be recorded in this case is the taxon entry which binds to the variable t. Secondly, this rule can also be violated whenever a taxon entry has two or more accepted names:

$$(\exists t, n_1, n_2)\ acc\_name(n_1, \_, L, t)\ \wedge\ acc\_name(n_2, \_, L, t)\ \wedge\ n_1 \neq n_2$$

This form of conflict is characterised by a taxon entry (t) and its two accepted names (n_1 and n_2). Therefore, this rule requires two consistency classes.

This approach to representing conflicts may appear rather clumsy at first sight. After all, it results in a large number of tables within the database and requires that the database schema be extended every time a new rule is added. However, since each rule represents a unique semantics, it follows that each conflict must also represent a unique semantics and therefore requires a unique representation. Moreover, while the addition of new rules *does* require the addition of new tables to the database schema, this is at least a form of schema update that can be carried out without requiring the entire database to be repopulated. The one real disadvantage of this approach that we have encountered so far is that queries over all conflicts (to find those in which a particular taxon participates, for example) can be awkward. However, this disadvantage can be mitigated by provided "canned query" interfaces for taxonomists which hide the extra complexity of such queries.

6. Implementation of LITCHI

We have chosen a repository-style architecture for LITCHI, illustrated in Figure 4 in which a central database is used to communicate information between a number of different software components. Three software components, each providing a different class of service, are required:

- The Data Import/Export Function (DIEF): this component provides services for importing taxonomic data into and exporting taxonomic data out of the database in a range of different formats. The services in this component are currently implemented using Visual Basic.
- The Conflict Reasoning Engine (CRE): this component provides services for merging checklists stored in the database and for searching for conflicts or other potential problems within checklists. This component is implemented using the Prolog language, connected to the central database through the ProData system [7]. ProData allows Prolog queries to be evaluated over a relational database, via an ODBC interface.
- The Interface for Taxonomists (IfT): this component acts as the front-end for the entire LITCHI system. It allows the taxonomist to invoke services provided by the other components, and to examine their results as represented by the contents of the database. The current version of this component is implemented using Visual Basic.

The architecture shown in Figure 4 has several advantages:

- It facilitates implementation of the range of services required for LITCHI by allowing different languages to be used for different components. For example, the IfT (which provides GUI services) is implemented in Visual Basic, while the CRE (which must be able to reason over FOL expressions) is implemented in Prolog.
- The use of a central database removes many of the difficulties inherent in communicating potentially large data sets between services. Services typically operate on complete checklists, which may contain thousands of species. Using a central database to store and manage this shared data means that services do not need to continually create and update large files or complex in-memory data structures.
- The central database model acts as a common interchange format, into which taxonomic data originating from different sources, in different formats, can be converted. The advantage of this over a common interchange *language* for our purposes is that the imported data can be queried using standard DBMS querying facilities.

7. Conclusions and future work

We have described an approach to the problem of detecting data-level semantic conflicts in databases which are to be integrated. This is a significant problem for

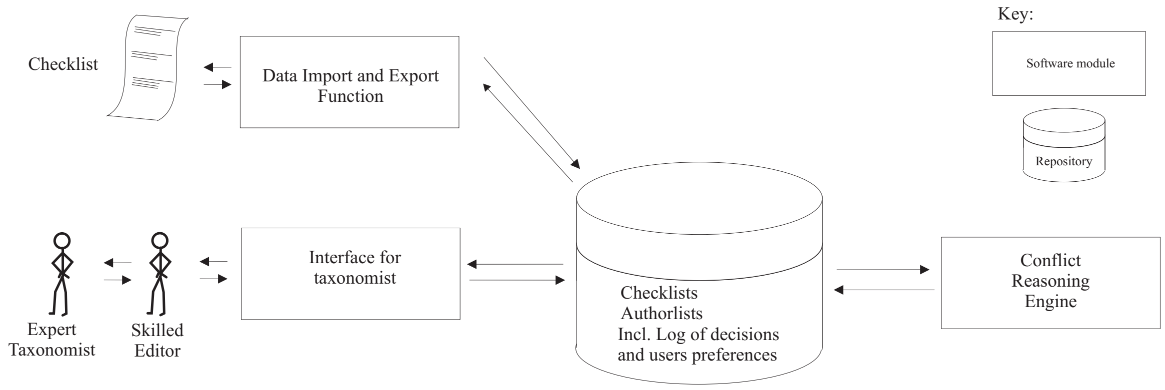

Figure 4. The LITCHI Architecture

those who wish to construct federations of biological information sources, since species names are often the key attribute by which the information in the component databases is linked together. Differences in the meanings of such names must therefore be identified prior to integration so that the necessary query transformations can be made to ensure that all relevant data is retrieved.

We have constructed a prototype system which makes use of a formal model of observed taxonomic practice to identify potential conflicts between taxonomic checklists. Whilst our model is as yet incomplete, the LITCHI system has already been able to produce some useful results. For example, sets of conflicts generated from several checklists have been analysed by the LITCHI biologists, and have led to refinements of the formal model. In particular, our most recent test compared 5808 names in the tribe Galegeae from the ILDIS World Database of Legumes with 1908 names from the Legumes of Northern Eurasia database. This test resulted in the detection of nearly 1500 potential conflicts, as shown in Table 1.

The largest number of conflicts were detected by a warning rule which checks that a particular kind of synonym is correctly annotated. Many of these warnings are caused by the fact that the creators of different lists use different conventions for abbreviating the names of authors — another example of a data-level semantic clash! Misspelling of author names on data entry is also common, and is responsible for many violations of rule W25. However, some of the violations may be caused by genuine differences of taxonomic opinion, which must be resolved by a taxonomic expert.

