行政院國家科學委員會專題研究計畫　成果報告

Ontology 驅動之生物資訊探勘環境

計畫類別：個別型計畫
計畫編號：NSC93-2213-E-009-076-
執行期間：93 年 08 月 01 日至 94 年 07 月 31 日
執行單位：國立交通大學資訊科學學系(所)

計畫主持人：陳俊穎

計畫參與人員：林君翰 詹亦秋 劉許吉 林建宏

報告類型：精簡報告

處理方式：本計畫可公開查詢

中　華　民　國 94 年 10 月 26 日

行政院國家科學委員會補助專題研究計畫 **X 成果報告**
□期中進度報告

Ontology 驅動之生物資訊探勘環境

Towards Ontology Driven Data Exploration

計畫類別： x 個別型計畫　　□ 整合型計畫
計畫編號：NSC 93-2213-E-009-076-
執行期間：　93 年 8 月 1 日 至　94 年 7 月 31 日

計畫主持人：陳俊穎
共同主持人：
計畫參與人員： 林君翰 詹亦秋 劉許吉 林建宏

成果報告類型(依經費核定清單規定繳交)：x 精簡報告　□完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：

中　華　民　國　　94　年　　10 月　　24 日

Towards Ontology Driven Data Exploration

Jing-Ying Chen

Abstract

To successfully unveil complex biological phenomena embedded deeply inside mutually correlated data, scientists need to perform proper data transformation and analysis. However, due to abundance, diversity, and heterogeneity of existing data sources and analytical tools scattered around the Internet, conducting data exploration can be frustrating, requiring complicated and often cumbersome data manipulation, tool acquisition and customization, etc., before actual analysis tasks can be started. This figure does not include knowledge and skills one needs to obtain in order to make effective use of these diverse biological resources. This project realizes a rule-based bioinformatics resource integration framework that permits users to set up rules to customize and coordinate heterogeneous data sources and tools in a cohesive manner. In addition, by incorporating ontologies into the system, the user can define more intelligent rules to enable advanced visualization effects and data analysis actions. The resulting system can serve as a basis for an automated, ontology-assisted data exploration environment that, we believe, can reduce human intervention yet increase the likelihood of successful scientific discovery in the long run.

Keywords: ontology, data mining, bioinformatics, integration

1. Introduction

Bioinformatics has become an integral part of life sciences nowadays. Complex, microscopic biological phenomena that interest biologists but are deeply embedded in abundant, often noisy empirical data can only be revealed with the help of appropriate software tools on suitably prepared data. When it comes to correlating new observations with existing data accumulated over the years by the global biological research community, such as data from prior experiments, knowledge conveyed in existing literatures, and so on., advanced tools and techniques are needed for the job, possibly by combining existing tools over existing data sources in novel ways.

While such discipline of computer-aided scientific discovery has gained recognitions world wide, the road to automatic, large-scale, and high-throughput data analysis is hampered by the fact that existing data sources and tools are highly heterogeneous and distributed. It often takes substantial portion of a researcher’s efforts just to set up necessary data conversion and tools before he/she can focus on data analysis. When the direction of investigation changes due to new findings or inclusion of new (types of) data, another round of labor-intensive hand crafting is ahead.

To shift some of the burden from scientists, extensive efforts have been poured into integrated bioinformatics trying to tackle the heterogeneity problem. For brevity, by heterogeneity we also refer to resource diversity and distributed-ness. Approaches to heterogeneous resource integration come in many forms. One common approach is the unification of access interfaces to

heterogeneous data sources and tools. For example, NCBI Entriz [1] is one of the top used sites today that provide uniform query interface to various internally hosted biological databases. SRS [2] is another popular platform that can host multiple databases and tools, giving users a Web-based workbench for data querying and analysis.

As also pointed out in [3], these essentially top-down, centralized approaches usually focus on specific domains of interests and build tightly integrated systems supporting designated functionality. While they can result in more effective systems for the problems they are designed for, they can not cope with system change and diverse users' needs easily.

