Database-backed decision trees with application to biological informatics

Robert A. Morris[1], Jacob K. Asiedu[1], William Haber[2], Fred SaintOurs[1], Robert D. Stevenson[1] and Hua Tang[1]

Abstract

We describe a mechanism for the identification of biological organisms through the use of enhanced taxonomic keys–decision trees with nodes augmented by property lists that can serve as arguments to web or local services that access databases or other resources about species, specimens, and ecosystems. Authors of these identification schemes can use simple spreadsheet tools to structure the identification abstractions, and middleware renders the resulting trees into many different forms, with the databases possibly discovered and queried at the time an identification is proposed.

Introduction

Cataloging the diversity of life and understanding the relations among organisms is a major goal of the biological sciences (Wilson and Peter 1988; 92; Reaka-Kudla, Wilson, and Wilson 1997). The current cataloging system is based on a hierarchical classification using a binomial (genus and "specific epithet" such as *Quercus alba*) to name species (Winston 1999). The system was designed by Linnaeus in the middle of the 18th century and is defined by International Codes of Nomenclature (ICZN 2005; ICBN 2000; Brickell et al. 2004; Sneath 1990; also see Winston 1999). While this system is currently being challenged by philosophical considerations and phylogenetic analyses using molecular data (Ereshefsky 1991; Ereshefsky 2001; Wheeler and Meier 2000), in practice descriptions of new species are still dependent on the Linnaean system. Taxonomists and field naturalists concerned with identifying organisms in the field or laboratory use decision trees called *(taxonomic) keys* which aid this identification (Pankhurst 1991; Winston 1999). Traditionally, these keys are binary trees ("dichotomous keys"), but when they are not, biologists sometimes call them *polychotomous keys.* Although familiar identification aids decide which species is under study, some taxonomic keys provide identification of larger taxonomic groupings, such as Genus or Family. The generic term for such a grouping is *taxon* (plural *taxa*). In this paper, we describe a method for the convenient generation of database-backed decision trees that has proved particularly useful for easing the task of the generation of polychotomous keys whose nodes support queries into descriptive or other specialized biological database software. In our own Electronic Field Guide software (Stevenson et al. 2003, and also [editors: we need to cite our other paper in this issue here]) we have used it to support multimedia polychotomous keys where the identification produces a descriptive taxon page detailing much of the life history, ecological data, and references to taxa that might have been confused with the one selected

Property List Decorated Graphs

We describe a data structure that finds its origins in the Association List (alist) of Lisp and has some similarity to those of the Entity-Attribute-Value (EAV) structure introduced by Marenco et al. 2003). Our data structure consists of an arbitrary graph with a collection of property lists, i.e., lists of key-value pairs, at each node. Although we call this a *Property List Decorated Graph* (PLD Graph), the property lists at each node have essentially the structure of an alist or a Java Map in that they require only a function on keys that returns unique values and a function that

[1] University of Massachusetts at Boston
[2] Missouri Botanical Garden

determines equality on values. Keys and values may be arbitrary objects in an object system realizing the PLD Graph. To avoid ambiguity in the use of the word “key”, we refer to identification keys as *taxonomic* keys, whether they are dichotomous or polychotomous and refer to the selector in the key-value pair as an *attribute* rather than a key, even though in traditional databases an attribute may not be structured. Our main applications lie in the field of biodiversity informatics and these are well handled with Trees or Directed Acyclic Graphs. As a consequence, in these cases we can describe our abstraction using an XML-Schema (W3C 2004b).We will focus on this case, but in a later section will describe the relationship to the EAV structure mentioned above. Our central applications are taxonomic keys where the property lists support queries into biodiversity and image databases on the web or locally.

In our case, biologists in the role of taxonomic key authors build the keys using Microsoft Excel, representing the nodes as rows, child relations in the graph by the choice of initial column, and the attribute-value pairs as Excel comments on a cell. A more extensive description is below. PLD Graphs have no inherent semantics. Applications are free to interpret them as they see fit. However, we can assert syntax rules that help applications validate a PLD Graph if the representation language supports such rules. In the case of XML the rules are an XML Schema, and in the case of Java they are a Java Interface. We enforce structure on an Excel representation by converting it to XML conforming to a particular Schema using a Java program whose success is the definition of the conformance of the Excel file. An informal set of rules for the production of the spreadsheet is available to the taxonomic key author, conformance to which suffices to allow correct application of the Java program. A Visual Basic addin to Excel provides a dynamic property list editor which constrains the Excel comments to syntactically conforming representations of the property lists. We focus here on hand-crafted taxonomic keys, but it is very common to dynamically construct keys from tables of attributes in which rows represent taxa. One common method allows a database user to select from attributes that most nearly partitions the taxa into partitions of equal size, and repeat this recursively on the remaining taxa after the choice is made. This strategy, sometimes called a *multi-access key*, exploits the order-independence of attributes not available in the static trees we focus on here. These taxonomic keys could also fit into our scheme if a mechanism was provided to attach the property lists, but we have not done so. Tables supporting this kind of representation are often called *taxon-by-character matrices* and are widely used in phylogenetic analyses not for identification but rather to represent hypotheses about evolution (Maddison and Maddison 2005). For a survey of taxonomic key software, see Dallwitz (2000).

