

Bioinformatics Service Reconciliation By Heterogeneous Schema Transformation

Lucas Zamboulis^{1,2}, Nigel Martin¹, and Alexandra Poulouvassilis¹

¹ School of Computer Science and Information Systems, Birkbeck, Univ. of London

² Department of Biochemistry and Molecular Biology, University College London

Abstract. This report focuses on the problem of bioinformatics service reconciliation in a generic and scalable manner so as to enhance interoperability in a highly evolving field. Using XML as a common representation format, but also supporting existing flat-file representation formats, we propose an approach for the scalable semi-automatic reconciliation of services, possibly invoked from within a scientific workflows tool. Service reconciliation may use the AutoMed heterogeneous data integration system as an intermediary service, or may use AutoMed to produce services that mediate between services. We discuss the application of our approach for the reconciliation of services in an example bioinformatics workflow. The main contribution of this research is an architecture for the scalable reconciliation of bioinformatics services.

1 Introduction

In recent years, the bioinformatics field has seen an explosion in the number of services offered to the community. These platform-independent software components have consequently been used for the development of complex tasks through service composition within workflows, thereby promoting reusability of services. However, the large number of services available impedes service composition and so developing techniques for semantic service discovery that would significantly reduce the search space is of great importance [LAWG05].

After discovering services that are relevant to one's interests, the next step is to identify whether these services are functionally compatible. Bioinformatics services are being independently created by many parties worldwide, using different technologies and data types, hindering integration and reusability [Ste02]. In particular, after discovering two such services, the researcher needs to first identify whether the output of the first is compatible with the input of the second based on a number of factors, such as the technology employed by each service, the representation format and the data type used.

In practice, compatible services are rare. Within Taverna³, service technology reconciliation is addressed by using Freefluo [OAF⁺04], an extensible workflow enactment environment that bridges the gap between web services and other service types, such as web-based REST services (stateless services that support

³ <http://taverna.sourceforge.net/>

caching). However, the researcher still needs to reconcile the outputs and inputs of services in terms of content, data type and representation format, spending time and effort in developing functionality that, even though essential for the services to interoperate, is irrelevant to the experiment.

The primary cause of this problem is the existence of multiple different data types and representation formats used even for basic concepts, such as DNA sequences. These data types and representation formats, used for the same or overlapping concepts, have been developed over the years by collaborative work between researchers and/or industry and so even though standardisation efforts are important and encouraged by the community, non-standardised efforts are likely to persist and new ones are bound to appear in this constantly evolving field. For this reason, service composition solutions that take into consideration this factor are essential. Unfortunately, there is currently an abundance of tools each concentrating on a specific data type and representation format (or combinations of pairs of data types and representation formats, when translation is needed) that are highly specific in accomplishing a certain task, rather than being generic [LBW⁺04]. As a result, reusability of existing tools is low.

Another common practice in bioinformatics is the use of flat-file representation formats for the overwhelming majority of data types, while the adoption rate of XML is low. This practice does not allow the application of Semantic Web technologies and solutions to their full extent, such as the semantic annotation of fields within a bioinformatics data type. For example, even though it is possible to annotate a service as having FASTA output, it is not possible to annotate the different fields within the non-tagged FASTA data type as being the sequence identifier, the sequence description and the sequence itself. But, even if a data type *is* tagged, e.g. UniProt, annotation cannot be performed in a generic way, as it would require data type-specific annotation tools.

We also observe that, even though the use of semantic annotations is key to service discovery and composition, service providers are disinclined to supply comprehensive annotations for their services. Relying on a centralised approach for such a task is clearly not scalable, and so any proposed solution for the reconciliation of bioinformatics services must ensure that the amount of required annotations is kept to a minimum and that it is reused as much as possible.

We argue that (a) the use of XML and (b) allowing the annotation and manipulation of service inputs and outputs at a fine-grained level, can boost service interoperability in a scalable manner. We therefore propose and exemplify an architecture for the reconciliation of services by exploiting the (manual) semantic annotation of service inputs and outputs using one or more interconnected ontologies, and the subsequent automatic restructuring of the XML output of one service to the required XML input of another. Although our approach uses XML as the common representation format, non-XML services are also supported by the use of converters to and from XML. Our schema and data transformation approach is supported by the AutoMed heterogeneous data integration system⁴ and can accommodate two types of service reconciliation: either using AutoMed

⁴ <http://www.doc.ic.ac.uk/automed>

as a service itself, e.g. from within a workflow tool, or using AutoMed to generate mediating services.

In the remainder of this report, Section 2 first reviews current approaches related to service interoperability. Section 3 then provides an overview of the AutoMed system, to the level of detail necessary for this report. Section 4 introduces our proposed approach for a scalable solution to the problem of bioinformatics service reconciliation. Section 5 presents our ongoing work in applying our approach to the reconciliation of bioinformatics services. Section 6 provides an overall discussion of our approach and gives our plans for future work.

2 Related Work

In the context of service composition, research such as [SK03, MBE03, BDSN02] has mainly focused on service technology reconciliation, service matchmaking and service routing, assuming that the outputs and inputs of services are a priori compatible. This assumption is restrictive, as it is often the case that two services are semantically compatible, but cannot interoperate due to incompatibilities in terms of data types and/or representation format.

In practice, this problem has forced service consumers to handle such incompatibilities with custom code from within the calling services. In an effort to minimise this issue and promote service reusability, the *myGrid* project⁵ has fostered the notion of *shims* [HSL⁺04], i.e. services that act as intermediaries between services and reconcile their output and input data. However, the problem here is that a new shim needs to be manually created for each pair of services that need to interoperate. [HSL05] states that, even though in theory the number of shim services that *myGrid* needs to provide is quadratic in the number of services it contains, the actual number of shims should be much smaller. However, in 2005 *myGrid* gave access to over 1,000 services (see [LAWG05]), and this number is now over 3,000, and so this manual approach is clearly not scalable.

[BL04] describes a scalable framework that uses mappings to one or more ontologies, possibly containing subtyping information, for reconciling the output of a service with the input of another. The sample implementation of this framework is able to use mappings to a single ontology in order to generate an XQuery query as the transformation program.

We observe that [BL04] only provides for shim generation, whereas our approach, by using the AutoMed data integration system, provides a uniform approach to workflow and data integration, both of which are key aspects of *in silico* biological experiments. Furthermore, the work presented here differs from [BL04] in a number of aspects and provides a more generic solution to the problem of bioinformatics service reconciliation. First, we also consider services that produce or consume non-XML data and also allow primitive data type reconciliation, whereas [BL04] assumes an XML setting and primitive data type compatibility. Moreover, we allow 1-*n* GLAV correspondences, compared to the

⁵ <http://www.mygrid.org.uk>

1-1 LAV correspondences of [BL04] and we also define a methodology for reconciling services that correspond to more than one ontology. We also note that our XML restructuring algorithm is able to avoid loss of information during data transformation, by analysing the hierarchical nature of the source and target schemas and by using subtype information provided by the ontologies.

[TAK05] also uses a mediator system for service composition. However, the focus is either to provide a service over the global schema of the mediator whose data sources are services, or to generate a new service that acts as an interface over other services. In contrast, we use the AutoMed toolkit to reconcile a sequence of semantically compatible services that need to form a pipeline: there is no need for a single ‘global schema’ or a single new service to be created.

Concerning the use of ontologies for data integration, a number of approaches have been proposed. For example, [ABFS02] uses an ontology as a virtual global schema for heterogeneous XML data sources using LAV mapping rules, while [CXH04] undertakes data integration using mappings between XML data sources and ontologies, transforming the source data into a common RDF format. In contrast, we use XML as the common representation format and focus on the restructuring of source data into a target XML format, rather than on integration.

3 Overview of AutoMed

AutoMed is a heterogeneous data transformation and integration system which offers the capability to handle virtual, materialised and hybrid data transformation/integration across multiple data models. It supports a low-level **hypergraph-based data meta-model (HDM)** and provides facilities for specifying higher-level modelling languages in terms of this HDM. An HDM schema consists of a set of nodes, edges and constraints, and each modelling construct of a higher-level modelling language is specified as some combination of HDM nodes, edges and constraints (the constraints are expressed in the IQL query language — see below).

For any modelling language \mathcal{M} specified in this way (via the API of AutoMed’s Model Definitions Repository) AutoMed provides a set of primitive schema transformations that can be applied to schema constructs expressed in \mathcal{M} . In particular, for every construct of \mathcal{M} there is an **add** and a **delete** primitive transformation which add to/delete from a schema an instance of that construct. For those constructs of \mathcal{M} which have textual names, there is also a **rename** primitive transformation.

Instances of modelling constructs within a particular schema are identified by means of their *scheme* enclosed within double chevrons $\langle\langle . . \rangle\rangle$. AutoMed schemas can be incrementally transformed by applying to them a sequence of primitive transformations, each adding, deleting or renaming just one schema construct (thus, in general, AutoMed schemas may contain constructs of more than one modelling language). A sequence of primitive transformations from one schema X_1 to another schema X_2 is termed a *pathway* from X_1 to X_2 and denoted by

$X_1 \rightarrow X_2$. All source, intermediate, and integrated schemas, and the pathways between them, are stored in AutoMed’s Schemas & Transformations Repository.

Each **add** and **delete** transformation is accompanied by a query specifying the extent of the added or deleted construct in terms of the rest of the constructs in the schema. This query is expressed in a functional query language, IQL [JPZ03]. Also available are **extend** and **contract** primitive transformations which behave in the same way as **add** and **delete** except that they state that the extent of the new/removed construct cannot be precisely derived from the rest of the constructs. Each **extend** and **contract** transformation takes a pair of queries that specify a lower and an upper bound on the extent of the construct. These bounds may be **Void** or **Any**, which respectively indicate no known information about the lower or upper bound of the extent of the new construct.

The queries supplied with primitive transformations can be used to translate queries or data along a transformation pathway $X_1 \rightarrow X_2$ (see [MP03a,MP03b] for details). For translating data from X_1 to data on X_2 the **add**, **extend** and **rename** steps are used. The queries supplied with primitive transformations also provide the necessary information for these transformations to be automatically *reversible*, in that each **add/extend** transformation is reversed by a **delete/contract** transformation with the same arguments (including the same query arguments), while each **rename** is reversed by a **rename** with the two arguments swapped. As discussed in [MP03a], this means that AutoMed is a **both-as-view (BAV)** data integration system: the **add/extend** steps in a transformation pathway correspond to Global-As-View (GAV) rules while the **delete** and **contract** steps correspond to Local-As-View (LAV) rules. If a GAV view is derived from solely **add** steps it will be *exact* in the terminology of [Len02]. If, in addition, it is derived from one or more **extend** steps using their lower-bound (upper-bound) queries, then the GAV view will be *sound (complete)* in the terminology of [Len02]. Similarly for LAV views. An in-depth comparison of BAV with the GAV and LAV approaches to data integration can be found in [MP03a], while [MP03b,MP06] discusses the use of BAV in a peer-to-peer data integration setting. [JTMP04] discusses how Global-Local-As-View (GLAV) rules [FLM99,MH03] can also be derived from BAV pathways. We note that AutoMed and BAV transform both schema and data together, and thus do not suffer from any data/schema divide.

4 Bioinformatics Service Reconciliation

In this section, we present the problems encountered during service reconciliation and describe our proposed approach for overcoming them, including a brief discussion of how our approach could be incorporated within a workflow tool. We then provide details of XML DataSource Schema (XMLDSS), the XML schema type used in our approach, and of our own earlier work on schema transformation using ontologies that has been extended to enable service reconciliation.

4.1 Proposed Approach

Consider a service S_1 that produces data that need to be consumed by another service S_2 . In general, the following issues need to be resolved when trying to handle data exchange between S_1 and S_2 :

1. **Data model heterogeneity:** different data models (e.g. legacy flat files and XML) or different schema types (e.g. DTD and XML Schema) may be used. It may also be the case that a service producing or consuming XML data does not have an accompanying XML schema.
2. **Semantic heterogeneity** refers to schematic differences caused by the use of different terminology, or describing the same information at different levels of granularity.
3. **Schematic heterogeneity** refers to schematic differences caused by modelling the same information in different ways. This heterogeneity is common to all data modelling languages, but is amplified in XML due to its hierarchical nature, as well as the possibility of using elements with a single text node and attributes interchangeably.
4. **Primitive data type heterogeneity** refers to differences caused when different schemas use different primitive data types for the same concept.

To resolve these issues, we propose the following 4-step approach, illustrated in Figure 1:

Step 1: XML as the common representation format. We handle differences in the representation format by using XML as the common representation format. If the output/input of a service is not in XML, then a format converter is needed to convert to/from XML.

Step 2: XMLDSS as the schema type. We use our own XMLDSS schema type for the XML documents input to and output by services. An XMLDSS schema can be automatically extracted from an XML document or automatically derived from an accompanying DTD/XML Schema, if one is available.

Step 3: Correspondences to typed ontologies. We use one or more ontologies as a ‘semantic bridge’ between services. Providers or users of services semantically annotate the inputs and outputs of services by defining correspondences between an XMLDSS schema and an ontology. Ontologies in our approach are typed, i.e. each concept is associated with a data type, and so defining correspondences resolves issues 2 and 4 discussed above.