In contrast, a bottom-up, component-based approach to bioinformatics resource integration takes an opposite direction, in that for specific or novel research topics, researchers can quickly assemble available tools to provide desirable functionality. This component-based integration trend is actually taking form recently along with the Web Services movement, where the computing industry is actively building an interoperability platform using existing Internet and Web standards. There have been many Web Services initiatives proposed by bioinformatics community (e.g. [4] [5]). In near future, more biological resources will be brought under the Web Services umbrella.

Despite these integration efforts, the issues raised before are not resolved, as integration platforms only reduce the complexity of integrating data and tools to some extent. Because heterogeneity still remains, questions such as what a tool does and how to use it, etc., are still left unanswered. On the other hand, heterogeneity is not necessarily a bad thing, but rather an essential ingredient to enable novel discoveries, especially in a fast growing field like bioinformatics. Therefore, it appears we need to embrace heterogeneity and look for tool assistance to enhance our ability to cope with heterogeneity.

This project implements an integrated bioinformatics environment, called BioOrch, with an aim at approaching the goal. Basically, BioOrch is a component composition platform that supports multiple strategies for customization and composition of diverse biological resources at different levels of abstraction, ranging from low-level resource development using platform-specific programming languages (for developers at development time), to flexible rule-based gluing mechanism (for advanced users to compose novel and sophisticated tools at deploy time or run time), to GUI-enabled assembly interfaces (for general end-users to mix and match resources quickly). In addition, BioOrch regards human intervention, or actually ingenuity, as an essential part of data exploration process, and provides mechanisms for enhanced user interaction and scientific visualization. Finally, to help manage the heterogeneity and provide more usable and intelligent exploration support, BioOrch users can create different ontologies to model their interested problem domains and use this information together with the basic gluing mechanism to coordinate tools in a generic way.

2. The BioOrch Architecture

We call bioinformatics resources managed in BioOrch, including both data sources and analytical tools, as **services**. In addition, to enable uniform mechanisms for service customization and composition, services are required to provide standard, XML-based interfaces.

Services are managed by containers. BioOrch consists of a set of containers distributed across the Internet, as illustrated in Fig. 1. In the figure, each container is responsible of connecting the services it hosts to other services possibly reside in remote containers. The connections may include complex data translation and control flow arrangement. However, each service does not know which other services they interact with. In particular, a service only interacts with its context provided by the container. Such a service architecture guideline is used to ensure that each service is sufficiently self-contained and thus has better reusability across different problem domains.

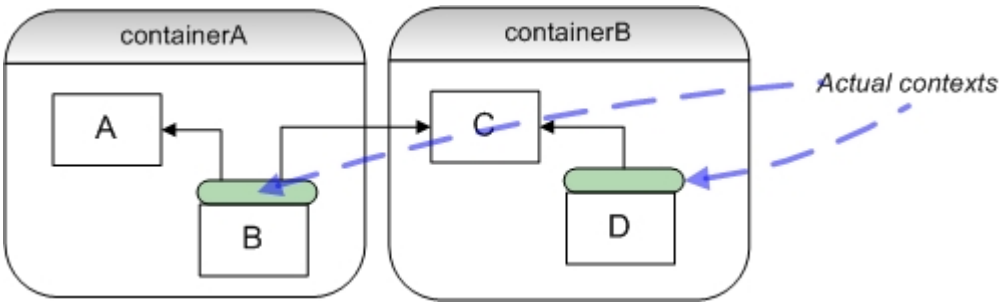

Figure 1. Basic service architecture that includes multiple autonomous service containers.