Taxonomic identification keys and the generation of XML representation of PLD Graphs

Taxonomic keys abound in print and on the web, and biology courses frequently include lessons on how to produce them. While there are several software programs for their production and use (Dallwitz 2000, we are aware of no abstractions from which they can be automatically or semi-automatically produced in a variety of ways from a single data structure, as we do in the work described in this paper. Perhaps the best approximations to such structures are exchange languages for identification software. The oldest of these the DELTA language, dating to 1980 and now in its fourth incarnation (Dallwitz et al. 1993), and the newest is the emerging standard for the Structure of Descriptive Data (TDWG 2005c) of the Taxonomic Data Working Group (TDWG 2005b). In fact, we will describe how we can turn a single PLD Graph defined by an author into one of several renderings of taxonomic identification keys. When a decision is reached, applications can represent that description with as little or as much descriptive detail as

the application warrants, ranging from a simple standard name representing the decision, to a complex, database-driven description of the object identified.

Queries as Properties

The main use to which we have so far put PLD Graphs is the production of multimedia taxonomic keys. At terminal nodes the property list supports a query into a web database that provides data about a taxon represented by the decision at the node. Examples are

attribute="http://efg.cs.umb.edu:8080/efg/servlet/search?"
value="scientificName=Mechanitis+lysimnia"

attribute="http://efg.cs.umb.edu:8080/soap/rpcrouter/getSpecies method=POST&"
value="scientificName=Mechanitis+lysimnia"

By use of more abstract properties, more sophisticated examples are possible in which the application uses dynamic mechanisms to formulate discovery, access and rendering of desired data, with the property simply being

attribute="scientificName"
value="Mechanitis+lysimnia"

In the first two examples, a taxonomic key author is choosing to build a key into a specific database, with a specific query protocol known in advance to her. This supports web applications that merely have to concatenate the keys and values. In the third the author delegates to the interpreting application the task of discovering and rendering the resulting species description. Some of our current applications query databases through interfaces that can return HTML (Figure 4) or can return XML allowing the querying application to render it into HTML or pass it to a collaborating application.

Structured PLs

As with Lisp alists, Java Maps and recent versions of EAV, the values in a property list can be arbitrary objects and the attributes are limited only by the requirement that each attribute determine a unique value and that equality among values be testable. In practice, the generality we support depends on both the target application and the support available in the representation language used by the PLD Graph author. Because spreadsheets, which have limited data structuring capability, are particularly familiar to taxonomic key authors, we largely limit ourself to unconstrained textual data for attributes. However, in the following section we describe a convention by which authors can structure values. An author or group of authors willing to use XML editing tools could provide a common XML Schema for their PLD Graphs. Such a Schema can drive tools for rendering graphs in domain specific applications, as we discuss next. By using a less general Schema than we do, application writers could provide metadata to tailor their look-and-feel more specifically to requirements of their applications.

Tools for PLD Graph generation and rendering

Although graph generation tools are available (e.g. Gansner and North 2000, and the graph package of the Diva Project 2005) we chose to focus on tools familiar to biologists, especially those practiced at generating taxonomic keys. For this reason, we established a set of conventions for using Excel spreadsheets for PLD Graph representation. The transformations we support to render the resulting graphs essentially limit the graph structure to trees. In this section, we will describe how we use these tools to generate several different renderings of the taxonomic tree represented abstractly by a given PLD tree in ways that are independent of the tree data. We will

refer to the programs that do this as the EFG Key Rendering Suite. The Suite is specifically designed to produce taxonomic keys whose final identification can easily produce a query against an internet-accessible database and use the result to produce a detailed description of the identified taxon, rendering that either in HTML or XML for forwarding to other applications. Our particular applications include queries into systems generated by our own Electronic Field Guide software (Stevenson et al. 2003 and [editors: we need to cite our other paper in this issue here]) and into portals and providers for the DiGIR (2005) protocol under continuing development for the exchange of biological collection data such as museum specimen records (TDWG 2005a)

The author begins by entering a decision criterion in plain text in a row of the spreadsheet. Such a criterion is called a *character* by biologists and is often some visible attribute of the class of objects being described. More precisely, the author typically enters one possible state of the character (e.g. “HW with dark central bar or spot: Horizontal bar present”, to indicate the state of a particular part of the hind wing of a butterfly; See Figure 3) and enters other values (e.g. “Dark spot present”) in the same column in a subsequent row. These cells, and data in some horizontally adjacent ones, represent each of the siblings which begin in a given column. In practice, this limits the graph structure to a tree, although we’ll see shortly that terminal nodes can represent the same decision. As children of a node are entered, rows are inserted with all the children starting in the column after that in which their parent starts. For each node, one or more adjacent cells in the same row contain text whose precise meaning is determined by the transformer that renders the representation into an identification application. Our current conventions support two cells in each row, one for the character state and one for a name—if there is one—of the group of organisms selected by the given state.[3]