Step 4: Schema and data transformation. We use the AutoMed toolkit to automatically transform the XMLDSS schema of the output of service S_1 to the XMLDSS schema of the input of service S_2 . This is achieved using the two automatic algorithms discussed in Section 4.4.

If service S_1 does not have an accompanying DTD or XML Schema for its output, sample XML output documents for this service must be provided, and these must represent all valid formats that S_1 is able to produce, so as to create an XMLDSS schema that represents all possible instances of the output of S_1 .

If it is not possible to provide sufficient sample documents, then an XMLDSS can be extracted at run-time for every new instance XML document output by S_1 . The same applies for the input of S_2 .

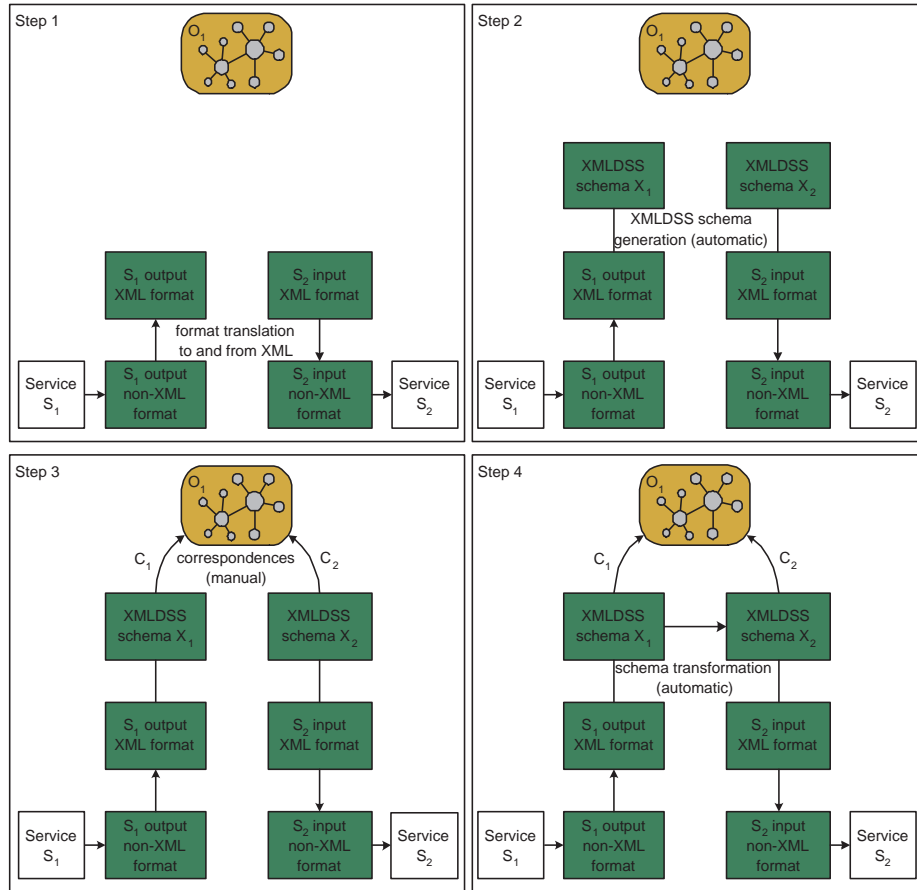


Fig. 1. Reconciliation of services S_1 and S_2 .

4.2 Integration of Approach With Workflow Tools

Our architecture for service reconciliation supports two different approaches identified below, depending on the preferred form of interoperability between AutoMed and the workflow tool.

Mediation service. With this approach, the workflow tool invokes service S_1 , receives its output, and submits this output and a handle on service S_2

to a service provided by the AutoMed system. This uses our approach to transform the output of S_1 to a suitable input for consumption by S_2 .

Shim generation. With this approach, the AutoMed system is used to generate shims, i.e. tools or services for the reconciliation of other services, by generating transformation scripts which are then incorporated within the workflow tool.

In the following, we provide an overview of the shim generation architecture. The mediation service architecture is described in more detail in Section 5.

With the shim generation approach, AutoMed is not part of the architecture, and so it is necessary to export AutoMed’s mediation functionality described and exemplified in Section 5. This functionality consists of the format converters, the algorithms for generating an XMLDSS schema from an XML document, DTD or XML Schema, and the XMLDSS schema transformation algorithms.

Format converters are not a part of the AutoMed toolkit and so can be used from within a workflow tool, without exporting any AutoMed functionality. The converters can be either incorporated within the workflow tool, or their functionality can be imported using services. As an example, a number of shims in *myGrid* are format converters.

The XMLDSS schema type is currently used only within the AutoMed system, but it does not require AutoMed functionality. As a result, the XMLDSS schema generation algorithms can be used from within a workflow tool in the same way as format converters.

The two XMLDSS schema transformation algorithms described in Section 4.4 are currently tightly coupled with the AutoMed system, since the algorithms use the Both-As-View data integration approach, which is currently supported only by the AutoMed system. In order to use our approach without integrating AutoMed with a workflow tool, we need to export this functionality. To this effect, we have designed an algorithm that derives a single XQuery query Q , able to materialise an XMLDSS schema X_2 using data from the data source of an XMLDSS schema X_1 , using the transformation pathway $X_1 \rightarrow X_2$ produced by our two algorithms. The algorithm for deriving query Q first uses AutoMed’s Query Processor to create the IQL view definition V of each construct c of X_2 in terms of constructs of X_1 , and to translate each V into an equivalent XQuery query, V_{XQuery} . The algorithm then creates a single XQuery query, Q , for materialising X_2 by following a bottom-up approach (not a top-down approach, as this would require the maintenance of pointers within the textual representation of Q). The algorithm first creates the XQuery queries for materialising the leaf elements of X_2 , together with their attributes and child text nodes. These queries are then used to create the queries that materialise the parent elements of the leaf elements, together with their attributes. This process is repeated until the root of X_2 is reached and the overall query Q is formulated. We list the algorithm in Appendix A.

4.3 XML DataSource Schema (XMLDSS)

The standard schema definition languages for XML are DTD and XML Schema. However, both of these provide grammars to which conforming documents adhere, and they do not explicitly summarise the tree structure of the data sources. In our schema transformation setting, tree-structured schemas are preferable as they facilitate schema traversal, structural comparison between a source and a target schema, and restructuring of the source schema. Moreover, such a schema type means that the queries supplied with AutoMed primitive transformations are essentially path queries, which are easily generated.

The AutoMed toolkit therefore supports a modelling language called *XML DataSource Schema* (XMLDSS), which summarises the tree structure of XML documents, much like DataGuides [GW97]. XMLDSS schemas consist of four kinds of constructs: **Element**, **Attribute**, **Text** and **NestList** (see [Zam04] for details of their specification in terms of the HDM). The last of these defines parent-child relationships either between two elements e_p and e_c or between an element e_p and the **Text** node. These are respectively identified by schemes of the form $\langle\langle i, e_p, e_c \rangle\rangle$ and $\langle\langle i, e_p, \text{Text} \rangle\rangle$, where i is the position of e_c or **Text** within the list of children of e_p in the XMLDSS schema.

In an XMLDSS schema there may be elements with the same name occurring at different positions in the tree. To avoid ambiguity, the identifier **elementName\$count** is used for each element, where **count** is incremented every time the same **elementName** is encountered in a depth-first traversal of the schema.

4.4 XML Schema and Data Transformation using Ontologies

This section describes the two algorithms used in our approach to transform XMLDSS schemas — in the setting of service reconciliation, these are the XMLDSS schemas of the outputs and inputs of services. Our own previous work in [Zam04,ZP04,ZP06] addressed the issue of XML schema and data transformation. This section describes an extended version of the approach presented in [ZP06], in that the expressiveness of the correspondences used in our approach has been enriched, and the algorithms used for XML schema and data transformation have been extended to support this.

Using XMLDSS as the schema type for XML data sources, we have developed two algorithms that are able to transform a source XMLDSS schema X_1 and its data to the structure of a target XMLDSS schema X_2 . The first algorithm, the schema conformance algorithm, uses manually defined correspondences between XMLDSS schemas X_1 and X_2 and an ontology O , in order to automatically transform X_1 and X_2 into equivalent schemas X'_1 and X'_2 that use the same terms as O . As a result, transformation pathways $X_1 \rightarrow X'_1$ and $X_2 \rightarrow X'_2$ are created. By the bidirectionality of BAV, a pathway $X'_2 \rightarrow X_2$ can be automatically derived from the pathway $X_2 \rightarrow X'_2$.

In [ZP06], a *correspondence* defines an **Element**, **Attribute** or **NestList** of an XMLDSS schema by means of an IQL query over a typed ontology.⁶ In particular, an **Element** e may map either to a **Class** c ; or to a path ending with a class-valued property of the form $\langle\langle p, c_1, c_2 \rangle\rangle$, where p is the property name and c_1 and c_2 are source and target classes; or to a path ending with a literal-valued property of the form $\langle\langle p, c, \text{Literal} \rangle\rangle$, where p is the property name and c the source class; additionally, the correspondence may state that the instances of a class are constrained by membership in some subclass. An XMLDSS **Attribute** may map either to a literal-valued property or to a path ending with a literal-valued property.

We now extend the correspondences of [ZP06] as follows. An XMLDSS scheme of the form $\langle\langle i, e, \text{Text} \rangle\rangle$ (where i denotes the order of $\langle\langle \text{Text} \rangle\rangle$ in the list of children of **Element** $\langle\langle e \rangle\rangle$) may map to a literal-valued property of the form $\langle\langle p, c, \text{Literal} \rangle\rangle$. In addition to 1-1 correspondences, we now also allow 1- n correspondences as follows. An **Element/Attribute** may map to more than one path over the ontology. In this case, n correspondences are required, each associating the same XMLDSS **Element/Attribute** to a different path over the ontology, and specifying an expression that determines the part of the extent of the **Element/Attribute** to which the correspondence applies (an example of this is given in Section 5). This expression is in general a select-project IQL query.

We note that these extended correspondences are GLAV, in contrast with the LAV correspondences defined in our own earlier work [ZP06], as an expression over an XMLDSS construct (rather than just an XMLDSS construct) maps to a path in the ontology.⁷

Our schema conformance algorithm (SCA) uses correspondences from an **Element** or **Attribute** to a single path over the ontology to rename that construct, ensuring consistency with the terminology of the ontology. In the case of a 1- n correspondence relating to an **Element** e with parent p , the algorithm first retrieves all relevant correspondences, then inserts n **Elements** under p (in the position previously held by e), named after the paths specified by the correspondences, and finally deletes e and its underlying structure. When inserting the n **Elements** under p , the algorithm also replicates the underlying structure of the old **Element** e under each one of the newly inserted **Elements**. A 1- n correspondence relating to an **Attribute** is handled similarly: the owner **Element** is replaced by n **Elements** with the same name, each containing a different **Attribute** named after the paths specified by the correspondences. A correspondence mapping an **Attribute** or a scheme of the form $\langle\langle i, e, \text{Text} \rangle\rangle$ in the XMLDSS to a literal-valued property in the ontology is used to perform primitive data type reconciliation: if the data type of the **Attribute** or scheme in the XMLDSS schema is not the same

⁶ In principle, it would be possible to use more high-level query languages such as XQuery to specify correspondences in our setting. Currently, AutoMed provides an XQuery-to-IQL translator component, capable of translating (possibly nested) FLWR XQuery queries to (possibly nested) select-project-join IQL queries.

⁷ Even though BAV pathways could have been used to express these GLAV mappings, we specify the mappings directly as GLAV rules for compactness.

as in the ontology, the algorithm replaces the `Attribute` or scheme by performing a type-casting operation.

After the transformation of schemas X_1 and X_2 into schemas X'_1 and X'_2 that use the same terms as O , a second algorithm, the schema restructuring algorithm (SRA) presented in [ZP06], automatically transforms X'_1 to the structure of X'_2 , producing a transformation pathway $X'_1 \rightarrow X'_2$. To do so, the SRA traverses X'_1 and X'_2 and first inserts into X'_1 those constructs present in X'_2 but not in X'_1 . After this *growing phase*, a *shrinking phase* follows, in which the SRA removes from X'_1 those constructs present in X'_1 but not in X'_2 . The SRA is able to generate synthetic structure to avoid loss of data caused by structural incompatibilities between X'_1 and X'_2 . The SRA is also able to use information that identifies an element/attribute in X'_1 to be either equivalent to, or a superclass of, or a subclass of an element/attribute in X'_2 . This information may be produced by, e.g. a schema matching tool or, in our context here, via correspondences to an ontology.

Consequently, an overall transformation pathway from X_1 to X_2 can now be obtained by composing the pathways $X_1 \rightarrow X'_1$, $X'_1 \rightarrow X'_2$ and $X'_2 \rightarrow X_2$. This pathway can be used to automatically transform data that is structured according to X_1 to be structured according to X_2 , and an XML document structured according to X_2 can finally be materialised (the pathway $X_1 \rightarrow X_2$ could also be used to translate queries expressed on X_2 to operate on X_1).