Service creation and composition is done through interaction incrementally with containers. Each container provides a network addressable interface that allows clients to manage the services it hosts. In addition to services, a container can also contain named templates according to which the container can create services instances. Below are some example templates.

```
<template name=”dataService” class=”bioorch.db.DataService” type="db/DataService"/>
<template name=”kMeans” class=”bioorch.tool.Kmeans” type="tool/Kmeans">
   <cntx name=”data” type=”db/DataService”/>
</template>
```

Note that the template format is container-specific. The example above implies a Java realization where the container will create Java objects based on their class names. The type attribute associated with each template is just an identifier (c.f. URI) that can be used by the container and/or developer to obtain further information about the service type, including I/O formats and semantics. As will become clear later, types are also important information for BioOrch to enable proper inter-service collaboration.

The <cntx> element above indicates that when the container creates a k-means service it should supply it with a service of type db/DataService. During service instantiation, the container needs to set up the context appropriately, although the binding can happen at design time, deploy time, or run time, and which service to bind may depend on user or other agents. The example below shows how to compose services dynamically.

```
<tool template=”dataService” path=”myDataMiner/dataServ”/>
<tool template=”kMeans” path=”myDataMiner/kMeans”>
   <cntx name=”data” path=”../dataServ”/>
   <config> configuration info </config>
</tool>
```

In general one can not expect to create new services by simply connecting available services directly without proper customization for each of them. As shown in previous example, a service may be customized by providing service-specific configuration information (through the <config> element). In addition, BioOrch provides more flexible mechanisms via XML-based rules. In particular, we define a rule-based programming language that can be used to create services. By placing these rule-based services inside a container and coordinate them with other ordinary services properly, as illustrated in Fig. 2, the composite service can function in new ways perceived by the composer. The figure only shows the general syntax of the rule language. We will describe some language details using examples in following sections. Note that the language is designed primarily for service coordination, rather than for general computation.

```
<rules>
   <cntx name=”...”/>
   <rule> ... </rule>
   </rule> ... </rule>
</rules>
```

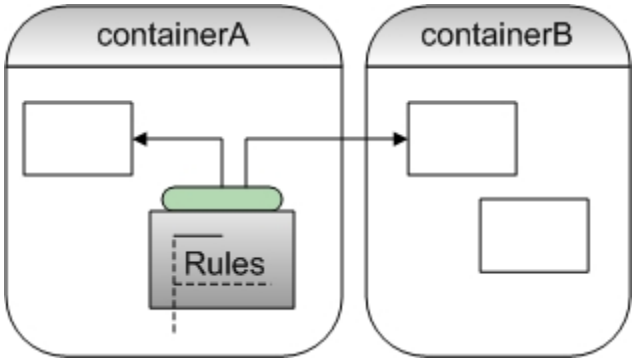

Figure 2. Services written in rules can work with other services.

One of the design objective of BioOrch is to use the same collaboration mechanisms to coordinate not only services exchanging XML messages, but also platform-dependent services such GUI applications that may interact with user and other services closely, possibly by exchanging platform-specific objects. This capability is necessary in domains such as scientific discovery where UI-rich visualization tools are inevitable.

For GUI-enabled services, or GUIs for short, we create a workbench service that can serve as workplace for other GUIs to reside in, as indicated in Fig. 3. Workbench itself is just a GUI so that it is subject to the same environmental responsibilities. Moreover, workbenches allow more

sophisticated desktop-wise collaboration among GUIs since workbenches can be configured with customization rules too. Desktop-wise collaboration will be described in the next section.

The notion of data exploration in this paper refers to a sequence of operations performed by a user or the system, where each operation may generate new data through queries or invocation of available tools on existing data. Since data exploration is an incremental process, some facilities are needed to hold intermediate investigation results. This can be easily done by introducing working sets services that organize data according to data types.