Figure 3 shows a portion of the spreadsheet giving rise to the tree fragment in Figure 1. Omitted from Figure 3 are most of the Excel comments implementing the property lists shown in the corresponding XML in Figure 2. The yellow box at the upper right is one such, not normally exposed to key authors, who instead use a menu-driven Visual Basic Excel addin, shown invoked at the bottom of Figure 3. This enforces the (possibly) complex structure of the property lists and remembers the property attributes and values across editing sessions. Terminal nodes (not shown) carry a property named “Scientific name” whose value our main application uses to construct database searches for further information about the named species (Figure 4). In the current Schema for our XML representation of PLD graphs, property lists are contained in a more complex wrapper called a Description. On each node, there may be multiple Descriptions, hence multiple property lists, which are thereby are given a context by their Description. A Description is typed as either a Name or a Narrative. A Description of type Name essentially is a label and (optional) property list whose ultimate purpose (aside from presentation of the label by user interfaces) is to provide, in the future, for the representation of more complex directed graphs by the addition of a locally unique identifier for internal references. A Narrative Description, along with its property list, is intended to describe the decision and provide for such things as glossary terms and character state illustrations. The latter is accomplished presently by a property that carries an XML attribute “kind”, taking one of several multi-media types as value, and whose attribute-value pair has data to aid in the access to the media. Both Description and Property objects can carry the author’s advice about which of multiple instances might be preferred if an application can only use a singleton. In general, attached to spreadsheet cells are optional Excel comments supporting the entry of attribute-value pairs in the form

[3] The convention could easily be extended to represent arbitrary graphs by using one cell for a node ID and another for a reference to another node. In fact, trees and directed acyclic graphs serve key authors quite well, and the DAGs are most typically simply decision trees with multiple paths to the same decision, which, we will see, is adequately represented by the conventions under discussion.

attribute=<attribute text>
value=<value text>

Rendering transformers, such as those we provide, are free to interpret these as they wish, but our present suite of transformation software places some syntactic and semantic restriction on the attribute and value text. Several strings are reserved for multimedia designations in the attributes. These presently include the tokens “IMAGE”, “VIDEO”, and “AUDIO”. For these attributes the values designate URIs (typically simply filenames, but also possibly queries into an image database) at which the media can be fetched by the application rendering the taxonomic key. In addition, several reserved strings serve to structure a collection of attribute-value pairs. These are

attribute=efg:Structure
value=efg:Structure:start#*name*

and

attribute=efg:Structure
value=efg:Structure:end#*name*

where *name* is a string chosen by the author, but restricted to strings that are of XML NMTOKEN type, i.e. meet the requirements of an XML name token. These two attribute-value pairs, which can be used recursively, serve to group the attribute-value pairs contained within them and are required to be properly balanced. Such structured properties tend to appear more at leaf nodes—where an identification decision has been reached—than at interior nodes. At the latter, authors typically provide simple media designation properties for the specification of illustrations of the character state represented by the node. Some of our interactive applications present these as navigation aids, with the character state text as a mouseover aid, and some present both media and text in tabular form.

Here is an example of a structured property list expressed as an Excel comment, in which the attribute “efg:Structure” serves to associate a species name, with a choice of sex. This permits the author to advise applications seeking species information that the male and female data may be different. Indeed, our own key deployment application, using the XML following the Excel comment, generates a query clause “ScientificName = Mechanitis lysimnia AND Sex = Male”. Thus

attribute=image
value=EFGImages/Mech-lys-md-509.jpg
attribute=efg:Structure
value=efg:start#TaxonReference
attribute=Taxon#Scientific name
value=Mechanitis lysimnia
attribute=Taxon#Sex
value=Male
attribute=efg:Structure
value=efg:end#TaxonReference

is rendered in XML as

```
<Properties>
    <Property kind="IMAGE" preferred="true">
        <propertyName>MediaResource</propertyName>
        <propertyValue> EFGImages/Mech-lys-md-509.jpg</propertyValue>
    </Property>
    <efgStructure>
        <Property>
```

```
            <propertyName>Scientific name</propertyName>
            <propertyValue> Mechanitis lysimnia</propertyValue>
        </Property>
        <Property>
            <propertyName>Sex</propertyName>
            <propertyValue>Male</propertyValue>
        </Property>
    </efgStructure>
</Properties>
```

Such XML induced a call to our descriptive database, and our middleware created the page shown in Figure 4 from XML returned by our service calling the database.

Key Deployment

To use the EFG Key Rendering Suite the author begins by saving the Excel spreadsheet in its native XML form.[4] A Java program of our devising then converts this to XML conforming to an XML Schema designed to be more convenient than the native Excel schema (mainly compensating for some minor quirks in the former, together with parsing the unstructured property list text in the Excel comments and rendering it as recursively structured XML). We call the result the *PLD file associated with the taxonomic key*. In fact we regard the XML Schema as the most abstract taxonomic key representation in the current EFG Key Rendering Suite and plan to make user-friendly applications that will directly produce and manipulate it, while preserving the familiar spreadsheet metaphor.

There are several advantages of generating XML files against a data-independent schema, all surrounding current XML technologies that we are exploiting. One is that it eases the task of generating rendering software in the XSL languages XSLT (W3C 1999 and XSL-FO W3C 2005). These programs thus become generators of renderings of various kinds, of which we have so far produced taxonomic keys rendered in HTML, keys rendered in the Wireless Markup Language (WML) to provide a text-only taxonomic key suitable for use in a web-enabled mobile phone or PDA, and keys rendered in PDF suitable for paper publication. Our application front-end provides for downloading the HTML renderings for use in devices not running servers, such as handheld computers that might be taken into the field. Each of these, and the Java applet tree browser illustrated in Figure 1, are produced on demand from a single PLD file.