Note that we do not assume the existence of a single ontology. As discussed in [ZP06], it is possible for XMLDSS schema X_1 to have a set of correspondences C_1 to an ontology O_1 , and for XMLDSS schema X_2 to have a set of correspondences C_2 to another ontology O_2 . Provided there is an AutoMed transformation pathway between O_1 and O_2 , either directly or through one or more intermediate ontologies, our approach can use C_1 and the transformation pathway between O_1 and O_2 to automatically produce a new set of correspondences C'_1 between X_1 and O_2 . As a result, this setting is now identical to a setting with a single ontology. There is a proviso here that the new set of correspondences C'_1 must conform syntactically to the correspondences accepted as input by the schema conformance process. Determining necessary conditions for this to hold is an area of future work.

5 Case Study

We now describe our approach to service reconciliation in more detail by specifying a sample bioinformatics workflow, which is used to demonstrate the use of AutoMed as a mediation service.

Figure 2 illustrates a sample workflow that will be used to demonstrate our approach. This workflow contains three services. The first takes as input an IPI⁸ accession number, e.g. IPI00015171, and outputs the corresponding IPI entry as a flat file using the UniProt⁹ format (see Appendix B.1, page 21). The

⁸ International Protein Index, see <http://www.ebi.ac.uk/IPI>.

⁹ Universal Protein Resource, see <http://www.ebi.uniprot.org>.

second receives an InterPro¹⁰ accession number and returns the corresponding InterPro entry (see Appendix B.5, page 47). The third service receives a Pfam¹¹ accession number and returns the corresponding Pfam entry. In this workflow, two transformations are needed, T_1 and T_2 ; T_1 extracts the InterPro accession from an IPI entry using the UniProt format, while T_2 extracts the Pfam accession from an InterPro entry.



Fig. 2. Sample Workflow.

We now apply the mediation service approach described in Section 4.2, for the reconciliation of the services of the workflow of Figure 2.

Step 1: XML as a common representation format. Service *getIPIEntry* outputs a flat file that follows the UniProt representation format¹² and contains a single entry consisting of multiple lines. Each line consists of two parts, the first being a two-character line code, indicating the type of data contained in the line, while the second contains the actual data, consisting of multiple fields.

Since UniProt also has an XML representation format specified by an XML Schema¹³, we created a format converter that, given an IPI flat file f that follows the UniProt format, converts f to an XML file conforming to that XML Schema (see Appendices B.2 and B.4, pages 22 and 29, respectively).

Service *getInterProEntry* outputs an XML file (see Appendix B.5, page 47) and so there is no need for a format converter. Concerning the input of the second and the third service, they each take as input a single string, representing an InterPro/Pfam accession number, respectively. The input XML documents for these contain a single XML element, `ip_acc` and `pf_acc`, respectively, with a PCData node as a single child, as shown below. For these, the required format converters simply implement the functionality of the XPath expressions `/ip_acc/text()` and `/pf_acc/text()`, respectively.

```

<ip_acc>InterPro_accession_string</ip_acc>
  <pf_acc>Pfam_accession_string</pf_acc>

```

Step 2: XMLDSS schema generation. After resolving representation format issues, we now give details on the generation of XMLDSS schemas for our setting. As discussed above, service *getIPIEntry* outputs a flat file which is con-

¹⁰ See <http://www.ebi.ac.uk/interpro>.

¹¹ See <http://www.sanger.ac.uk/Software/Pfam>.

¹² IPI also supports the FASTA representation format, containing less information.

¹³ Available at <http://www.pir.uniprot.org/support/docs/uniprot.xsd>

verted to an XML file that conforms to the UniProt XML Schema. An XMLDSS schema for the output of this service is automatically derived from that XML Schema (see Appendices B.3 and B.4, pages 24 and 29, respectively). Similarly, an XMLDSS schema for the output of service *getInterProEntry* is automatically derived using the InterPro DTD schema¹⁴ (see Appendices B.6, page 50).

Concerning the input of the second and the third service, the corresponding XMLDSS schemas are automatically extracted by using a single sample XML document for each, such as the ones given earlier.

Step 3: Correspondences. After generating the required XMLDSS schemas for our example workflow, we need to specify the correspondences between these schemas and an ontology. In this case, we have used the typed *my* Grid OWL domain ontology¹⁵.

In general, all XMLDSS elements and attributes should be mapped to the ontology. However, it may be the case that a certain element or attribute does not have a corresponding term in the ontology. In this case, such constructs are not affected by our subsequent algorithms that use the correspondences to transform X_1 to the structure of X_2 . An advantage of this is that data transformation is still possible with only a partial set of correspondences from an XMLDSS schema to the ontology. This property is particularly significant in terms of the applicability and scalability of our approach, as it allows for incrementally defining the full set of correspondences between an XMLDSS schema and an ontology: one can define only those correspondences relevant to the specific problem at hand, instead of the full set of correspondences.

In our example, this means that we only need to specify correspondences for those constructs of the XMLDSS schema of the output of *getIPIEntry* that contribute to the input of service *getInterProEntry*. Consequently, we need to specify correspondences concerning only two constructs, $\langle\langle\text{dbReference}\$9\rangle\rangle$ and $\langle\langle\text{dbReference}\$9, \text{id}\rangle\rangle$ ¹⁶. These are shown in the table below. The first models an entry in a bioinformatics data resource, whose type is specified by $\langle\langle\text{dbReference}\$9, \text{type}\rangle\rangle$. The type of a resource is modelled in IPI using data values, whereas in the ontology it is modelled as classes, and so n correspondences are required for this construct, where n is the number of different types of resources that IPI supports and that also exist in the ontology. Each of these correspondences maps $\langle\langle\text{dbReference}\$9\rangle\rangle$ to a class in the ontology representing a bioinformatics data resource record and specifies the part of the extent of $\langle\langle\text{dbReference}\$9\rangle\rangle$ to which the correspondence applies. For example, the first correspondence states that those instances of $\langle\langle\text{dbReference}\$9\rangle\rangle$ whose $\langle\langle\text{dbReference}\$9, \text{type}\rangle\rangle$ Attribute has a data value of 'InterPro', map to the $\langle\langle\text{InterPro_record}\rangle\rangle$ ontology class. Due

¹⁴ Available at <ftp://ftp.ebi.ac.uk/pub/databases/interpro/interpro.dtd>

¹⁵ Available from <http://www.mygrid.org.uk/ontology>.

¹⁶ In the XMLDSS schema of Appendix B.3, page 24, Element $\langle\langle\text{dbReference}\$9\rangle\rangle$ is found using the XPath expression $(/uniprot/entry)[10]$, i.e. it is the right sibling of Element comment.

to space limitations, but without loss of generality, we only provide two such correspondences, the ones related to InterPro and Pfam, respectively.

Construct:	⟨⟨dbReference\$9⟩⟩
Extent:	[$d \{d, t\} \leftarrow \langle\langle\text{dbReference}\$9, \text{type}\rangle\rangle; t = ' \text{InterPro}'$]
Path:	⟨⟨InterPro_record⟩⟩
Construct:	⟨⟨dbReference\$9⟩⟩
Extent:	[$d \{d, t\} \leftarrow \langle\langle\text{dbReference}\$9, \text{type}\rangle\rangle; t = ' \text{Pfam}'$]
Path:	⟨⟨Pfam_record⟩⟩
Construct:	⟨⟨dbReference\$9, id⟩⟩
Extent:	[$\{d, i\} \{d, i\} \leftarrow \langle\langle\text{dbReference}\$9, \text{id}\rangle\rangle;$ $\{d, t\} \leftarrow \langle\langle\text{dbReference}\$9, \text{type}\rangle\rangle; t = ' \text{InterPro}'$]
Path:	[$\{ir, l\} \{ia, ir\} \leftarrow \langle\langle\text{part_of}, \text{InterPro_accession}, \text{InterPro_record}\rangle\rangle;$ $\{ia, l\} \leftarrow \langle\langle\text{datatype}, \text{InterPro_accession}, \text{Literal}\rangle\rangle$]
Construct:	⟨⟨dbReference\$9, id⟩⟩
Extent:	[$\{d, i\} \{d, i\} \leftarrow \langle\langle\text{dbReference}\$9, \text{id}\rangle\rangle;$ $\{d, t\} \leftarrow \langle\langle\text{dbReference}\$9, \text{type}\rangle\rangle; t = ' \text{Pfam}'$]
Path:	[$\{pr, l\} \{pa, pr\} \leftarrow \langle\langle\text{part_of}, \text{Pfam_accession}, \text{Pfam_record}\rangle\rangle;$ $\{pa, l\} \leftarrow \langle\langle\text{datatype}, \text{Pfam_accession}, \text{Literal}\rangle\rangle$]

The XMLDSS schema of the input of service *getInterProEntry* consists of a single Element construct, ⟨⟨ip_acc⟩⟩, which corresponds in the ontology to class ⟨⟨InterPro_accession⟩⟩. It also contains a NestList construct, ⟨⟨1, ip_acc, Text⟩⟩, and therefore the correspondences are defined as in the table below. The correspondences for the XMLDSS schema of the input of the third service, *getPfamEntry*, are similar and are therefore not listed.

Construct:	⟨⟨ip_acc\$1⟩⟩
Extent:	⟨⟨ip_acc\$1⟩⟩
Path:	[$ia \{ia, ir\} \leftarrow \langle\langle\text{part_of}, \text{InterPro_accession}, \text{InterPro_record}\rangle\rangle$]
Construct:	⟨⟨1, ip_acc\$1, Text⟩⟩
Extent:	⟨⟨1, ip_acc\$1, Text⟩⟩
Path:	[$\{ia, l\} \{ia, ir\} \leftarrow \langle\langle\text{part_of}, \text{InterPro_accession}, \text{InterPro_record}\rangle\rangle;$ $\{ia, l\} \leftarrow \langle\langle\text{datatype}, \text{InterPro_accession}, \text{Literal}\rangle\rangle$]

Step 4: Schema transformation. After manually specifying correspondences, the SCA and SRA algorithms can automatically transform the outputs of services *getIPIEntry* and *getInterProEntry* to the required inputs for services *getInterProEntry* and *getPfamEntry* respectively.

Concerning the output of service *getIPIEntry*, the SCA first retrieves all correspondences related to ⟨⟨dbReference\$9⟩⟩ (in this case 2 correspondences) and inserts ⟨⟨InterPro_record\$1⟩⟩ and ⟨⟨Pfam_record\$1⟩⟩, using the correspondences' expressions to select the appropriate ⟨⟨dbReference\$9⟩⟩ instances, i.e. those that have a type Attribute with value 'InterPro' and 'Pfam' respectively. As dis-

cussed in Section 4.4, the SCA then replicates under the newly inserted Elements the structure located under $\langle\langle\text{dbReference}\$9\rangle\rangle$ (again using the correspondences' expressions to select the appropriate structure), and then removes $\langle\langle\text{dbReference}\$9\rangle\rangle$. Note that this removal is postponed until after any other insertions are performed. This is because other insertions may need to use the extent of $\langle\langle\text{dbReference}\$9\rangle\rangle$ in the queries supplied with the AutoMed transformations.

The SCA then retrieves all correspondences related to $\langle\langle\text{dbReference}\$9, \text{id}\rangle\rangle$ (in this case 2 correspondences) and inserts Attributes $\langle\langle\text{InterPro_record}\$1, \text{InterPro_record.part.of.InterPro_accession}\rangle\rangle$ and $\langle\langle\text{Pfam_record}\$1, \text{InterPro_record.part.of.Pfam_accession}\rangle\rangle$, using the correspondences' expressions to select the appropriate $\langle\langle\text{dbReference}\$9, \text{id}\rangle\rangle$ instances (as discussed earlier, $\langle\langle\text{dbReference}\$9\rangle\rangle$ has not yet been removed). Concerning primitive data types, $\langle\langle\text{dbReference}\$9, \text{id}\rangle\rangle$ is of type `string`, and the same applies for all accession numbers in the *my*Grid domain ontology, so there is no need for any type-casting operations.

Concerning the input of service *getInterProEntry*, the SCA uses the first correspondence to rename $\langle\langle\text{ip_acc}\$1\rangle\rangle$ to $\langle\langle\text{InterPro_record.part.of.InterPro_accession}\$1\rangle\rangle$, while the second correspondence, which is a primitive data type reconciliation correspondence, is of no consequence as both the input of the service and the ontology model InterPro accessions using type `string`.

After the application of the SCA, the XMLDSS schema X_2 of the input of service *getInterProEntry* contains three constructs, $\langle\langle\text{InterPro_record.part.of.InterPro_accession}\$1\rangle\rangle$, $\langle\langle\text{Text}\rangle\rangle$ and a NestList linking these two constructs. The XMLDSS schema of the output of service *getIPIEntry*, X_1 , contains a number of constructs, but the only ones relevant to those of X_2 are $\langle\langle\text{InterPro_record}\$1\rangle\rangle$ and $\langle\langle\text{InterPro_record}\$1, \text{InterPro_record.part.of.InterPro_accession}\rangle\rangle$. The schema restructuring algorithm (SRA) therefore applies a number of contract transformations supplied with the queries Void and Any, so as to remove non-relevant constructs. The only non-trivial transformation is the attribute-to-element transformation: first Element $\langle\langle\text{InterPro_record.part.of.InterPro_accession}\$1\rangle\rangle$ is added to X_1 using the extent of Attribute $\langle\langle\text{InterPro_record}\$1, \text{InterPro_record.part.of.InterPro_accession}\rangle\rangle$, then NestList $\langle\langle\text{InterPro_record.part.of.InterPro_accession}\$1, \text{Text}\rangle\rangle$ is added, again using the Attribute extent, and finally the Attribute is deleted.