Of the other rules, all but a handful of the conflicts represent either inconsistent name usage between checklists or compatible cases which our formal model is as yet unable to recognise. However, seven of the conflicts (all detected by rule C27) were of particular interest to the LITCHI taxonomists since they were found to be errors in the original ILDIS data sets, and not to be caused by the integration of two independently derived checklists. For example, the following fragment of the ILDIS checklist exhibits a violation of the rule that no full name may appear as both an accepted name and a synonym within the same checklist:

```
Astragalus refractus Boiss. & Buhse (accepted name)

Astragalus buhseanus Bunge (accepted name)
   Astragalus askius Bunge var. buhseanus Boiss.
                                          (synonym)
   Astragalus refractus Boiss. & Buhse (synonym)
```

The presence of these conflicts is both interesting and surprising. The software which was used to input the original checklist data is designed to forbid the operator from introducing such errors into the database. Moreover, a considerable amount of effort has been put into inspecting the ILDIS data manually for errors such as these. One potential explanation is that the conflicts were introduced early in the life of ILDIS as a result of merging data from a third party source that had not been fully validated. However, it is clear that no matter how much effort and talent is put into manual verification of checklists, the sheer size of the data sets means that errors will inevitably creep in. Although originally designed for detection of conflicts *between* checklists, it appears that the LITCHI software may also have a useful role to play in data cleansing of individual checklists, prior to integration.

As we have said, our model of Botanical Nomenclature is as yet incomplete, and we must therefore continue to work with the LITCHI prototype in order both to test out suggested new rules and to further test the existing ones. In particular, we have only just begun to explore the semantics of subspecies and varieties, and this is therefore one obvious area for future work.

Rule	Rule Description	Conflicts
C27	A full name may not appear as both an accepted name and a synonym in any given checklist.	43
C4	A full name may not appear as the accepted name of more than one taxon in any given checklist.	372
C5	A full name may not appear as the synonym of more than one taxon in any given checklist.	112
W25	Within any given checklist, two names may not contain the same Latin components (but different authorities) unless labelled as a homonym or a misapplied name.	812
C26	Within any given checklist, every full name which is indicated to have been misapplied by the form of its authority must be labelled as a misapplied name, and *vice versa*.	32

Table 1. Summary of Conflicts Discovered by LITCHI

We also intend to widen our formal model to include the subtly different conventions used in zoological and bacteriological nomenclature. In doing so, we will have the opportunity to test the generality of our prototype architecture, and the ease with which new rules sets can be added to it.

However, the next major step for the LITCHI project is the introduction of facilities which support the taxonomist user in resolving the conflicts that are detected. For example, we would like our system to be able to distinguish between conflicts which can be repaired automatically without human intervention and those which require the judgement of a taxonomic editor for their resolution. There is also scope for automatic generation of "suggested" repairs, based on an analysis of the violated constraints. Our long-term goal is to provide a software tool which will support the taxonomist throughout the entire process of integrating checklists, from the initial detection of conflicts through to their satisfactory resolution.

References

[1] C. Batini, M. Lenzerini, and S. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, Dec. 1986.

[2] R. Froese and D. Pauly. *FishBase 98: Concepts, Design and Data Sources.* International Center for Living Aquatic Resources Management, MCPO Box 2631, 0718 Makati City, Philippines, 1998. `http://www.fishbase.org`.

[3] W. Greuter, F. Barrie, H.-M. Burdet, W. Chaloner, V. Demoulin, D. Hawksworth, P. Jorgensen, D. Nicolson, P. Silva, P. Trehane, and J. McNeill. *International Code of Botanical Nomenclature.* Koeltz Scientific Books, 1994.

[4] C. Jeffrey. *Biological Nomenclature.* Edward Arnold (Publishers) Ltd., third edition, 1989.

[5] D. Karunaratna, W. Gray, and N. Fiddian. Establishing a Knowledge Base to Assist Integration of Heterogeneous Databases. In S. Embury, N. Fiddian, W. Gray, and A. Jones, editors, *Advances in Databases: Proceedings of 16th British National Conference on Databases (BNCOD16)*, pages 103–118, Cardiff, U.K., July 1998. Springer-Verlag.

[6] J. Lock and K. Simpson. *Legumes of West Asia.* Royal Botanic Gardens Kew, London, 1991.

[7] R. Lucas. *ProData Interface for SICStus Prolog.* KeyLink Computers Ltd., Sept. 1996. Version 4.

[8] E. Sciore, M. Siegel, and A. Rosenthal. Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems. *ACM Transactions on Database Systems*, 19(2):254–290, June 1994.

[9] A. Sheth, S. Gala, and S. Navathe. On Automatic Reasoning for Schema Integration. *International Journal of Intelligent and Cooperative Information Systems*, 2(1):23–50, 1993.

[10] N. Singh. Unifying Heterogeneous Information Models. *Communications of the ACM*, 4(5):37–44, 1998.

[11] I. Sutherland, S. Embury, W. Gray, A. Jones, F. Bisby, S. Brandt, J. Robinson, and R. White. Knowledge Integrity Testing for the Integration of Taxonomic Databases. In D. Gilbert, editor, *Proceedings of the Second Workshop on Constraints in Bioinformatics/Biocomputing*, Pisa, Italy, Oct. 1998.

[12] G. Yakovlev, A. Sytin, and Y. Roskov. *Legumes of Northern Eurasia.* Royal Botanic Gardens, Kew, London, 1996.

[13] J. Zarucchi, P. Winfield, R. Polhill, S. Hollis, F. Bisby, and R. Allkin. The ILDIS Project on the World's Legume Species Diversity. In F. Bisby, G. Russell, and R. Pankhurst, editors, *Designs for a Global Plant Species Information System.* Oxford University Press, Systematics Association, 1993. Special Volume No. 48.