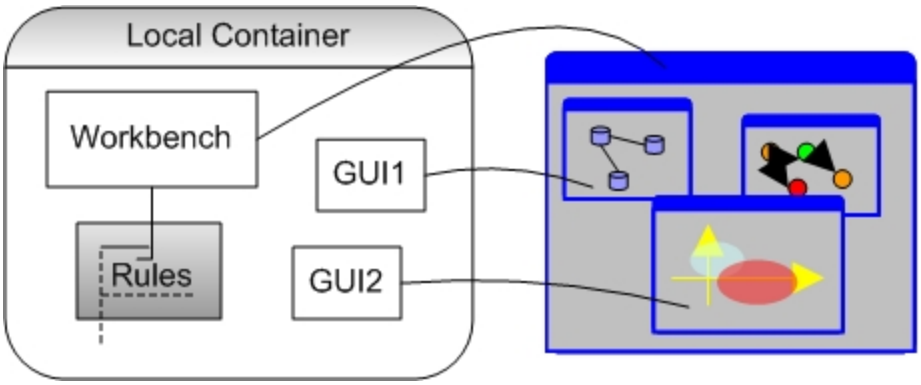

Figure 3. Workbench and GUI-enabled services.

We illustrate the use of working sets and other constructs discussed so far using a simplified Microarray data analysis system we developed internally to verify the BioOrch architecture. As shown in Fig. 4 below, we show relevant data types in rectangle boxes and tools in rounded boxes without giving actual XML representation for them.

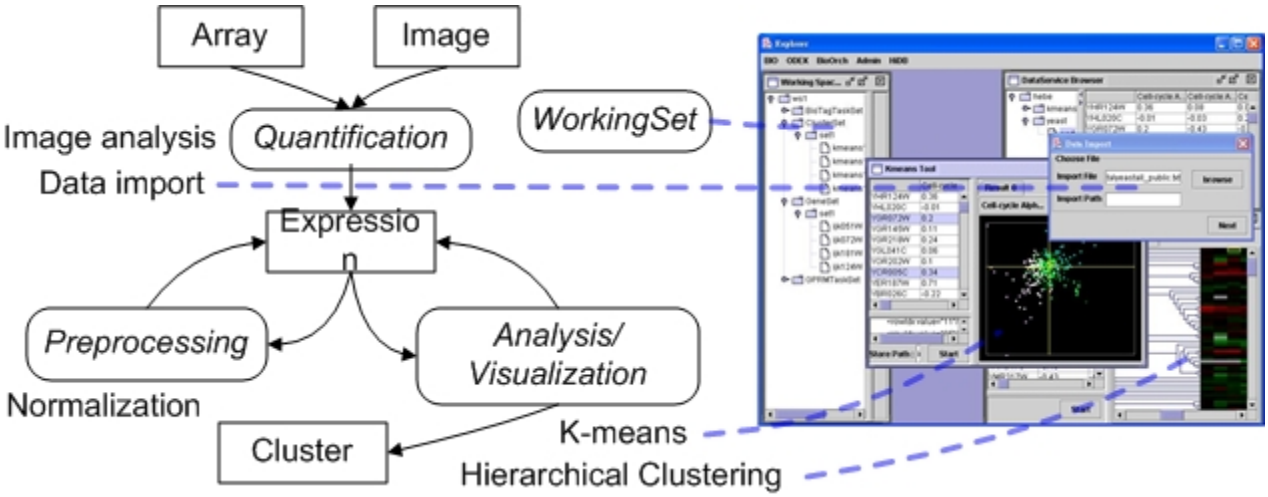

Figure 4. A simplified Microarray data analysis system implemented in BioOrch.

As an example, the rules shown in Fig. 5 below prescribe that when an Expression data is right clicked, candidate tools should be included in a popup menu along with other Expression-specific actions for user to choose.