After the application of the SRA, we finally employ the XMLDSS schema materialisation algorithm defined in [ZP04] to materialise X_2 , i.e. the input of service *getInterProEntry*, using data from the data source of X_1 , i.e. the output of service *getIPIEntry*, using the transformation pathway $X_1 \rightarrow X'_1 \rightarrow X'_2 \rightarrow X_2$.

The application of Step 4 for the second part of our workflow is similar.

6 Conclusions and Future Work

In this report we have presented a generic and scalable architecture for bioinformatics service reconciliation within a wider data transformation framework. Our approach makes no assumptions about representation format, primitive data

type usage or the number of ontologies used. Moreover, this approach can be used either dynamically or statically from within a workflow tool.

The architecture exploits format converters to establish a common XML format for all service inputs and outputs, thus reducing the overall complexity of service reconciliation by establishing a common representation format. Service inputs and outputs are then abstracted using the XMLDSS schema type which can be automatically generated either from XML documents, or from accompanying DTD or XML Schema specifications using our algorithms.

A significant contribution of our approach is the ability to use correspondences to multiple ontologies for defining the semantics of services. This ‘semantic bridge’ is then used by two automatic algorithms that use the correspondences to allow data transformation between services. The schema conformance algorithm is able to use 1-1 and 1- n GLAV correspondences to ontologies, in order to produce schemas with no semantic heterogeneity. The schema restructuring algorithm then restructures the source schema to the target schema. This algorithm is able to avoid loss of information that may be caused due to structural incompatibilities of the data sources.

While correspondences to ontologies must be manually or semi-automatically produced, an advantage of our approach is that correspondence reusability is promoted by allowing the use of multiple ontologies. Moreover, our approach does not require the full set of correspondences to be defined, but instead allows the definition of only those correspondences between the XMLDSS schema and the ontology that are relevant to the problem at hand - we therefore allow an incremental approach for the definition of correspondences.

The architecture has been illustrated with a real bioinformatics workflow, characteristic of those currently available with string-based inputs. Future workflows with more complex inputs are to be expected, which our architecture will also readily support.

Concerning the integration of our approach with workflow tools, we defined two possible architectures. The first, using AutoMed as a mediation service, can easily be used from within a workflow tool by invoking AutoMed as a service and does not require the tool to provide XMLDSS generation functionality, or to support XQuery. On the other hand, the shim generation architecture does not require the dynamic collaboration of AutoMed with the workflow tool, as AutoMed is used to statically generate shims, which can then be incorporated into any workflow tool that supports XQuery.

Our current implementation has supported testing of the transformation pathways underpinning the service reconciliation examples presented within the AutoMed toolkit. Ongoing work is aimed at the integration of our approach with the Taverna workflow tool. The resulting implementation will be evaluated within the proteomics grid infrastructure being developed in the ISPIDER project [ZFB⁺06].

In future work, we will investigate the necessary conditions under which a transformation of a set of correspondences produces a new set of correspondences that conforms to the required format of our conformance algorithm, as

discussed at the end of Section 4.4. Other extensions to our work include investigating the effect of constraints on XMLDSS schemas and/or the ontologies on our approach, and also considering the effect of the evolution of the inputs and outputs of services.

Acknowledgements. The work presented in this report was funded by a BB-SRC grant. The authors would also like to thank the members of the ISPIDER project and especially Khalid Belhajjame, Suzanne Embury and Norman Paton for the fruitful discussions that helped shape the work presented in this report.

References

- [ABFS02] B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-based integration of XML web resources. In *Proc. of Int. Semantic Web Conference*, pages 117–131, 2002.
- [BDSN02] B. Benatallah, M. Dumas, Q.Z. Sheng, and A.H.H. Ngu. Declarative composition and peer-to-peer provisioning of dynamic web services. In *Proc. of ICDE'02*, pages 297–308. IEEE Computer Society, 2002.
- [BL04] S. Bowers and B. Ludäscher. An ontology-driven framework for data transformation in scientific workflows. In *Proc. of Data Integration in the Life Sciences (DILS'04)*, pages 1–16, 2004.
- [CXH04] I. F. Cruz, H. Xiao, and F. Hsu. An ontology-based framework for XML semantic integration. In *Proc. IDEAS'04*, pages 217–226, 2004.
- [FLM99] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *National Conference on Artificial Intelligence*, pages 67–73. AAAI Press, 1999.
- [GW97] R. Goldman and J. Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *Proc. VLDB'97*, pages 436–445, 1997.
- [HSL⁺04] D. Hull, R. Stevens, P. Lord, C. Wroe, and C. Goble. Treating shimantic web syndrome with ontologies. In *Proc. of Advanced Knowledge Technologies workshop on Semantic Web Services*, 2004. ISSN:1613-0073.
- [HSL05] D. Hull, R. Stevens, and P. Lord. Describing web services for user-oriented retrieval. In *Proc. of W3C Workshop on Frameworks for Semantics in Web Services*, 2005. Available at <http://www.w3.org/2005/04/FSWS/accepted-papers.html>.
- [JPZ03] E. Jasper, A. Poulouvasilis, and L. Zamboulis. Processing IQL queries and migrating data in the AutoMed toolkit. AutoMed Technical Report 20, July 2003.
- [JTMP04] E. Jasper, N. Tong, P.J. McBrien, and A. Poulouvasilis. View generation and optimisation in the AutoMed data integration framework. In *Proc. of 6th Baltic Conference on Databases and Information Systems*, 2004.
- [LAWG05] P. Lord, P. Alper, C. Wroe, and C. Goble. Feta: A light-weight architecture for user oriented semantic service discovery. In *Proc. of European Semantic Web Conference (ESWC'05)*, pages 17–31, 2005.
- [LBW⁺04] P. Lord, S. Bechhofer, M. Wilkinson, G. Schiltz, D. Gessler, D. Hull, C. Goble, and L. Stein. Applying Semantic Web Services to bioinformatics: Experiences gained, lessons learnt. In *Proc. of Int. Semantic Web Conference*, pages 350–364, 2004.

- [Len02] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. PODS'02*, pages 233–246, 2002.
- [MBE03] B. Medjahed, A. Bouguettaya, and A.K. Elmagarmid. Composing web services on the Semantic Web. *VLDB Journal*, 12(4):333–351, 2003.
- [MH03] J. Madhavan and A.Y. Halevy. Composing mappings among data sources. In *Proc. of VLDB'03*, pages 572–583, 2003.
- [MP03a] P. McBrien and A. Poulouvasilis. Data integration by bi-directional schema transformation rules. In *Proc. ICDE'03*. ICDE, March 2003.
- [MP03b] P. McBrien and A. Poulouvasilis. Defining peer-to-peer data integration using both as view rules. In *Proc. Workshop on Databases, Information Systems and Peer-to-Peer Computing (at VLDB'03), Berlin*, 2003.
- [MP06] P.J. McBrien and A. Poulouvasilis. P2P query reformulation over Both-as-View data transformation rules. In *Proc. of Databases, Information Systems and Peer-to-Peer Computing (DBISP2P'06)*, page TBC. Springer, 2006.
- [OAF⁺04] T. Oinn, M. Addis, J. Ferris, et al. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.
- [SK03] B. Srivastava and J. Koehler. Web Service composition - current solutions and open problems. In *Proc. of Workshop on Planning for Web Services (ICAPS'03)*, pages 28–35, 2003.
- [Ste02] L. Stein. Creating a bioinformatics nation. *Nature*, 417:119–120, May 2002.
- [TAK05] S. Thakkar, J.L. Ambite, and C. A. Knoblock. Composing, optimizing, and executing plans for bioinformatics web services. *VLDB Journal*, 14(3):330–353, 2005.
- [Zam04] L. Zamboulis. XML data integration by graph restructuring. In *Proc. British National Conference on Databases (BNCOD'04)*, pages 57–71, 2004.
- [ZFB⁺06] L. Zamboulis, H. Fan, K. Belhajjame, J. Siepen, A. Jones, N.J. Martin, A. Poulouvasilis, S.J. Hubbard, S. Embury, and N.W. Paton. Data access and integration in the ISPIDER proteomics grid. In *Proc. Data Integration for the Life Sciences 2006 (DILS'06), LNCS 4075*, pages 3–18, 2006.
- [ZP04] L. Zamboulis and A. Poulouvasilis. Using AutoMed for XML Data Transformation and Integration. In *Proc. International Workshop on Data Integration over the Web (at CAiSE'04)*, pages 58–69, 2004.
- [ZP06] L. Zamboulis and A. Poulouvasilis. Information sharing for the Semantic Web - a schema transformation approach. In *Proc. International Workshop Data Integration and the Semantic Web (at CAiSE'06)*, pages 275–289, 2006.

A XQuery Query Generation Algorithm

Panel 1 provides the algorithm used to derive a single XQuery query Q , able to materialise a target XMLDSS schema X_2 using data from XMLDSS schema X_1 , provided there is an AutoMed transformation pathway $X_1 \rightarrow X_2$.

In line 4 of the algorithm, an IQL query is translated into XQuery. The IQL-to-XQuery translator supports a subset of the XQuery language, and in particular can translate (possibly nested) select-project-join-union IQL queries into (possibly nested) XQuery FLWR expressions. Note that, since the schema transformation algorithms use custom IQL functions to generate synthetic extent, the translator is responsible for creating the equivalent XQuery functions.

In lines 7 and 13, the algorithm creates XQuery queries able to materialise Elements of X_2 using data from the data source of X_1 . These queries actually create Element instances whose labels are unique identifiers. This is because, in order to preserve the correct parent-child relationships in the materialised instance of X_2 , the XQuery queries in line 13 must perform equijoins instead of simple joins, and therefore contain a `WHERE` clause that makes use of these identifiers. These are of the form `elementName$count_sid&instanceCount`, where `sid` is the schema unique identifier provided by the AutoMed Repository, and `instanceCount` is an instance counter.

Algorithm 1: XQuery Query Generation Algorithm

```
Input: XMLDSS source schema  $X_1$ , XMLDSS target schema  $X_2$ 
Output: XQuery query able to materialise  $X_2$  using data from  $X_1$ 
/* Create an XQuery view definition for each construct  $c$  of  $X_2$  */
1 Let Views be an array with two columns
2 for (each NestList construct  $c$  of  $X_2$ ) do
3   | Create the IQL view definition  $V$  of  $c$ 
4   | Translate  $V$  into an equivalent XQuery view definition,  $V_{XQuery}$ 
5   | Add  $(c, V_{XQuery})$  to Views
/* Create XQuery queries that materialise leaf elements  $e$  */
6 for (each leaf element  $e$  of  $X_2$ ) do
7   | Create XQuery query  $q_e$  materialising  $e$ , its attribute and child text nodes
8   | Replace the entry for  $e$  in Views with  $(e, q_e)$ 
/* Create XQuery query  $Q$  that materialises  $X_2$  */
9 Let  $d$  be the DOM version of  $X_2$ 
10 Let Elms be a list containing the parent elements of the leaf elements
11 while (Elms is not empty) do
12   | Remove the first item of Elms,  $p$ 
13   | Create XQuery query  $q_p$  that uses Views to materialise  $p$ , its attribute and
    | child text nodes.
14   | Replace the entry for  $p$  in Views with  $(p, q_p)$ 
15   | if (Elms does not contain the parent of  $p$ ) then
16   | | Add the parent element of  $p$  at the end of list Elms.
```

B Documents for *getIPIEntry* → *getInterProEntry* → *getPfamEntry*

This section provides some of the documents used for or produced in the process of the reconciliation of the services of workflow *getIPIEntry* → *getInterProEntry* → *getPfamEntry*. In particular, we provide the documents related to the outputs of the second and the third services, as those related to the inputs of the second and third services are simple and are provided in the main text of this report.

We first list the documents related to the output of service *getIPIEntry*. Appendix B.1, page 21, lists the output of service *getIPIEntry* given IPI accession IPI00015171, i.e. a UniProt-style flat file representation of an IPI entry. Appendix B.2, page 22, lists the XML version of Appendix B.1, produced by the IPI flat-file-to-XML format converter, as discussed in Section ???. Appendix B.3, page 24, lists the XMLDSS schema automatically derived from the UniProt XML Schema, listed in Appendix B.4, page 29.

We then list the documents related to the output of service *getInterProEntry*. Appendix B.5, page 47, lists the output of service *getIPIEntry* given InterPro accession IPR003959, i.e. an XML file corresponding to the DTD listed in Appendix B.6, page 50. This DTD is used to automatically derive the XMLDSS schema listed in Appendix B.7, page 53.