XSLT and XSL-FO are somewhat specialized and difficult or impossible to use where XML-related languages are not the target of the rendering. In particular, long preceding the current work, we supported one kind of multi-media key as a Java applet. As mentioned above, in this rendering the interior nodes have pictures that are illustrative of the character state at the node. The applet presents a tree browser (Figure 1) essentially the same as a traditional file browser but with the given illustrations in place of a folder or document icon at each node. Based on the PLD Schema, we now use the Castor Java Databinding Framework (Exolab 2005) to generate the Schema dependent portions of an applet that uses the PLD file as input to provide the location of the character state images and the arguments to database queries that provide taxon descriptive pages at the leaf nodes. Generating, rather than handcrafting, the code that consumes the PLD file provides us improved robustness as well as insulation from evolution of the PLD schema. Bearing in mind that the PLD Schema is itself a particular realization of the abstract PLD, our strategy supports generation of these applets from any such realization that is expressible as an XML-Schema.

[4] This requires recent versions of Excel, from the Office XP suite or later.

Finally, we remark that when XML data conforms to an advertised Schema, certain Web Service clients can be partially automated (See for example the Apache Web Services Invocation Framework, WSIF (Apache 2005), and corresponding facilities in Microsoft .NET (Microsoft MSDN 2005). Some applications in our lab make use of WSIF and we expect that the Key Rendering Suite will ultimately be folded into those. Presently, the Suite is implemented as a collection of servlets executing under the Tomcat servlet engine (Apache Jakarta 2005)

Applications

Here we briefly describe several taxonomic keys that we have produced using the approach described above. They include a key to the plant families of the Monteverde, Costa Rica Cloud Forest, keys to the Ithomid family of butterflies ("Clearwings") of Costa Rica, a key to the aquatic invertebrates of Massachusetts, and a winter key to the invasive plants of New England. The plant family key is unusual in that many published keys assume that the user can distinguish one family from another, which is rarely true of inexperienced observers. The Ithomid butterflies are a beautiful family that ranges from vividly colored species to species parts of whose wings are, literally, transparent. Some species are particularly difficult to distinguish from one another. Aquatic invertebrates are a diverse group which can often be identified only to the Genus level using conventional field or laboratory instruments. They are important worldwide because they are often indicators of the health of watersheds and other ecosystems. The invasive plants key is unusual not only for being seasonal[5], but also because its leaf nodes generate queries not to descriptive species pages, but rather to an image library. Other projects under design include keys and descriptive data for tropical ants and keys to aquatic invertebrates of Costa Rica.

One appeal of our approach to some our naturalist collaborators is the orthogonality it yields between the key and the databases which can be automatically queried for further information when an identification is reached (including for confirmation that the identification is correct). To configure a key for a particular web-accessible database we need only record some query templates that can be read by the applications generated by the generation suite. Front ends can be easily built that offer the user a mix-and-match collection of keys and databases.[6] To date we have generated taxonomic keys that query databases implemented in ObjectStore (a commercial OODB), FileMaker Pro 6 (a commercial low-cost relational database favored by many biologists) and the Open Source RDBMS MySQL.

The Visual Basic Excel comment manager we provide makes it quite easy to produce taxonomic keys using pre-existing data for attributes and values or to develop that data for a new project. Keys can be easily modularized by a rendering application invoking a subkey using some particular convention about properties. In our Key Rendering Suite, a terminal node with a property { attribute=URL, value=<name> } invokes a local key named by <name>. Deploying corrections to a key using the Suite is trivial: one merely uploads the saved Excel file to a server, which notices that a new version is available and produces the PLD file. The actual rendering is done on the first client request for a particular rendering of the updated key, and that is cached for performance enhancement. Although we have done no formal evaluations at this writing, we have held informal and formal workshops for key authors. In a formal workshop, field naturalists each produced a useful key to the ten Massachusetts native turtles with an hour of instruction. They were given images, previously stored on our server, which they organized according to characters

[5] In work now underway, we are designing keys that can be dynamically restricted to time and place.
[6] In work in progress we are designing mechanisms by which taxonomic keys can transparently discover appropriate data sources and query templates through registry mechanisms such as UDDI (2005). This includes generating DiGIR queries at leaf nodes.

of their choosing and built and deployed picture-based keys with about an hour of work. We believe that working in a field of their expertise, most biologists could produce a key of this size in a few minutes, and more complex ones in a matter of minutes per node. The present software lacks “drag-and-drop” image management, but has a limited configurable image selection and preview facility that relieves authors of some of the tedious work of assigning images to illustrations of attribute values.

Relation to Prior Work

Lisp association lists and the Java Map interface each support the expressive power of the data structure on which we base the work described here. We are unaware of explicit applications that depend on extracting property lists from the nodes of graphs for database queries, but it would hardly be difficult to do in either language. Although more complex than our current XML, the Resource Description Framework, RDF (W3C 2004a), has a similar purpose and it is straightforward to transform a PLD file into RDF.

Each of our renderings is, in essence, a visualization of the PLD Graph for a specialized audience. Among the most powerful graph visualization systems is GraphVis (Gansner and North 2000). GraphVis provides a formal graph language and tools to generate representations of graphs, especially for visualization. GraphVis has been used to generate tools for the visualization of RDF.

