行政院國家科學委員會專題研究計畫 成果報告

生醫分析系統之語意整合(第 3 年)
研究成果報告(完整版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 96-2221-E-468-011-MY3
執 行 期 間 ：98 年 08 月 01 日至 99 年 07 月 31 日
執 行 單 位 ：亞洲大學生物科技學系

計 畫 主 持 人：蔡進發
共 同 主 持 人：許承瑜、蔡崇豪
計畫參與人員：碩士班研究生-兼任助理人員：陳桓榆
碩士班研究生-兼任助理人員：郭宜忠
碩士班研究生-兼任助理人員：朱志峰
碩士班研究生-兼任助理人員：陳雅倩
博士班研究生-兼任助理人員：歐忠維

處 理 方 式 ：本計畫涉及專利或其他智慧財產權，2 年後可公開查詢

中 華 民 國 99 年 06 月 22 日

行政院國家科學委員會補助專題研究計畫 ■ 成 果 報 告 □期中進度報告

生醫分析系統之語意整合

計畫類別：■ 個別型計畫　　□ 整合型計畫
計畫編號：NSC 96－2221－E－468－011－MY3
執行期間：　98 年　08 月　01 日至　　99 年　07 月　31 日

計畫主持人：蔡進發教授
共同主持人：許承瑜教授、蔡崇豪教授
計畫參與人員：

成果報告類型(依經費核定清單規定繳交)：□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：亞洲大學

中　華　民　國　99　年　06　月　04　日

摘　要

本計畫建立一個語意生醫分析系統，允許生物醫學的研究人員透過以自然語言查詢方式，綜合查詢複雜的生物資訊數據及影像訊息。由於生醫資訊研究領域的多樣性，我們提出一個具有語意功能的描述語言(SCDL)作為一個描述問題的通用語言與樣式。在本計畫中我們提供許多查詢範例以及其對應的 SCDL。對於複雜的應用，多重 SCDL 查詢可以藉由控制架構做連結，我們亦提出一個可以映對使用者的查詢到一個或多個現存的應用服務。最後，我們將很快的將系統導成網頁服務系統並公開發表，這將使生醫資訊相關研究社群更容易分享大家的相關研究成果。

關鍵詞：語意計算、語意描述語言、生物醫學應用、知識系統

Abstract

In this project, a BioSemantic System is presented to bridge complex biological/biomedical research problems and computational solutions via semantic computing. Due to the diversity of problems in various research fields, a semantic capability description language (SCDL) is proposed to serve as a common language and generic form for problem formalization. Several queries as well as their corresponding SCDL descriptions are provided as examples in this study. For complex applications, multiple SCDL queries may be connected via control structures. We also present an algorithm to map a user request to one or more existing services if exist. Finally, we will rapidly deploy this system into a web application. This makes it easy for the research community to share the results obtained from proposed research.

***Keywords*:** Semantic computing; semantic capability description language; biomedical applications; knowledge base

目　錄

報告內容

本報告內容發表刊物名稱、卷期及出版日期：

刊物名稱：International Journal of Semantic Computing (IJSC)

卷期：Volume: 2, Issue: 2

出版日期：June 2008

頁碼：pp. 291-308

International Journal of Semantic Computing

World Scientific
www.worldscientific.com

USING SCDL FOR INTEGRATING TOOLS AND DATA FOR COMPLEX BIOMEDICAL APPLICATIONS

Shu Wang[1], Rouh-Mei Hu[2], Han C.W. Hsiao[3], David A. Hecht[3,4], Albert K.L. Ng[3],
Rong-Ming Chen[5], Phillip C.Y. Sheu[1,3], Jeffrey J.P. Tsai[3,6]

[1]Department of Electrical Engineering & Computer Science, University of California at Irvine, USA
[2]Department of Biotechnology, Asia University, Taiwan
[3]Department of Bioinformatics, Asia University, Taiwan
[4]Department of Chemistry, Southwestern College, USA
[5]Department of Computer Science & Information Engineering, National University of Tainan, Taiwan
[6]Department of Computer Science, University of Illinois at Chicago, USA

Received 20 October 2007
Revised 21 January 2008
Accepted 23 March 2008

Current bioinformatics tools or databases are very heterogeneous in terms of data formats, database schema, and terminologies. Additionally, most biomedical databases and analysis tools are scattered across different web sites making interoperability across such different services more difficult. It is desired that these diverse databases and analysis tools be normalized, integrated and encompassed with a semantic interface such that users of biological data and tools could communicate with the system in natural language and a workflow could be automatically generated and distributed into appropriate tools. In this paper, the BioSemantic System is presented to bridge complex biological/biomedical research problems and computational solutions via semantic computing. Due to the diversity of problems in various research fields, the semantic capability description language (SCDL) plays an important role as a common language and generic form for problem formalization. Several queries as well as their corresponding SCDL descriptions are provided as examples. For complex applications, multiple SCDL queries may be connected via control structures. We present an algorithm to map a user request to one or more existing services if exist.