B.1 IPI Entry IPI00015171 (UniProt Flat-File Version)

```
ID IPI00015171.4 IPI; PRT; 128 AA.
AC IPI00015171; IPI00784015;
DT 01-OCT-2001 (IPI Human rel. 2.00, Created)
DT 04-SEP-2005 (IPI Human rel. 3.10, Last sequence update)
DE SIMILAR TO AFG3-LIKE PROTEIN 1.
OS Homo sapiens (Human).
OC Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
OC Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
OX NCBI_TaxID=9606;
CC -!- CHROMOSOME: 16.
CC -!- START CO-ORDINATE: 88566495.
CC -!- END CO-ORDINATE: 88590519.
CC -!- STRAND: 1.
DR UniProtKB/Swiss-Prot; O43931; AFG31_HUMAN; -.
DR Vega; OTTHUMP00000080121; OTTHUMG00000072815; M.
DR ENSEMBL; ENSP00000373622; ENSG00000167540; -.
DR H-InvDB; HIT000013102; HIX0013375; -.
DR UniParc; UPI000059D3F9; -; -.
DR HGNC; 314; AFG3L1; -.
DR Entrez Gene; 172; AFG3L1; -.
DR UniGene; Hs.534773; -; -.
DR trome; HTR002494; -; -.
DR UTRdb; BB153156; -; 5'UTR.
DR UTRdb; BB281690; -; 5'UTR.
DR UTRdb; BB394662; -; 5'UTR.
DR RZPD; Hs.534773; -; Clones and other research material.
DR CleanEx; HS_AFG3L1; -; -.
DR InterPro; IPR003959; AAA_ATPase_core.
DR Pfam; PF00004; AAA; 1.
SQ SEQUENCE 128 AA; 13733 MW; B7335211CD58D03B CRC64;
MRPGRFHRQI YTGPPYIKGR SSIFKVHLRP LKLDKSLNKD TLARKLAVLT PGFPGVHHTP
GQGAPLRTVP APGAAALHPG AALRPHVHDA RGPGRSRAAVL RVDHYGGSGR PEEGHPECLR
PGCAVWGE
//
```

B.2 IPI Entry IPI00015171 (XML Version)

```
<uniprot>
  <entry created="2001-10-01" modified="2005-09-04" version="3.10">
    <accession>IPI00015171</accession>
    <accession>IPI00784015</accession>
    <organism key="1">
      <name scientific="Homo sapiens"/>
      <name common="Human"/>
      <lineage>
        <taxon>Eukaryota</taxon>
        <taxon>Metazoa</taxon>
        <taxon>Chordata</taxon>
        <taxon>Craniata</taxon>
        <taxon>Vertebrata</taxon>
        <taxon>Euteleostomi</taxon>
        <taxon>Mammalia</taxon>
        <taxon>Eutheria</taxon>
        <taxon>Primates</taxon>
        <taxon>Catarrhini</taxon>
        <taxon>Hominidae</taxon>
        <taxon>Homo</taxon>
      </lineage>
      <dbReference id="9606" key="2" type="NCBI_TaxID">
        </dbReference>
    </organism>
    <!-- CHROMOSOME: 16. -->
    <!-- START CO-ORDINATE: 88566495. -->
    <!-- END CO-ORDINATE: 88590519. -->
    <!-- STRAND: 1. -->
    <dbReference id="043931" key="3" type="Swiss-Prot">
      <property type="entry id" value="AFG31_HUMAN">
        </property>
      <property type="master" value="0">
        </property>
    </dbReference>
    <dbReference id="OTTHUMP00000080121" key="4" type="Vega">
      <property type="gene id" value="OTTHUMG00000072815">
        </property>
      <property type="master" value="1">
        </property>
    </dbReference>
    <dbReference id="ENSP00000373622" key="5" type="ENSEMBL">
      <property type="gene id" value="ENSG00000167540"/>
      <property type="master" value="0"/>
    </dbReference>
    <dbReference id="HIT000013102" key="6" type="H-InvDB">
      <property type="H-Inv cluster id" value="HIX0013375"/>
      <property type="master" value="0"/>
    </dbReference>
  </entry>
</uniprot>
```

```

<dbReference id="UPI000059D3F9" key="7" type="UniParc">
</dbReference>
<dbReference id="HGNC:314" key="8" type="HGNC">
  <property type="HGNC official gene symbol" value="AFG3L1"/>
</dbReference>
<dbReference id="172" key="9" type="Entrez Gene">
  <property type="default gene symbol" value="AFG3L1"/></dbReference>
<dbReference id="Hs.534773" key="10" type="UniGene"/>
<dbReference id="HTR002494" key="11" type="trome"/>
<dbReference id="BB153156" key="12" type="UTRdb"/>
<dbReference id="BB281690" key="13" type="UTRdb"/>
<dbReference id="BB394662" key="14" type="UTRdb"/>
<dbReference id="Hs.534773" key="15" type="RZPD"/>
<dbReference id="HS_AFG3L1" key="16" type="CleanEx"/>
<dbReference id="IPR003959" key="17" type="InterPro">
  <property type="entry name" value="AAA_ATPase_core"/>
  <property type="master" value="0"/>
</dbReference>
<dbReference id="PF00004" key="18" type="Pfam">
  <property type="method name" value="AAA"/>
  <property type="number of hits" value="1"/>
</dbReference>
<sequence crc64="B7335211CD58D03B" length="128" mass="13733">
  MRPGRFHRQIYTGPPYIKGRSSIFKVHLRPLKLDKSLNKDTLARKLAVLT
  PGFPGVHHTPGQGAPLRTVPAPGAAALHPGAALRPHVHDARGPGSRAAVL
  RVDHYGGSGRPEEGHPECLRPGCAVWGE
</sequence>
</entry>
</uniprot>

```

B.3 UniProt XMLDSS

```
<uniprot>
  <entry created="text" dataset="text" modified="text" version="text">
    <accession>
      text
    </accession>
    <name>
      text
    </name>
    <protein evidence="text" type="text">
      <name evidence="text" ref="text">
        text
      </name>
      <domain>
        <name evidence="text" ref="text">
          text
        </name>
      </domain>
      <component>
        <name evidence="text" ref="text">
          text
        </name>
      </component>
    </protein>
    <gene>
      <name/>
    </gene>
    <organism key="text">
      <name/>
      <dbReference evidence="text" id="text" key="text" type="text">
        <property type="text" value="text"/>
      </dbReference>
      <lineage>
        <taxon>
          text
        </taxon>
      </lineage>
    </organism>
    <organismHost key="text">
      <name/>
      <dbReference evidence="text" id="text" key="text" type="text">
        <property type="text" value="text"/>
      </dbReference>
      <lineage>
        <taxon>
          text
        </taxon>
      </lineage>
    </organismHost>
  </entry>
</uniprot>
```

```

<geneLocation evidence="text" type="text">
  <name/>
</geneLocation>
<reference evidence="text" key="text">
  <citation city="text" country="text" date="text"
    db="text" first="text" institute="text"
    last="text" name="text" number="text"
    publisher="text" type="text" volume="text">
    <title>
      text
    </title>
    <editorList>
      <person name="text"/>
      <consortium name="text"/>
    </editorList>
    <authorList>
      <person name="text"/>
      <consortium name="text"/>
    </authorList>
    <locator>
      text
    </locator>
    <dbReference evidence="text" id="text" key="text" type="text">
      <property type="text" value="text"/>
    </dbReference>
    <citingCitation city="text" country="text" date="text"
      db="text" first="text" institute="text"
      last="text" name="text" number="text"
      publisher="text" type="text" volume="text">
      <title>
        text
      </title>
      <editorList>
        <person name="text"/>
        <consortium name="text"/>
      </editorList>
      <authorList>
        <person/>
        <consortium/>
      </authorList>
      <locator>
        text
      </locator>
      <dbReference/>
      <citingCitation city="text" country="text" date="text"
        db="text" first="text" institute="text"
        last="text" name="text" number="text"
        publisher="text" type="text" volume="text">
        <title>
          text
        </title>

```

```

</title>
<editorList>
  <person/>
  <consortium/>
</editorList>
<authorList>
  <person/>
  <consortium/>
</authorList>
<locator>
  text
</locator>
<dbReference/>
<citingCitation city="text" country="text" date="text"
  db="text" first="text" institute="text"
  last="text" name="text" number="text"
  publisher="text" type="text" volume="text">
  <title>
    text
  </title>
  <editorList/>
  <authorList/>
  <locator>
    text
  </locator>
  <dbReference/>
  <citingCitation/>
</citingCitation>
</citingCitation>
</citation>
</reference>
<comment error="text" evidence="text" locationType="text" mass="text"
  method="text" name="text" status="text" type="text">
  <text>
    text
  </text>
<absorption>
  <max>
    text
  </max>
  <text>
    text
  </text>
</absorption>
<kinetics>
  <KM>
    text
  </KM>
  <Vmax>

```

```

        text
    </Vmax>
</text>
    text
</text>
</kinetics>
<phDependence>
    text
</phDependence>
<redoxPotential>
    text
</redoxPotential>
<temperatureDependence>
    text
</temperatureDependence>
<link uri="text"/>
<location sequence="text">
    <begin position="text" status="text"/>
    <end position="text" status="text"/>
    <position position="text" status="text"/>
</location>
<event evidence="text" namedIsoforms="text" type="text">
    text
</event>
<comment>
    text
</comment>
<isoform>
    <id>
        text
    </id>
    <name/>
    <sequence ref="text" type="text"/>
    <note evidence="text">
        text
    </note>
</isoform>
<interactant intactId="text">
    <id>
        text
    </id>
    <label>
        text
    </label>
</interactant>
<organismsDiffer AutoMedPrimitiveXSDType="xs:boolean">
    text
</organismsDiffer>
<experiments AutoMedPrimitiveXSDType="xs:integer">
    text

```

```

        </experiments>
        <note>
            text
        </note>
    </comment>
    <dbReference/>
    <keyword evidence="text" id="text">
        text
    </keyword>
    <feature description="text" evidence="text" id="text" ref="text"
        status="text" type="text">
        <original>
            text
        </original>
        <variation>
            text
        </variation>
        <location sequence="text">
            <begin position="text" status="text"/>
            <end position="text" status="text"/>
            <position position="text" status="text"/>
        </location>
    </feature>
    <evidence attribute="text" category="text" date="text"
        key="text" type="text"/>
    <sequence checksum="text" length="text" mass="text"
        modified="text" version="text">
        text
    </sequence>
</entry>
<copyright>
    text
</copyright>
</uniprot>

```

B.4 UniProt XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--*****
UniProt Knowledgebase
Version:    $Revision: 1.27 $
Date:      $Date: 2006/05/24 13:37:45 $

Copyright (c) 2003 UniProt consortium
All rights reserved.
*****-->
<xs:schema targetNamespace="http://uniprot.org/uniprot"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://uniprot.org/uniprot"
elementFormDefault="qualified">
  <!-- XML Schema definition for the UniProt XML format
  Tested with:
  -XSV (XML Schema Validator), http://www.ltg.ed.ac.uk/~ht/xsv-status.html
  -SQC (XML Schema Quality Checker), http://www.alphaworks.ibm.com/tech/xmlsqc
  -MSV (Multi-Schema XML Validator),
      http://www.sun.com/software/xml/developers/multischema/
  -XMLSpy, http://www.xmlspy.com/
  -->
  <!-- Name definition begins -->
  <xs:complexType name="proteinNameType">
    <xs:annotation>
      <xs:documentation>The name type is used for protein names occurring in an
      entry, which are represented in a flat file as DE lines.</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="evidence" type="xs:string" use="optional"/>
        <xs:attribute name="ref" type="xs:string" use="optional">
          <xs:annotation>
            <xs:documentation>This is referring to a possible EC number
            (ENZYME database cross reference).</xs:documentation>
          </xs:annotation>
        </xs:attribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="geneNameType">
    <xs:annotation>
      <xs:documentation>The gene name type is used for gene information.
      </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="evidence" type="xs:string" use="optional"/>
        <xs:attribute name="type" use="required">
          <xs:simpleType>
```

```

        <xs:restriction base="xs:string">
            <xs:enumeration value="primary"/>
            <xs:enumeration value="synonym"/>
            <xs:enumeration value="ordered locus"/>
            <xs:enumeration value="ORF"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:complexType name="organismNameType">
    <xs:annotation>
        <xs:documentation>The name type is used for source organism names.
        </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="type" use="required">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="common"/>
                        <xs:enumeration value="full"/>
                        <xs:enumeration value="scientific"/>
                        <xs:enumeration value="synonym"/>
                        <xs:enumeration value="abbreviation"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="statusType">
    <xs:annotation>
        <xs:documentation>The status attribute provides a known/unknown flag.
        </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="status" use="optional" default="known">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="known"/>
                        <xs:enumeration value="unknown"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

```

```

<!-- Name definition ends -->
<!-- Definition of the protein begins -->
<xs:complexType name="proteinType">
  <xs:annotation>
    <xs:documentation>The protein element stores all the information found in
      the DE line of a flatfile entry.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="name" type="proteinNameType" maxOccurs="unbounded"/>
    <xs:element name="domain" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>The domain list is equivalent to the INCLUDES
          section of the DE line.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="name" type="proteinNameType" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="component" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>The component list is equivalent to the CONTAINS
          section of the DE line.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="name" type="proteinNameType" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="type">
    <xs:simpleType>
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="fragment"/>
        <xs:enumeration value="fragments"/>
        <xs:enumeration value="version1"/>
        <xs:enumeration value="version2"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="evidence" type="xs:string" use="optional">
    <xs:annotation>
      <xs:documentation>This contains all evidences that are connected
        to the complete DE line.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>
<!-- Definition of the protein ends -->

```