More specifically, when the GUI generates a XML event, the immediate element under <if> is matched against the event structurally. If the match succeeds, additional actions will be taken, one by one, based on the elements that follow. For example, the <rw:ask> element send a <popup> message back to the workbench, effectively pops up the menu as indicated.

```
<rules>
  <rule>
    <if> <rightClick objType="Expression"/>
      <rw:ask>
        <popup>
          <action label="MyKmeansTool">
            <open path="tool/kmeans"/>
          </action>
          <action label="MyHierTool">
            <open path="tool/hier"/>
          </action>
        </popup>
      </rw:ask>
    </if>
    <ok/>
  </rule>
</rules>
```

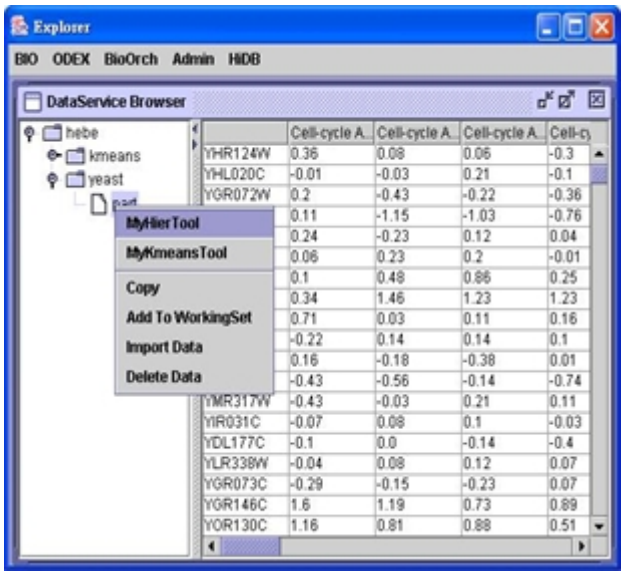

Figure 5. Desktop level collaboration in a workbench. Note that the both the menu bar and the menu being popped up do not belong to individual GUIs but synthesized by the workbench.

3. Ontology-Driven Data Exploration

The base system presented so far requires users to define specific rules that open and coordinate tools properly. As a result, new rules need to be created when similar data types or tools are added into the system.

To relieve users from such burden, we can equip BioOrch with ontology support using the facilities described above straightforwardly. Fig. 6 depicts such an ontology-driven data exploration architecture. In general, rules specified using ontologies are more abstract and reusable. Below we use several examples to demonstrate some advanced exploration scenarios.

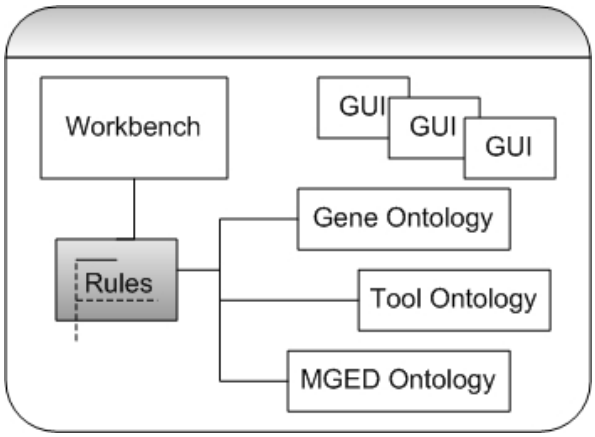

Figure 6. Equipping workbenches with ontology support.

The rules shown in Figure 7, for example, indicate that when a user right clicks a data set of a specific type, we would want all available tools that accept the data type can be included in the pop-up menu for user to choose. The rules achieve the goal using the user-defined ontology that is shown graphically (we do not show the actual XML representation here).

```
<rule>
  <if>
    <rightClick objType="@objType"/>
    <rw:eq>
      <rw:var name="actions"/>
      <rw:call name="otserv">
        <ontologyQuery>
          <rel entity="Tool" name="isA"/>
          <rel entity="@type" name="input"/>
        </ontologyQuery>
      </rw:call>
    </rw:eq>
  </if>
  <rw:ask>
    <popup>
      <rw:var name="actions"/>
    </popup>
  </rw:ask>
</rule>
```

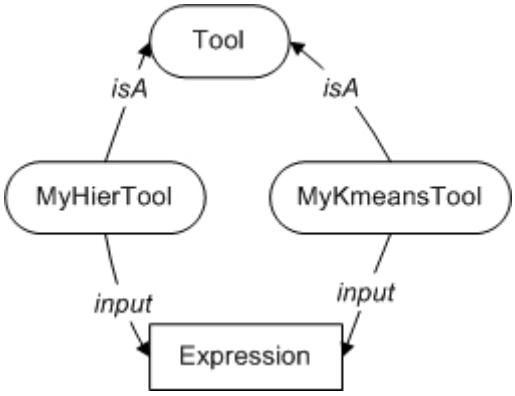

Figure 7. Workbench rules to tailor user interface.