Related work whose design goals most nearly intersect our own is the EAV/CR (“Entity-Attribute-Value with Classes and Relationships” system (Marenco et al. 2003) designed in the Yale Center for Medical Informatics and applied in SenseLab, a project which is part of the National Human Brain Project (HBP). (We became aware of this work when we recently began designing applications to auditory neuroscience in another part of HBP EarLab). The EAV/CR data structure is conceptually an unordered list of triples (*e*,*a*,*v*) in which *e* represents on object of scientific interest, *a* a named attribute and *v* a possibly structured value. If a list of references to other entities were attached to each entity, then an EAV/CR dataset would represent a PLD graph with edges given by those references. The main purpose of the EAV/CR is to carry metadata along with data representations, together with information suitable for automatically generating applications against the data. Part of the Yale project’s purpose is to permit the data to be self-describing. For us to do so to the extent that EAV/CR does, we would need to communicate the PLD XML-Schema along with any given PLD file. That strategy is not unusual in XML data interchange mechanisms, but there is not presently any standard way of doing so, making it hard to design agent software to exploit it with only query-time Schema information (Skonnard 2005).

Future Work

Much of the motivation for our underlying Schema in the XML representation was the representation in an early version of SDD (TDWG 2005c) of “authored keys” which, albeit without property lists, was in turn based on a presentation we gave at a TDWG annual meeting. It was not difficult to generate SDD authored keys from our own Schema in the early versions. As this article goes to press, the SDD schema has been finalized and formally proposed as a TDWG standard, the taxonomic keys now being called “StoredKeys” to distinguish them from keys generated from tables or matrices of characters and taxa. We expect to extend the StoredKey data type by property-list support to make it particularly easy to exchange taxonomic keys with those of other SDD conformant systems.

A central goal in our current work is the production of sub keys on demand, localized to season and location or meeting other constraints, such as restrictions to specific groups of species. This

requires dynamically extracting sub-graphs based on the constraints, which may be expressed in the property lists or provided by an external ontology.

Acknowledgements

The work was partially supported by Grants DBI-0113058 and DBI-0111540 from the US National Science Foundation to Morris and Stevenson and a subcontract to Morris on a Human Brain Project grant R01 DC04731 from the National Institutes of Health to Prof. David C. Mountain, Boston University Department of Biomedical Engineering. Haber is the author of the keys to Costa Rican butterflies and plants. SaintOurs is the author of the key to Massachusetts aquatic invertebrates. Additional thanks to Jennifer Forman, author of the winter invasive plants key, for critique of some of the keys and assistance with the aquatic invertebrate key. The Java applet key is derived from code originally produced by Matthew Passel supported in part by NSF Grant DBI-9808462 to Stevenson and Morris.