```

<!-- Definition of the geneLocation begins -->
<xs:complexType name="geneLocationType">
  <xs:annotation>
    <xs:documentation>Defines the locations/origins of the shown
      sequence (OG line).</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="name" type="statusType" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="apicoplast"/>
        <xs:enumeration value="chloroplast"/>
        <xs:enumeration value="cyanelle"/>
        <xs:enumeration value="hydrogenosome"/>
        <xs:enumeration value="mitochondrion"/>
        <xs:enumeration value="non-photosynthetic plastid"/>
        <xs:enumeration value="nucleomorph"/>
        <xs:enumeration value="plasmid"/>
        <xs:enumeration value="plastid"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="evidence" type="xs:string" use="optional"/>
</xs:complexType>
<!-- Definition of the geneLocation ends -->
<!-- Citation type section begins -->
<xs:complexType name="citationType">
  <xs:annotation>
    <xs:documentation>The citation type stores all information about a citation.
      The equivalent information in the flatfile can be found in the RA (authors),
      RT (title), RX (PubMed/MEDLINE/DOI IDs) and RL (citation location information
      such journal name, volume numbers, pages, etc.) lines.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="title" type="xs:string" minOccurs="0">
      <xs:annotation>
        <xs:documentation>The title of the citation. Stored in the RT line in
          the flatfile format.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="editorList" type="nameListType" minOccurs="0">
      <xs:annotation>
        <xs:documentation>The editors of a book. Stored in the RL line in the
          flatfile format. Only valid for books. Example:
            RL (In) Magnusson S., Ottesen M., Foltmann B., Dano K.,
            RL Neurath H. (eds.);
            RL Regulatory proteolytic enzymes and their inhibitors, pp.163-172,
            RL Pergamon Press, New York (1978).
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>

```

```

        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="authorList" type="nameListType" minOccurs="0">
    <xs:annotation>
        <xs:documentation>The authors of the citation. Stored in the RA line in
        the flatfile format, except for citing citation where it is stored in
        the RL line. Example (standard citation):
            RA Galinier A., Bleicher F., Negre D., Perriere G., Duclos B.,
            RA Cozzone A.J., Cortay J.-C.;
        Example (citing citation):
            RL Unpublished results, cited by:
            RL Shelnutt J.A., Rousseau D.L., Dethmers J.K., Margoliash E.;
            RL Biochemistry 20:6485-6497(1981).
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="locator" type="xs:string" minOccurs="0">
    <xs:annotation>
        <xs:documentation>The location information of an electronic (or online)
        article. It is in most cases the unprocessed RL line of an electronic
        article. Examples:
            RL (In) Plant Gene Register PGR98-023.
            RL (In) Worm Breeder's Gazette 15(3):34(1998).
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="dbReference" type="dbReferenceType" minOccurs="0"
    maxOccurs="unbounded">
    <xs:annotation>
        <xs:documentation/>
    </xs:annotation>
</xs:element>
<xs:element name="citingCitation" type="citationType" minOccurs="0">
    <xs:annotation>
        <xs:documentation>Used by type: unpublished results.</xs:documentation>
    </xs:annotation>
</xs:element>
</xs:sequence>
<xs:attribute name="type" use="required">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="book"/>
            <xs:enumeration value="journal article"/>
            <xs:enumeration value="online journal article"/>
            <xs:enumeration value="patent"/>
            <xs:enumeration value="submission"/>
            <xs:enumeration value="thesis"/>
            <xs:enumeration value="unpublished observations"/>
            <xs:enumeration value="unpublished results"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>

```

```

        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name="date" use="optional">
    <xs:simpleType>
        <xs:union memberTypes="xs:date xs:gYearMonth xs:gYear"/>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name="name" type="xs:string" use="optional"/>
<xs:attribute name="volume" type="xs:string" use="optional"/>
<xs:attribute name="first" type="xs:string" use="optional"/>
<xs:attribute name="last" type="xs:string" use="optional"/>
<xs:attribute name="publisher" type="xs:string" use="optional"/>
<xs:attribute name="city" type="xs:string" use="optional"/>
<xs:attribute name="db" type="xs:string" use="optional"/>
<xs:attribute name="country" type="xs:string" use="optional"/>
<xs:attribute name="number" type="xs:string" use="optional">
    <xs:annotation>
        <xs:documentation>Used by type: patent.</xs:documentation>
    </xs:annotation>
</xs:attribute>
<xs:attribute name="institute" type="xs:string" use="optional">
    <xs:annotation>
        <xs:documentation>Used by type: thesis.</xs:documentation>
    </xs:annotation>
</xs:attribute>
</xs:complexType>
<xs:complexType name="consortiumType">
    <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>
<xs:complexType name="personType">
    <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>
<xs:complexType name="nameListType">
    <xs:choice maxOccurs="unbounded">
        <xs:element name="person" type="personType"/>
        <xs:element name="consortium" type="consortiumType"/>
    </xs:choice>
</xs:complexType>
<!-- Definitions for SPTTr's additional citation information begins -->
<xs:complexType name="sourceDataType">
    <xs:annotation>
        <xs:documentation>Contains all information about the source this citation is
referring to (RC line). The used child-element names are equivalent to the
tokens used in the RC line. Examples:
        RC STRAIN=Sprague-Dawley; TISSUE=Liver;
        RC STRAIN=Holstein; TISSUE=Lymph node, and Mammary gland;
        RC PLASMID=IncFII R100;
    </xs:documentation>
    </xs:annotation>

```

```

<xs:choice maxOccurs="unbounded">
  <xs:element name="species">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="ref" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="strain">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="evidence" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="plasmid">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="evidence" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="transposon">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="evidence" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="tissue">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="evidence" type="xs:string" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:choice>
</xs:complexType>
<xs:group name="sptrCitationGroup">
  <xs:annotation>

```

```

        <xs:documentation>Groups the scope (RP lines) and source data (RC lines)
                                lists.</xs:documentation>
</xs:annotation>
<xs:sequence>
  <xs:element name="scope" type="xs:string" maxOccurs="unbounded">
    <xs:annotation>
      <xs:documentation>Contains a scope regarding a citation. There is
                            no classification yet. (RP lines).</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="source" type="sourceDataType" minOccurs="0">
    <xs:annotation>
      <xs:documentation>Contains all information about the source this
                            citation is referring to (RC line).</xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:sequence>
</xs:group>
<!-- Definitions for SPTr's additional citation information ends -->
<xs:complexType name="referenceType">
  <xs:annotation>
    <xs:documentation>Stores all information of the reference block in SPTr
                        (RN, RP, RC, RX, RA, RT and RL line).</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="citation" type="citationType"/>
    <xs:group ref="sptrCitationGroup"/>
  </xs:sequence>
  <xs:attribute name="evidence" type="xs:string" use="optional"/>
  <xs:attribute name="key" type="xs:string" use="required"/>
</xs:complexType>
<!-- Citation type section ends -->
<!-- Comment definition begins -->
<xs:group name="bpcCommentGroup">
  <xs:sequence>
    <xs:element name="absorption" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="max" type="xs:string" minOccurs="0" maxOccurs="1"/>
          <xs:element name="text" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="kinetics" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="KM" type="xs:string" minOccurs="0"
                                maxOccurs="unbounded"/>
          <xs:element name="Vmax" type="xs:string" minOccurs="0"
                                maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:group>

```

```

        <xs:element name="text" type="xs:string" minOccurs="0"
                                maxOccurs="1"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="phDependence" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="redoxPotential" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="temperatureDependence" type="xs:string" minOccurs="0"
                                maxOccurs="1"/>
</xs:sequence>
</xs:group>

<xs:complexType name="commentType">
  <xs:annotation>
    <xs:documentation>The comment element stores all information found in the
    CC lines of the flatfile format. If there is a defined structure to the CC
    comment, the extracted is displayed in the various defined attributes and
    child-elements. See the documentation of these attributes/elements for more
    details.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="text" type="xs:string" minOccurs="0" maxOccurs="1">
      <xs:annotation>
        <xs:documentation>If a CC line type does not have a defined structure,
        the text of this comment is stored in the element.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:group ref="bpcCommentGroup"/>
    <xs:choice minOccurs="0" maxOccurs="1">
      <xs:sequence>
        <xs:element name="link" minOccurs="0" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>This stored the URIs defined in the WWW and
            FTP tokens of the database (online information in the XML format)
            CC comment type.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:attribute name="uri" type="xs:anyURI" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:choice>
    <xs:sequence>
      <xs:element name="location" type="locationType" minOccurs="0"
                                maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>The information of the mass spectrometry comment
          is stored in the attributes:
          -molWeight (molecular weight)
          -mwError (error of the molecular weight)
          -msMethod (the method used for the mass spectrometry)
        </xs:documentation>
      </xs:element>
    </xs:sequence>
  </xs:sequence>
</xs:complexType>

```

```

        -range (which amino acids were measured. It's not mentioned if the
        complete sequence as shown in the entry was measured)
        </xs:documentation>
    </xs:annotation>
</xs:element>
</xs:sequence>
<xs:sequence>
    <xs:element name="event" type="eventType" minOccurs="1" maxOccurs="4"/>
    <xs:element name="comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="isoform" type="isoformType" minOccurs="0"
                                                maxOccurs="unbounded"/>
</xs:sequence>
<xs:sequence>
    <xs:element name="interactant" type="interactantType" minOccurs="2"
                                                maxOccurs="2"/>
    <xs:element name="organismsDiffer" type="xs:boolean" minOccurs="1"
                                                default="false"/>
    <xs:element name="experiments" type="xs:integer" minOccurs="1"
                                                maxOccurs="1"/>
</xs:sequence>
</xs:choice>
<xs:element name="note" type="xs:string" minOccurs="0" maxOccurs="1">
    <xs:annotation>
        <xs:documentation>If a CC line type contains a "NOTE=", the text of
        that note is stored in this element.</xs:documentation>
    </xs:annotation>
</xs:element>
</xs:sequence>
<xs:attribute name="name" type="xs:string" use="optional">
    <xs:annotation>
        <xs:documentation>States the name of the online information if there
        is one.</xs:documentation>
    </xs:annotation>
</xs:attribute>
<xs:attribute name="mass" type="xs:float" use="optional">
    <xs:annotation>
        <xs:documentation>First the molecular weight which has been
        determined.</xs:documentation>
    </xs:annotation>
</xs:attribute>
<xs:attribute name="error" type="xs:string" use="optional">
    <xs:annotation>
        <xs:documentation>The accuracy with which the molecular weight has been
        measured.</xs:documentation>
    </xs:annotation>
</xs:attribute>
<xs:attribute name="method" type="xs:string" use="optional">
    <xs:annotation>
        <xs:documentation>The method which has been used. Common values are
        ELECTROSPRAY, MALDI, FAB and PLASMA DESORPTION.</xs:documentation>
    </xs:annotation>

```

```

    </xs:annotation>
</xs:attribute>
<xs:attribute name="status" type="xs:string" use="optional">
  <xs:annotation>
    <xs:documentation>Some comments have a status reflecting their reliability.
      Common values are BY SIMILARITY, POTENTIAL and PROBABLE.</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="locationType" type="xs:string" use="optional">
  <xs:annotation>
    <xs:documentation>Defines the type of the location where RNA editing takes
      place. Common values are "Displayed", "Not_applicable" and "Undetermined".
    </xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="type" use="required">
  <xs:annotation>
    <xs:documentation>Stores the type of a comment. These are simply lower case
      conversions of the flatfile CC comment topics, with two exceptions.
      &quot;PTM&quot; is an abbreviation and stands for &quot;posttranslational
      modification&quot; and the CC topic &quot;DATABASE&quot; is translated to
      &quot;online information&quot;, which is a more accurate description of the
      content of this comment.</xs:documentation>
  </xs:annotation>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="allergen"/>
      <xs:enumeration value="alternative products"/>
      <xs:enumeration value="biotechnology"/>
      <xs:enumeration value="biophysicochemical properties"/>
      <xs:enumeration value="catalytic activity"/>
      <xs:enumeration value="caution"/>
      <xs:enumeration value="cofactor"/>
      <xs:enumeration value="developmental stage"/>
      <xs:enumeration value="disease"/>
      <xs:enumeration value="domain"/>
      <xs:enumeration value="enzyme regulation"/>
      <xs:enumeration value="function"/>
      <xs:enumeration value="induction"/>
      <xs:enumeration value="miscellaneous"/>
      <xs:enumeration value="pathway"/>
      <xs:enumeration value="pharmaceutical"/>
      <xs:enumeration value="polymorphism"/>
      <xs:enumeration value="PTM"/>
      <xs:enumeration value="RNA editing"/>
      <xs:enumeration value="similarity"/>
      <xs:enumeration value="subcellular location"/>
      <xs:enumeration value="subunit"/>
      <xs:enumeration value="tissue specificity"/>
      <xs:enumeration value="toxic dose"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>

```