It is possible and desirable to create new ontologies and rules and let BioOrch conduct more intelligent data exploration automatically. Consider the example in Figure 8, which shows that when a user investigates a particular data set, he/she can instruct the system to automatically find interesting correlations with other data sets that are currently in the working sets. A specific ontology can be created to characterize what an “interestingness-finding” tool should do: given a

data set (of a specific type it is designed for), it can compute some kind of correlations with other data sets (e.g., those in current working sets) and return a “normalized” interestingness scale (e.g. 0 to 1.0) and list them in the order of significance. Different interestingness-finding tools will have different internal logics, possibly by consulting with other services. The results from all these tools can be collected and sorted for the user for further examination (not shown).

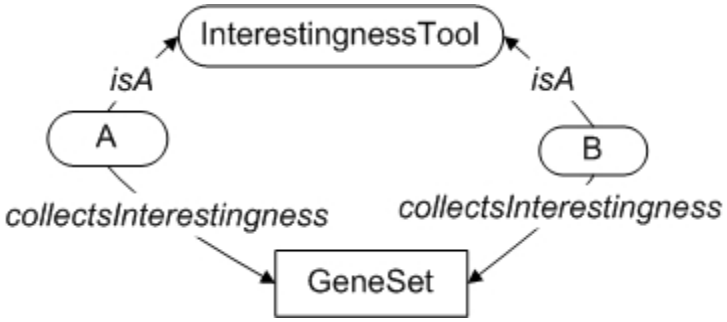

Figure 8. Ontology for advanced data exploration.

The rules for the example in Fig. 8 is given below:

```
<rule>
  <if>
    <interestingness objType="@objType" objID=”@objID”>
    <rw:eq>
      <rw:var name="intTools"/>
      <rw:call name="myToolOntology">
        <ontologyQuery>
          <rel entity="InterestingnessTool" name="isA"/>
          <rel entity="@objType" name="collectInterestingness"/>
        </ontologyQuery>
      </rw:call>
    </rw:eq>
  </if>
  <rw:ask>
    <broadcast scope=    ll?
      <tools> <rw:var name=”intTools”> </tools>
      <find dataset=”@objID”>
    </broadcast>
  </rw:ask>
</rule>
```

More elaborative scenarios can be conceived without complicated programming (but some modifications to existing tools may be needed). Once these ontologies and rules are set up properly (by end user or other domain experts), new tools and data types can be included in the system with minor update to the ontologies, and immediately extends the system in exploration dimension with the same exploration strategies .

4. Discussion and Related Work

The design and implementation of BioOrch is an ongoing process. The original emphasis of BioOrch development is on the construction of a bioinformatics service composition framework at the Internet scale, similar to the Web Services initiative [6]. The use of XML as the primary format for both accessing services and combining them is based on the long-term objective of enabling a more usable data exploration environment with advanced, intelligent data mining facilities, with less (total) development complexity. Our vision coincides with many objectives outlined in [7]. In particular, the concept of distributed and incremental learning presented there, where analysis should be carried out “in place” without transmitting data among multiple servers, is one of the direction planned for BioOrch in near future.

Using BioOrch as a generic service composition framework, we have implemented an initial prototype for Microarray data analysis, although further development is needed to integrate other commonly used Microarray resources. We have also integrated other bioinformatics services hosted by other researchers in the field, including a Motif secondary structure prediction and a literature mining system employing natural language processing techniques. Further collaboration with other researchers in the field is the top priority for our next step of development.