Figures

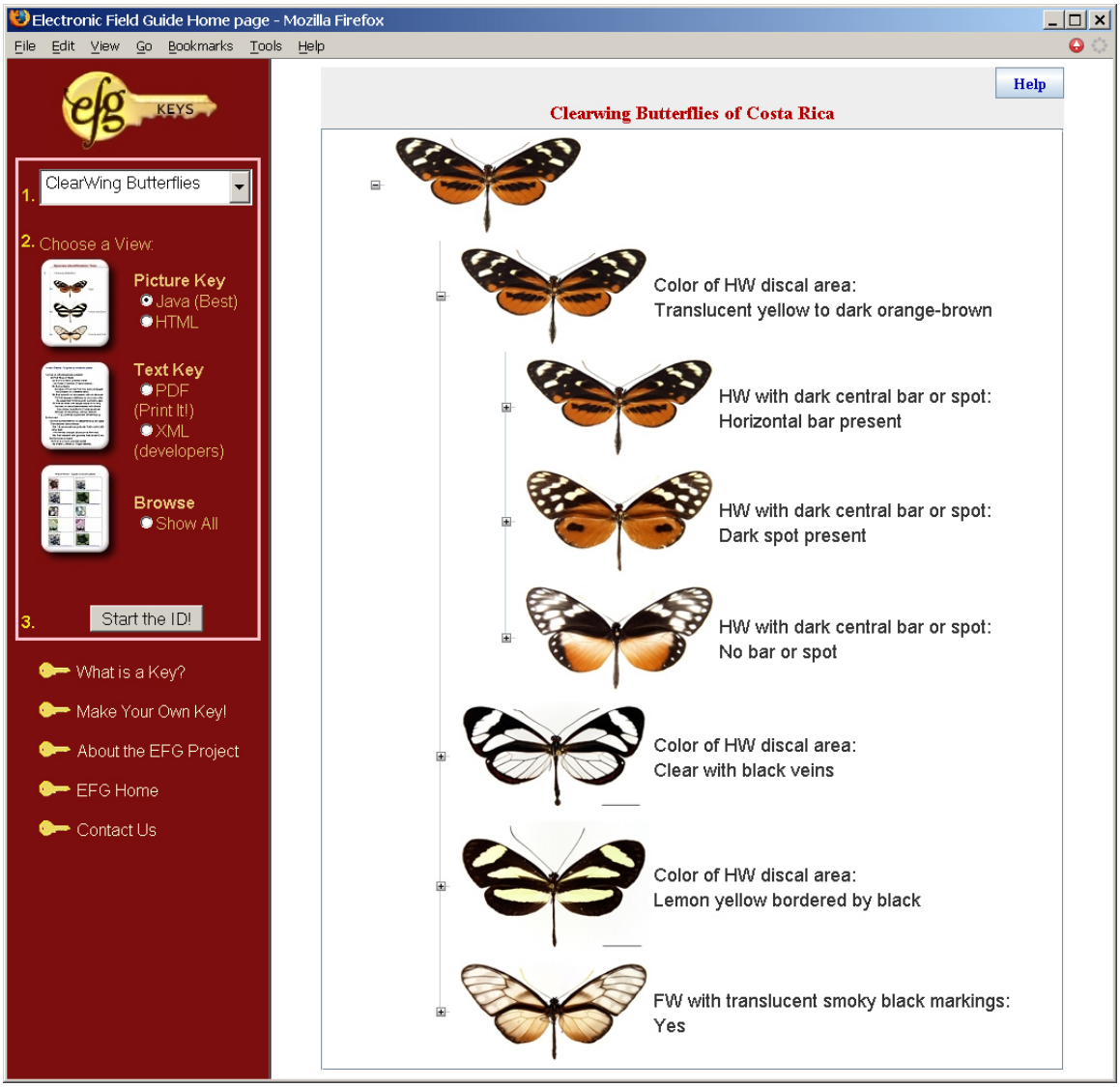

Figure 1. Portion of Java Applet tree browser showing choices offered user for identification of Ithomid butterflies of Costa Rica. Selecting "+" opens the display to show its child nodes. Other tree visualizations are supported without any editing required of the taxonomic key author. The tree nodes may be optionally named. In this application, the name drives a mouse-over that gives additional information about the choice.

```
<VisualKeyNodes>
    <VisualKeyNode>
        <Descriptions>
            <Description descriptionType="Narrative">
                <text>HW with dark central bar or spot: Horizontal bar present</text>
            </Description>
            <Description descriptionType="Name">
                <Properties>
                    <Property kind="IMAGE">
                        <propertyName>MediaResource</propertyName>
                        <propertyValue>Mech-poly-md-509-184.jpg</propertyValue>
                    </Property>
                </Properties>
            </Description>
        </Descriptions>
        <VisualKeyNodes>
            <!- - subtree - - >
        </VisualKeyNodes>
    </VisualKeyNode>

    <VisualKeyNode>
        <Descriptions>
            <Description descriptionType="Narrative">
                <text>HW with dark central bar or spot: Dark spot present</text>
            </Description>
            <Description preferred="true" descriptionType="Name">
                <Properties>
                    <Property kind="IMAGE" preferred="true">
                        <propertyName>MediaResource</propertyName>
                        <propertyValue>Hypo-lyca-md-526-184.jpg</propertyValue>
                    </Property>
                </Properties>
            </Description>
        </Descriptions>
        <VisualKeyNodes>
            <!—subtree –>
      </VisualKeyNodes>
    </VisualKeyNode>
    <VisualKeyNode>
        <Descriptions>
            <Description descriptionType="Narrative">
                <text>HW with dark central bar or spot: No bar or spot</text>
            </Description>
            <Description preferred="true" descriptionType="Name">
                <Properties>
                    <Property kind="IMAGE" preferred="true">
                        <propertyName>MediaResource</propertyName>
                        <propertyValue> Hyal-exce-md-504-184.jpg</propertyValue>
                    </Property>
                </Properties>
            </Description>
        </Descriptions>
        <VisualKeyNodes>
       <!—subtree –>
      </VisualKeyNodes>
    </VisualKeyNode>
</VisualKeyNodes>
```

Figure 2. Portion of XML driving the rendering shown in Figure 1. See text for further details.