```

        <xs:enumeration value="online information"/>
        <xs:enumeration value="mass spectrometry"/>
        <xs:enumeration value="interaction"/>
    </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="evidence" type="xs:string" use="optional"/>
</xs:complexType>
<xs:complexType name="eventType">
    <xs:annotation>
        <xs:documentation>This element stores information about events that cause
                                an alternative product.</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="type" use="required">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="alternative splicing"/>
                        <xs:enumeration value="alternative initiation"/>
                        <xs:enumeration value="alternative promoter"/>
                        <xs:enumeration value="ribosomal frameshifting"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="namedIsoforms" type="xs:int" use="optional"/>
            <xs:attribute name="evidence" type="xs:string" use="optional"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="isoformType">
    <xs:annotation>
        <xs:documentation>Contains all information on a certain isoform including
                                references to possible features defining the sequence.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="id" type="xs:string" maxOccurs="unbounded"/>
        <xs:element name="name" maxOccurs="unbounded">
            <xs:complexType>
                <xs:simpleContent>
                    <xs:extension base="xs:string">
                        <xs:attribute name="evidence" type="xs:string" use="optional"/>
                    </xs:extension>
                </xs:simpleContent>
            </xs:complexType>
        </xs:element>
        <xs:element name="sequence">
            <xs:complexType>
                <xs:attribute name="type" use="required">
                    <xs:simpleType>

```

```

        <xs:restriction base="xs:string">
            <xs:enumeration value="not described"/>
            <xs:enumeration value="described"/>
            <xs:enumeration value="displayed"/>
            <xs:enumeration value="external"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
    <xs:attribute name="ref" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
<xs:element name="note" minOccurs="0">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="evidence" type="xs:string" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<!-- Comment definition ends -->
<!-- DB reference definition begins -->
<xs:complexType name="dbReferenceType">
    <xs:annotation>
        <xs:documentation>Database cross-references, equivalent to the flatfile
            format's DR line.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="property" type="propertyType" minOccurs="0"
            maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" type="xs:string" use="required">
        <xs:annotation>
            <xs:documentation>The name of the database this cross-reference is
                referring to.</xs:documentation>
        </xs:annotation>
    </xs:attribute>
    <xs:attribute name="id" type="xs:string" use="required">
        <xs:annotation>
            <xs:documentation>The ID referred to.</xs:documentation>
        </xs:annotation>
    </xs:attribute>
    <xs:attribute name="evidence" type="xs:string" use="optional"/>
    <xs:attribute name="key" type="xs:string" use="required"/>
</xs:complexType>
<xs:complexType name="propertyType">
    <xs:attribute name="type" type="xs:string" use="required"/>
    <xs:attribute name="value" type="xs:string" use="required"/>

```

```

</xs:complexType>
<!-- DB reference definition ends -->
<!-- Feature definition begins -->
<xs:complexType name="positionType">
  <xs:attribute name="position" type="xs:unsignedLong" use="optional"/>
  <xs:attribute name="status" use="optional" default="certain">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="certain"/>
        <xs:enumeration value="uncertain"/>
        <xs:enumeration value="less than"/>
        <xs:enumeration value="greater than"/>
        <xs:enumeration value="unknown"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>

<xs:complexType name="locationType">
  <xs:annotation>
    <xs:documentation>A location can be either a position or have
both a begin and end.</xs:documentation>
  </xs:annotation>
  <xs:choice>
    <xs:sequence>
      <xs:element name="begin" type="positionType" minOccurs="1"/>
      <xs:element name="end" type="positionType" minOccurs="1"/>
    </xs:sequence>
    <xs:element name="position" type="positionType"/>
  </xs:choice>
  <xs:attribute name="sequence" type="xs:string" use="optional"/>
</xs:complexType>

<xs:group name="interactantGroup">
  <xs:sequence>
    <xs:element name="id" type="xs:string" minOccurs="1"/>
    <xs:element name="label" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:group>
<xs:complexType name="interactantType">
  <xs:group ref="interactantGroup" minOccurs="0"/>
  <xs:attribute name="intactId" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="featureType">
  <xs:annotation>
    <xs:documentation>Currently there is only one basic feature type, but
this will change in future with enhancement of the FT line parsers.
  </xs:documentation>
  </xs:annotation>

```

```

<xs:sequence>
  <xs:element name="original" type="xs:string" minOccurs="0"/>
  <xs:element name="variation" type="xs:string" minOccurs="0"
    maxOccurs="unbounded"/>
  <xs:element name="location" type="locationType"/>
</xs:sequence>
<xs:attribute name="type" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="active site"/>
      <xs:enumeration value="binding site"/>
      <xs:enumeration value="calcium-binding region"/>
      <xs:enumeration value="chain"/>
      <xs:enumeration value="coiled-coil region"/>
      <xs:enumeration value="compositionally biased region"/>
      <xs:enumeration value="cross-link"/>
      <xs:enumeration value="disulfide bond"/>
      <xs:enumeration value="DNA-binding region"/>
      <xs:enumeration value="domain"/>
      <xs:enumeration value="glycosylation site"/>
      <xs:enumeration value="helix"/>
      <xs:enumeration value="initiator methionine"/>
      <xs:enumeration value="lipid moiety-binding region"/>
      <xs:enumeration value="metal ion-binding site"/>
      <xs:enumeration value="modified residue"/>
      <xs:enumeration value="mutagenesis site"/>
      <xs:enumeration value="non-consecutive residues"/>
      <xs:enumeration value="non-terminal residue"/>
      <xs:enumeration value="nucleotide phosphate-binding region"/>
      <xs:enumeration value="peptide"/>
      <xs:enumeration value="propeptide"/>
      <xs:enumeration value="region of interest"/>
      <xs:enumeration value="repeat"/>
      <xs:enumeration value="selenocysteine"/>
      <xs:enumeration value="sequence conflict"/>
      <xs:enumeration value="sequence variant"/>
      <xs:enumeration value="short sequence motif"/>
      <xs:enumeration value="signal peptide"/>
      <xs:enumeration value="site"/>
      <xs:enumeration value="splice variant"/>
      <xs:enumeration value="strand"/>
      <xs:enumeration value="topological domain"/>
      <xs:enumeration value="transit peptide"/>
      <xs:enumeration value="transmembrane region"/>
      <xs:enumeration value="turn"/>
      <xs:enumeration value="unsure residue"/>
      <xs:enumeration value="zinc finger region"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>

```

```

        <xs:attribute name="status" type="xs:string" use="optional"/>
        <xs:attribute name="id" type="xs:string" use="optional"/>
        <xs:attribute name="description" type="xs:string" use="optional"/>
        <xs:attribute name="evidence" type="xs:string" use="optional"/>
        <xs:attribute name="ref" type="xs:string" use="optional"/>
    </xs:complexType>
    <!-- Feature definition ends -->
    <!-- Organism definition begins -->
    <xs:complexType name="organismType">
        <xs:sequence>
            <xs:element name="name" type="organismNameType" maxOccurs="unbounded"/>
            <xs:element name="dbReference" type="dbReferenceType" maxOccurs="unbounded"/>
            <xs:element name="lineage" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="taxon" type="xs:string" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="key" type="xs:string" use="required"/>
    </xs:complexType>
    <!-- Organism definition ends -->
    <!-- Keyword definition begins -->
    <xs:complexType name="keywordType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="evidence" type="xs:string" use="optional"/>
                <xs:attribute name="id" type="xs:string" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!-- Keyword definition ends -->
    <!-- sequence definition ends -->
    <xs:complexType name="sequenceType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="length" type="xs:integer" use="required"/>
                <xs:attribute name="mass" type="xs:integer" use="required"/>
                <xs:attribute name="checksum" type="xs:string" use="required"/>
                <xs:attribute name="modified" type="xs:date" use="required"/>
                <xs:attribute name="version" type="xs:integer" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!-- sequence definition ends -->
    <!-- Evidence definition begins -->
    <xs:complexType name="evidenceType">
        <xs:annotation>
            <xs:documentation>The evidence element is equivalent to the actual evidence

```

```

        (**EV line).</xs:documentation>
</xs:annotation>
<xs:attribute name="category" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="curator"/>
      <xs:enumeration value="import"/>
      <xs:enumeration value="program"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="type" use="required" type="xs:string"/>
<xs:attribute name="attribute" type="xs:string" use="optional"/>
<xs:attribute name="date" type="xs:date" use="required"/>
<xs:attribute name="key" type="xs:string" use="required"/>
</xs:complexType>
<!-- Evidence definition ends -->
<!-- Entry type definition ends -->
<xs:element name="entry">
  <xs:annotation>
    <xs:documentation>A (public) SPTr entry</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="accession" type="xs:string" maxOccurs="unbounded"/>
      <xs:element name="name" type="xs:string" maxOccurs="unbounded"/>
      <xs:element name="protein" type="proteinType"/>
      <xs:element name="gene" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="geneNameType" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="organism" type="organismType" maxOccurs="unbounded"/>
      <xs:element name="organismHost" type="organismType" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element name="geneLocation" type="geneLocationType" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element name="reference" type="referenceType" maxOccurs="unbounded"/>
      <xs:element name="comment" type="commentType" nillable="true" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element name="dbReference" type="dbReferenceType" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element name="keyword" type="keywordType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="feature" type="featureType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="evidence" type="evidenceType" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element name="sequence" type="sequenceType"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

<xs:attribute name="dataset" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Swiss-Prot"/>
      <xs:enumeration value="TrEMBL"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="created" type="xs:date" use="required"/>
<xs:attribute name="modified" type="xs:date" use="required"/>
<xs:attribute name="version" type="xs:integer" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="copyright" type="xs:string"/>
<!-- Definition of the content of the root element "uniprot" -->
<xs:element name="uniprot">
  <xs:annotation>
    <xs:documentation>Contains a collection of SPTr entries.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="entry" maxOccurs="unbounded"/>
      <xs:element ref="copyright" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

B.5 InterPro Entry IPR003959

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE interprodb SYSTEM "http://srs.ebi.ac.uk/interpro.dtd">
<InterProEntrySet>
<interpro id="IPR003959" type="Domain" short_name="AAA_ATPase_core" protein_count="7917">
  <name>AAA ATPase, core</name>
  <abstract>&lt;p&gt;AAA ATPases (ATPases Associated with diverse cellular Activities)
form a large, functionally diverse protein family belonging to the AAA+ superfamily
of ring-shaped P-loop NTPases, which exert their activity through the energy-
dependent unfolding of macromolecules <cite idref="PUB00014778" />,
<cite idref="PUB00014779" />. These proteins are involved in a range of processes,
including protein degradation, membrane fusion, microtubule severing, peroxisome
biogenesis, signal transduction and the regulation of gene expression.
&lt;/p&gt;&lt;p&gt;AAA ATPases assemble into oligomeric assemblies (often hexamers)
that form a ring-shaped structure with a central pore. These proteins produce a
molecular motor that couples ATP binding and hydrolysis to changes in conformational
states that can be propagated through the assembly in order to act upon a target
substrate, either translocating or remodelling the substrate
<cite idref="PUB00033933" />. &lt;/p&gt; &lt;p&gt; AAA ATPases contain one or two
conserved ATP-binding domains, which contain two conserved motifs, A and B. These
ATP-binding domains are often attached to various other functional domains. The
functional variety seen between AAA ATPases is in part due to their extensive number
of accessory domains and factors, and to their variable organisation within
oligomeric assemblies, in addition to changes in key functional residues within the
ATPase domain itself.&lt;/p&gt; &lt;p&gt;More information about these proteins can
be found at Protein of the Month: AAA ATPases <cite idref="PUB00033938"/>.&lt;/p&gt;
</abstract>
<class_list>
  <classification id="GO:0005524" class_type="GO">
    <category>Molecular Function</category>
    <description>ATP binding</description>
  </classification>
</class_list>
<example_list />
<pub_list>
  <publication id="PUB00014778">
    <author_list>Koonin E.V., Aravind L., Leippe D.D., Iyer L.M.</author_list>
    <title>Evolutionary history and higher order classification of AAA+ ATPases.</title>
    <db_xref db="PUBMED" dbkey="15037234" /><journal>J. Struct. Biol.</journal>
    <location firstpage="11" lastpage="31" volume="146" issue="1-2" />
    <year>2004</year></publication>
  <publication id="PUB00014779">
    <author_list>Lupas A.N., Frickey T.</author_list>
    <title>Phylogenetic analysis of AAA proteins.</title>
    <db_xref db="PUBMED" dbkey="15037233" /><journal>J. Struct. Biol.</journal>
    <location firstpage="2" lastpage="10" volume="146" issue="1-2" />
    <year>2004</year>
  </publication>
  <publication id="PUB00033933">
```