As the foundation of BioOrch is maturing, our emphasis has shifted to advanced data exploration support. We have incorporate some of the common ontologies, including Gene Ontology [8] and MGED [9] ontology (for Microarray experiments). There are systems that also employ ontologies to aid tool selection and composition, such as MyGrid [5], RiboWeb [10], etc. One major difference between BioOrch and these ontology-aware systems is that we provide generic support for consulting and combining multiple ontologies and use them in dynamic, potentially sophisticated service collaboration. Note that different ontologies can have different capabilities, for example permitting reasoning and validation. However, in the examples described about we force a common access interface to ontologies and adopt a set operation model that can work closely with working sets, yet provide a simple cross-ontology query mechanism.

5. Conclusions

We have proposed a flexible bioinformatics resource integration environment that permits customization and composition of services at development time, deployment time, and run time, targeting users with different skills and background. We showed how to incorporate ontologies into the system to help users model their own problem domains of interests and use these ontologies in conjunction with rule-based service gluing mechanism to enable more generic and intelligent data exploration support. Although not explicitly stated in this paper, the use of (standard) ontologies in the system can promote more meaningful service collaboration, and ultimately provide a global collaboration environment among researchers, not just services.

References

1. Entriz, NCBI, http://www.ncbi.nlm.nih.gov/Entrez/index.html
2. SRS, Lion bioscience SRS, http://www.lionbioscience.com/solutions/ products/srs
3. A. Siepel, A. Farmer, A. Tolopko, M. Zhuang, P. Mendes, W. Beavis and B. SobralV. Baran, M. Colonna, Bioinformatics. Vol. 17 no. 1, 83-94 (2001)
4. The NC BioGrid, http://www.ncbiogrid.org/
5. R.D. Stevens, A.J. Robinson and C.A. Gobel, Bioinformatics. Vol. 19 suppl. 1, i302-i304 (2003)
6. Web Services Activity, WWW Consortium, http://www.w3.org/2002/ws/
7. A. Silvescu, J. Reinoso-Castillo, C. Andorf, V. Honavar and D. Dobbs, Ontology-Driven Information Extraction and Knowledge Acquisition from Heterogeneous, Distributed Biological Data Sources. In: Proceedings of the IJCAI-2001 Workshop on Knowledge Discovery from Heterogeneous, Distributed, Autonomous, Dynamic Data and Knowledge Sources Acquisition from Heterogeneous, Distributed, Autonomous Biological Data Sources.
8. The Gene Ontology Consortium, Gene Ontology: tool for the unification of biology. Nature America Inc. 2000.
9. Microarray Gene Expression Data (MGED) Society, http://www.mged.org/
10. R.B. Altman, et al, RiboWeb : An Ontology-Based System for Collaborative Molecular Biology, IEEE Intelligent System, 68-76 (1999)
11. 劉許吉, Ontology Driven Bioinformatics Data Exploration, Master thesis, National Chiao Tung Univ, Computer and Information Science Dept., Jun. 2004.
12. 林建宏, Toward a Distributed Bioinformatics Environment: A Workflow approach, Master thesis, National Chiao Tung Univ, Computer and Information Science Dept., Jun. 2004.

Self Evaluation

The project objective is to investigate proper, flexible, and intelligent use of (user-defined) ontologies to guide scientists in exploring existing distributed and heterogeneous biological resources. The proposed approach is novel and unique compared to state-of-the-art research and development efforts. Although it is not easy to quantitatively assess the usefulness and effectiveness of our proposed method, largely due to the explorative nature of the data transformation and analysis process in bioinformatics field, we believe our approach paves the way to future systematic support of more intelligent and automatic data mining and knowledge discovery. The overall system is a joint work of several students, including [11] and [12], and is still developed towards a integrated bioinformatics environment. Although the project goal is achieved, however, further enrichment of the system and development of applications to real bioinformatics problems are needed.