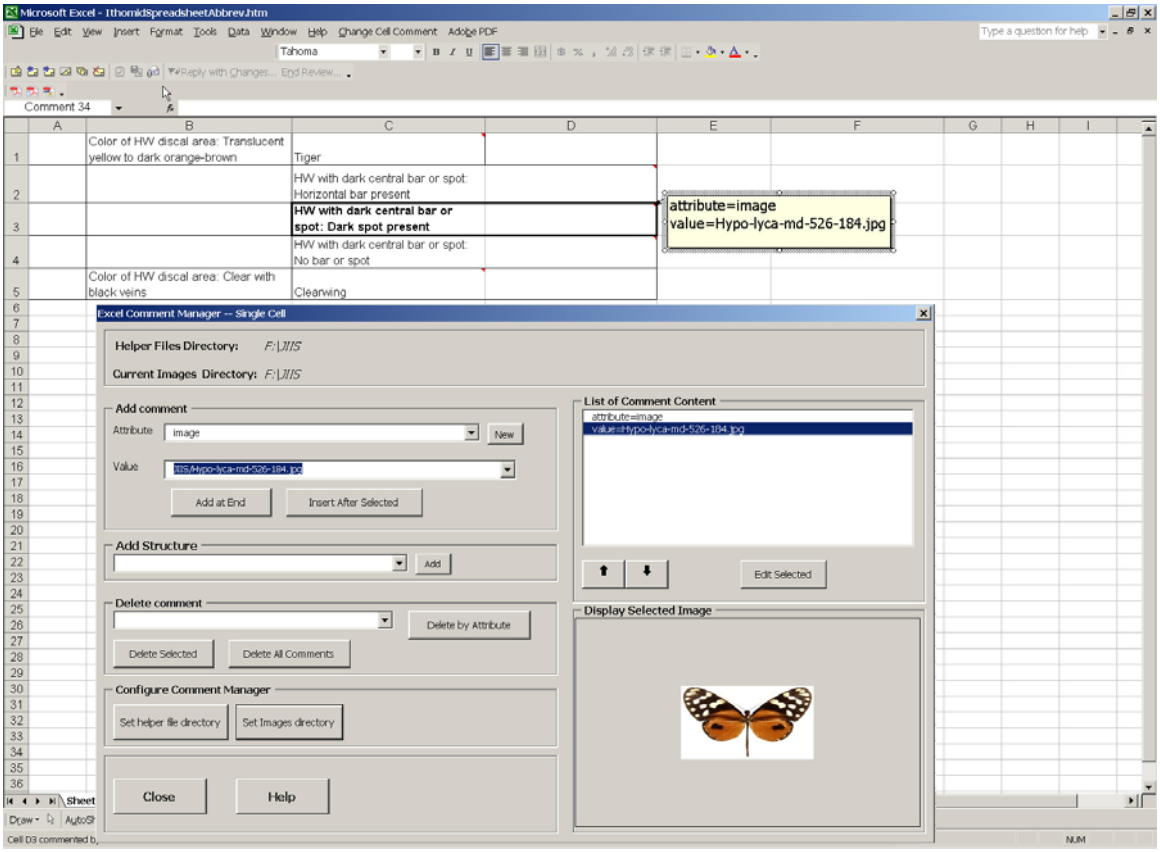

Figure 3. Portion of spreadsheet corresponding to Figures 1 and 2. Yellow box is comment opened for display only. Normally, authors invoke the constraint-based CommentManager shown at the bottom. This prevents syntax errors, remembers attributes and values, and offers previews of images used to illustrate attribute values. The resulting spreadsheet is saved in Microsoft native XML format and automatically converted to XML obeying our own schema upon upload to the server deploying the key. Applications render that XML in several different forms, including the tree browser illustrated in Figure 1.

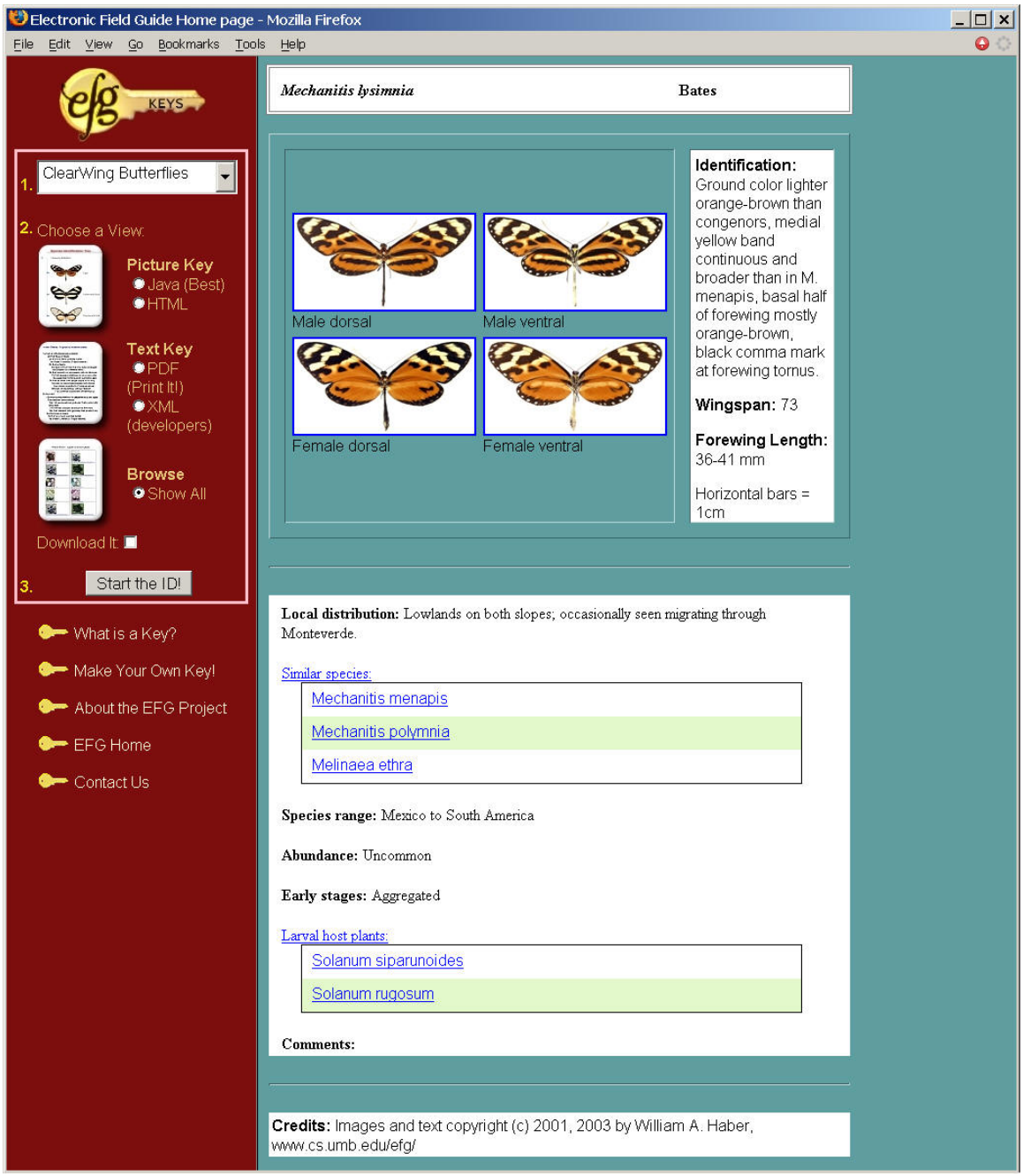

Figure 4. Result of making a database call to a service offering descriptive data with argument "Scientific name=Mechanitis lysimnia". This particular service can offer HTML or, if requested by the client, XML for further processing. In our present user interfaces, the user would normally select a link carrying the species name upon reaching a leaf node of the id tree. The key deployment framework would then construct and execute the appropriate service call.