```

    <author_list></author_list>
    <title>Proteasomes and their associated ATPases: A destructive combination.</title>
    <db_xref db="PUBMED" dbkey="16919475" /><journal>J. Struct. Biol.</journal>
    <year>2006</year>
  </publication>
  <publication id="PUB00033938">
    <author_list></author_list><title>Protein of the Month AAA ATPases.</title>
    <url>http://www.ebi.ac.uk/interpro/potm/2006_8/Page1.htm</url><year>2006</year>
  </publication>
</pub_list>
<parent_list><rel_ref ipr_ref="IPR003593" /></parent_list>
<contains><rel_ref ipr_ref="IPR003960" /></contains>
<found_in>
  <rel_ref ipr_ref="IPR000470" /><rel_ref ipr_ref="IPR000641" />
  <rel_ref ipr_ref="IPR001270" /><rel_ref ipr_ref="IPR001984" />
  <rel_ref ipr_ref="IPR004605" /><rel_ref ipr_ref="IPR004815" />
  <rel_ref ipr_ref="IPR012178" /><rel_ref ipr_ref="IPR012763" />
  <rel_ref ipr_ref="IPR013461" /><rel_ref ipr_ref="IPR014232" />
  <rel_ref ipr_ref="IPR014251" /><rel_ref ipr_ref="IPR014252" />
</found_in>
<member_list>
  <db_xref protein_count="7917" db="PFAM" dbkey="PF00004" name="AAA" />
</member_list>
<external_doc_list>
  <db_xref db="PANDIT" dbkey="PF00004" />
</external_doc_list>
<structure_db_links>
  <db_xref db="PDB" dbkey="1s3s" /><db_xref db="PDB" dbkey="1r7r" />
  <db_xref db="PDB" dbkey="1r6b" /><db_xref db="PDB" dbkey="1qzm" />
  <db_xref db="PDB" dbkey="1qvr" /><db_xref db="PDB" dbkey="1oz4" />
  <db_xref db="PDB" dbkey="1njg" /><db_xref db="PDB" dbkey="1njf" />
  <db_xref db="PDB" dbkey="1lv7" /><db_xref db="PDB" dbkey="1ksf" />
  <db_xref db="PDB" dbkey="1jr3" /><db_xref db="PDB" dbkey="1jvk" />
  <db_xref db="PDB" dbkey="1j7k" /><db_xref db="PDB" dbkey="1iy2" />
  <db_xref db="PDB" dbkey="1iy1" /><db_xref db="PDB" dbkey="1iy0" />
  <db_xref db="PDB" dbkey="1ixz" /><db_xref db="PDB" dbkey="1ixs" />
  <db_xref db="PDB" dbkey="1ixr" /><db_xref db="PDB" dbkey="1iqp" />
  <db_xref db="PDB" dbkey="1in8" /><db_xref db="PDB" dbkey="1in7" />
  <db_xref db="PDB" dbkey="1in6" /><db_xref db="PDB" dbkey="1in5" />
  <db_xref db="PDB" dbkey="1in4" /><db_xref db="PDB" dbkey="1hqc" />
  <db_xref db="PDB" dbkey="1e32" /><db_xref db="CATH" dbkey="3.40.50.300" />
  <db_xref db="CATH" dbkey="1.10.8.60" /><db_xref db="SCOP" dbkey="c.37.1.20" />
</structure_db_links>
<taxonomy_distribution>
  <taxon_data name="Fungi" proteins_count="832" />
  <taxon_data name="Human" proteins_count="132" />
  <taxon_data name="Mouse" proteins_count="132" />
  <taxon_data name="Virus" proteins_count="52" />
  <taxon_data name="Archaea" proteins_count="277" />
  <taxon_data name="Metazoa" proteins_count="1807" />

```

```
<taxon_data name="Bacteria" proteins_count="4407" />
<taxon_data name="Chordata" proteins_count="567" />
<taxon_data name="Nematoda" proteins_count="45" />
<taxon_data name="Eukaryota" proteins_count="3179" />
<taxon_data name="Fruit Fly" proteins_count="83" />
<taxon_data name="Rice spp." proteins_count="179" />
<taxon_data name="Arthropoda" proteins_count="287" />
<taxon_data name="Green Plants" proteins_count="696" />
<taxon_data name="Unclassified" proteins_count="2" />
<taxon_data name="Cyanobacteria" proteins_count="310" />
<taxon_data name="Plastid Group" proteins_count="1214" />
<taxon_data name="Other Eukaryotes" proteins_count="125" />
<taxon_data name="Arabidopsis thaliana" proteins_count="189" />
<taxon_data name="Caenorhabditis elegans" proteins_count="45" />
<taxon_data name="Synechocystis PCC 6803" proteins_count="12" />
<taxon_data name="Saccharomyces cerevisiae" proteins_count="35" />
</taxonomy_distribution>
<sec_list><sec_ac acc="IPR001939" /></sec_list>
</interpro>
</InterProEntrySet>
```

B.6 InterPro XMLDSS

```
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by ALEX KANAPIN
      (EMBL OUTSTATION THE EBI) -->

<!-- Root element of InterPro database-->
<!ELEMENT interprodb (release | interpro+ | deleted_entries)*>
<!-- Release information-->
<!ELEMENT release (dbinfo)+>
<!--The dbinfo block is used to store release information about the referenced databases,
      either member databases such as PFAM, or databases that are used in the production of
      InterPro such as TrEMBL. At least one of the release or date attributes should be
      present.-->
<!ELEMENT dbinfo EMPTY>
<!ATTLIST dbinfo
      dbname NMTOKEN #REQUIRED
      version CDATA #IMPLIED
      entry_count CDATA #IMPLIED
      file_date CDATA #IMPLIED
>
<!--The abstract, a manually curated free text area containing a summary of the current
      state of knowledge about the patterns that make up this InterPro entry. Layout markup
      within this block is converted to XML literal characters during the post-processing of
      the Oracle dump.-->
<!ELEMENT abstract (#PCDATA | cite | db_xref | taxon | reaction)*>
<!ELEMENT author_list (#PCDATA)>
<!ELEMENT book_title (#PCDATA)>
<!ELEMENT category (#PCDATA)>
<!ELEMENT child_list (rel_ref)+>
<!ELEMENT cite EMPTY>
<!ATTLIST cite
      idref CDATA #REQUIRED
>
<!ELEMENT class_list (classification+)>
<!--Represents classification in the Gene Ontology (www.geneontology.org), a heirarchical
      classification of gene product location, encapsulation and function.-->
<!ELEMENT classification (category, description)>
<!ATTLIST classification
      id CDATA #REQUIRED
      class_type CDATA #REQUIRED
>
<!ELEMENT contains (rel_ref)+>
<!ELEMENT db_xref EMPTY>
<!ATTLIST db_xref
      db (BLOCKS | CATH | CAZY | COG | COMe | EC | GO | INTERPRO | IUPHAR | MEROPS |
          MSDsite | PANDIT | PDB | PFAM | PIRSF | PRINTS | PRODOM | PROFILE | PROSITE |
          PROSITEDOC | PUBMED | SCOP | SMART | SMODEL | SSF | SWISSPROT | TIGRFAMs |
          TREMBL | PANTHER | GENE3D) #REQUIRED
      version CDATA #IMPLIED
      dbkey CDATA #IMPLIED
      name CDATA #IMPLIED
```

```

    protein_count CDATA #IMPLIED
  >
  <!--This element contains the accession number of a single deleted InterPro entry.-->
  <!ELEMENT del_ref (description)*>
  <!ATTLIST del_ref
    id CDATA #REQUIRED
  >
  <!--If present, this contains a list of deleted IPRs-->
  <!ELEMENT deleted_entries (del_ref)+>
  <!--Generic description node, just contains a block of text. Meaning depends upon
    relative context.-->
  <!ELEMENT description (#PCDATA)>
  <!ELEMENT example_list (example)*>
  <!ELEMENT example (#PCDATA | db_xref)*>
  <!--Examples of this InterPro entry hitting proteins from SWISS-PROT and TrEMBL.-->
  <!ELEMENT external_doc_list (db_xref)+>
  <!ELEMENT structure_db_links (db_xref)+>
  <!ELEMENT taxonomy_distribution (taxon_data)+>
  <!ELEMENT taxon_data (#PCDATA)>
  <!ELEMENT found_in (rel_ref)+>
  <!ELEMENT interpro (name | sec_list? | abstract | class_list? | example_list | pub_list |
    external_doc_list? | member_list | parent_list? | child_list? |
    contains* | found_in* | structure_db_links* | taxonomy_distribution*)+>
  <!ATTLIST interpro
    id CDATA #REQUIRED
    type NMTOKEN #REQUIRED
    short_name CDATA #REQUIRED
    protein_count CDATA #REQUIRED
  >
  <!ATTLIST taxon_data
    name CDATA #REQUIRED
    proteins_count CDATA #REQUIRED
  >
  <!ELEMENT journal (#PCDATA)>
  <!ELEMENT location EMPTY>
  <!ATTLIST location
    firstpage CDATA #IMPLIED
    lastpage CDATA #IMPLIED
    volume CDATA #IMPLIED
    issue CDATA #IMPLIED
  >
  <!ELEMENT member_list (db_xref)+>
  <!--This is actually a description of the entry, and not the name. The short name is held
    in the attribute list of the 'interpro' element in order to make parsing through the
    file for a given name more efficient.-->
  <!ELEMENT name (#PCDATA)>
  <!--Note - changed name of node from 'parlist' to 'parent_list'-->
  <!ELEMENT parent_list (rel_ref)>
  <!ELEMENT protein EMPTY>
  <!ATTLIST protein

```

```

    id CDATA #REQUIRED
  >
  <!--A list of publications used within this InterPro entry.-->
  <!ELEMENT pub_list (publication)*>
  <!--Represents a single published source of data used by the InterPro entry. It is
    referenced by the 'cite' tag within the abstract and various other places. This
    replaces the over-loaded publication tag that we had before and allows a much cleaner
    looking schema.-->
  <!ELEMENT publication (author_list | title? | db_xref? | journal? | book_title? |
    location? | url? | year)+>
  <!ATTLIST publication
    id CDATA #IMPLIED
  >
  <!ELEMENT reaction (#PCDATA)>
  <!--This is a reference to another InterPro entry-->
  <!ELEMENT rel_ref EMPTY>
  <!ATTLIST rel_ref
    ipr_ref CDATA #REQUIRED
    type CDATA #IMPLIED
  >
  <!--This block stores information that is specific to this release of the XML file.-->
  <!ELEMENT sec_ac EMPTY>
  <!ATTLIST sec_ac
    acc CDATA #REQUIRED
  >
  <!--Secondary accession numbers are stored in this list.-->
  <!ELEMENT sec_list (sec_ac+)>
  <!ELEMENT taxon (#PCDATA)>
  <!ATTLIST taxon
    tax_id CDATA #IMPLIED
  >
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT type (#PCDATA)>
  <!--This should contain a URL for an online resource relevant to the given publication.
    Note that the type is now restricted to the w3c schema uriReference, and that any
    contents must therefore comply with the definition for this type.-->
  <!ELEMENT url (#PCDATA)>
  <!ELEMENT year (#PCDATA)>

```

B.7 InterPro XMLDSS

```
<interprodb>
  <release>
    <dbinfo dbname="text" entry_count="text" file_date="text"
              version="text"/>
  </release>
  <interpro id="text" protein_count="text" short_name="text" type="text">
    <name>text</name>
    <contains>
      <rel_ref ipr_ref="text" type="text"/>
    </contains>
    <parent_list>
      <rel_ref ipr_ref="text" type="text"/>
    </parent_list>
    <child_list>
      <rel_ref ipr_ref="text" type="text"/>
    </child_list>
    <found_in>
      <rel_ref ipr_ref="text" type="text"/>
    </found_in>
    <member_list>
      <db_xref db="text" dbkey="text" name="text"
              protein_count="text" version="text"/>
    </member_list>
    <external_doc_list>
      <db_xref db="text" dbkey="text" name="text"
              protein_count="text" version="text"/>
    </external_doc_list>
    <example_list>
      <example>
        text
        <db_xref db="text" dbkey="text" name="text"
                protein_count="text" version="text"/>
        text
      </example>
    </example_list>
    <pub_list>
      <publication id="text">
        <location firstpage="text" issue="text" lastpage="text"
                  volume="text"/>
        <author_list>text</author_list>
        <url>text</url>
        <year>text</year>
        <book_title>text</book_title>
        <db_xref db="text" dbkey="text" name="text"
                protein_count="text" version="text"/>
        <journal>text</journal>
        <title>text</title>
      </publication>
    </pub_list>
  </interpro>
</interprodb>
```

```

</pub_list>
<abstract>
  text
  <taxon tax_id="text">text</taxon>
  text
  <db_xref db="text" dbkey="text" name="text"
            protein_count="text" version="text"/>
  text
  <reaction>text</reaction>
  text
  <cite idref="text"/>
  text
</abstract>
<structure_db_links>
  <db_xref db="text" dbkey="text" name="text"
            protein_count="text" version="text"/>
</structure_db_links>
<sec_list>
  <sec_ac acc="text"/>
</sec_list>
<class_list>
  <classification class_type="text" id="text">
    <description>text</description>
    <category>text</category>
  </classification>
</class_list>
<taxonomy_distribution>
  <taxon_data name="text" proteins_count="text">
    text</taxon_data>
</taxonomy_distribution>
</interpro>
<deleted_entries>
  <del_ref id="text">
    <description>text</description>
  </del_ref>
</deleted_entries>
</interprodb>

```