Reference List

Apache (2005, accessed 2005) Welcome to WSIF: Web Services Invocation Framework. http://ws.apache.org/wsif/

Apache Jakarta (2005, accessed 2005) Tomcat. http://jakarta.apache.org/tomcat/

Brickell C.D.et al. (2004) *International Code of Nomenclature for Cultivated Plants.* International Society for Horticultural Science (ISHS) , Leuven, Belgium.

Dallwitz, M. J. (2000, accessed 2005) A Comparison of Interactive Identification Programs. http://delta-intkey.com/www/comparison.htm

Dallwitz, M. J., Paine T.A., and Zurcher E.J. (1993, accessed 2005) User's guide to the DELTA System: a general system for processing taxonomic descriptions. 4th edition. http://biodiversity.uno.edu/delta/

Diva Project (2005, accessed 2005) Diva home page. http://embedded.eecs.berkeley.edu/diva/

EarLab (2005) EarLab: A digital warehouse of auditory models and data. http://earlab.bu.edu/

Ereshefsky M. (1991) *The Units of Evolution: Essays on the Nature of Species*. MIT Press.

Ereshefsky M. (2001) *The Poverty of the Linnaean Hierarchy A Philosophical Study of Biological Taxonomy* . Cambridge University Press, Cambridge.

Exolab (2005, accessed 2005) The Castor Project. http://castor.exolab.org/

Gansner and North. An open graph visualization system and its applications to software engineering. Software Practice and Experience 30(10), 1203-1233. 2000.

ICBN (2000) *International Code of Botanical Nomenclature (St Louis Code). Regnum Vegetabile 138.* Koeltz Scientific Books., Königstein.

ICZN (2005, accessed 2005) International Code of Zoological Nomenclature. http://www.iczn.org/code.htm

Maddison, W. P. and Maddison D.R. (2005, accessed 2005) Mesquite: a modular system for evolutionary analysis. http://mesquiteproject.org/mesquite/mesquite.html

Marenco, L., Tosches, N., Crasto, C., Shepherd, G., Miller, P.L., & Nadkarni, P.M. (2003) Achieving Evolvable Web-Database Bioscience Applications Using the EAV/CR Framework: Recent Advances. *J Am Med Inform Assoc.*

Microsoft MSDN (2005) .NET Framework. http://msdn.microsoft.com/netframework/

Pankhurst R.J. (1991) *Practical Taxonomic Computing*. Cambridge University Press, Cambridge.

Reaka-Kudla M.L.et al. (1997) *Biodiversity II : understanding and protecting our biological resources*. Joseph Henry Press, Washington, D.C.

Skonnard, A. (2005) Web Services and Datasets. http://msdn.microsoft.com/msdnmag/issues/03/04/XMLFiles/

Sneath P.H.A. (1990) *International Code of Nomenclature of Bacteria: Bacteriological Code, 1990 Revision*. ASM Press.

Stevenson, R.D., Haber, W., & Morris, R.A. (2003) Conservation Ecology: Electronic Field Guides and User Communities in the Ecoinformatics Revolution. *Conservation Ecology [online]* **7**(1): 3. [online] URL: http://www.consecol.org/vol7/iss1/art3/

TDWG (2005a, accessed 2005) CODATA/TDWG Working Group on Access to Biological Collection Data (ABCD). http://www.bgbm.org/TDWG/CODATA/Schema/default.htm

TDWG (2005b, accessed 2005b) IUBS Taxonomic Database Working Group. http://www.tdwg.org/

TDWG (2005c, accessed 2005c) Taxonomic Database Working Group, Subgroup on the Structure of Descriptive Data (SDD). http://www.tdwg.org/sddhome.html

W3C (1999, accessed 2005) XSL Transformations (XSLT) Version 1.0; W3C Recommendation 16 November 1999. http://www.w3.org/TR/xslt

W3C (2004a, accessed 2005a) RDF: Resource Description Framework. http://www.w3.org/RDF/

W3C (2004b, accessed 2005b) XML Schema. http://www.w3.org/XML/Schema

W3C (2005, accessed 2005) XSL-FO Introduction and Overview. http://www.w3.org/TR/xsl/slice1.html#section-N742-Formatting

Wheeler Q. & Meier R. (2000) *Species concepts and phylogenetic theory : a debate*. Columbia University Press, New York .

Wilson. The diversity of life. 92. Belknap Press of Harvard University Press.

Wilson E.O. & Peter F.M. (1988) *National Forum on BioDiversity: Biodiversity*. National Academy Press, Washington, D.C.

Winston J.E. (1999) *Describing species : practical taxonomic procedure for biologists*. Columbia University Press, New York .