Keywords: Semantic computing; semantic capability description language; biomedical applications

1. Introduction

Since the development of chain termination of a DNA sequencing method by Sanger and his colleagues in 1977 [1] and the subsequent development of computational methods for data retrieval and analysis [2-6], bioinformatics has become a new area of research. Many new experimental technologies have been rapidly developed that include: systematic analysis of gene expression profiles at the transcriptional level as well as the translational level using DNA microarrays, 2D protein gel electrophoresis and mass spectroscopy [7-9]; yeast two-hybrid system for detection of protein-protein interactions [10]; and NMR

or X-ray crystallography for the resolution of protein 3D structures [11]. These advances and new technologies have resulted in the rapid accumulation of immense amounts and types of data. These data can be found and data-mined in primary databases containing large-scale experimental data such as GenBank [12] and secondary databases providing biology knowledge such as Pfam [13], Transfact [14], GO [15] and KEGG [16].

Many tools have been developed for biomedical applications such as sequence alignment, gene finding, genome assembly, analysis of differential expression, protein structure alignment, protein structure prediction, prediction of protein-protein interactions, and modeling of evolution. While most of the databases and tools are available on web and easily accessible for users, current bioinformatics tools or databases are very heterogeneous in the following aspects:

(1) The input and output formats of different tools are generally restricted to fixed formats which are different.
(2) Databases were constructed on different systems or platforms in different formats (schema).
(3) The terminologies, such as gene name, gene ID or accession number, are heterogeneous.

As most biomedical databases and analysis tools are scattered across different web sites users have to partition their jobs manually into several tasks and do them separately. Training and technical support are often necessary for a user to design a correct workflow. Sometimes, different data models and document structures make two tools incompatible. Thus, interoperability across such different services becomes more difficult.

It is desired that different databases and analysis tools be normalized, integrated and encompassed with a semantic interface such that users of biological data and tools could communicate with the system in nature language and a workflow could be automatically created and distributed into appropriate tools. Biologists should be allowed to concentrate on their research and not the job of interfacing disparate systems and data sets. Usability is of importance to the future of bioinformatics tools. Increased usability has been linked to decreased training expenditures and time, as well as to improving human performance and productivity, ensuring better quality of work, and minimizing the risk of user error in data entry [17].

Instead of focusing on analysis of one characteristic of a gene/protein at a time, a new generation system for bioinformatics analysis must be capable of offering all information or knowledge regarding a gene/protein in response to a simple query. Moreover, a future bioinformatics system must be able to predict and to model basic principles of systems of higher complexity, like the interaction networks in cellular processes and the phenotypes of whole organisms.

In this paper we describe the BioSemantic System which is a framework that allows heterogeneous tools and data to be integrated via a service-oriented architecture (SOA) for declarative access. [18] This paper is organized as follows. In Section 2, the structure of the BioSemantic System is described. Section 3 gives an introduction to SCDL

(Semantic Capability Description Language), a language the BioSemantic System uses to describe what a service does rather than what a service needs, with examples illustrating some representative biomedical applications. Section 4 discusses how multiple SCDL queries can be connected via control structures for complex applications and how user requests are processed in BioSemantic. These are followed by discussion and conclusions in Section 5.

2. BioSemantic System

The ultimate objective of the BioSemantic System is to provide an integrated framework for prospective users to facilitate their works, such as biological and biomedical knowledge retrieval, management, discovery, capture, sharing, delivery and presentation. As illustrated in Figure 1, the system is able to provide a number of different web services (or service bases), which can be incrementally plugged in to the system. Each service has its own database as well as functions (or tools) to perform the tasks mentioned above. Accordingly, a common language for these supported service bases to communicate with the system is necessary to formalize and formulate a variety of problems. Semantic Capability Description Language (SCDL) is thus proposed to meet this requirement, and will be introduced in more detail in the next section.

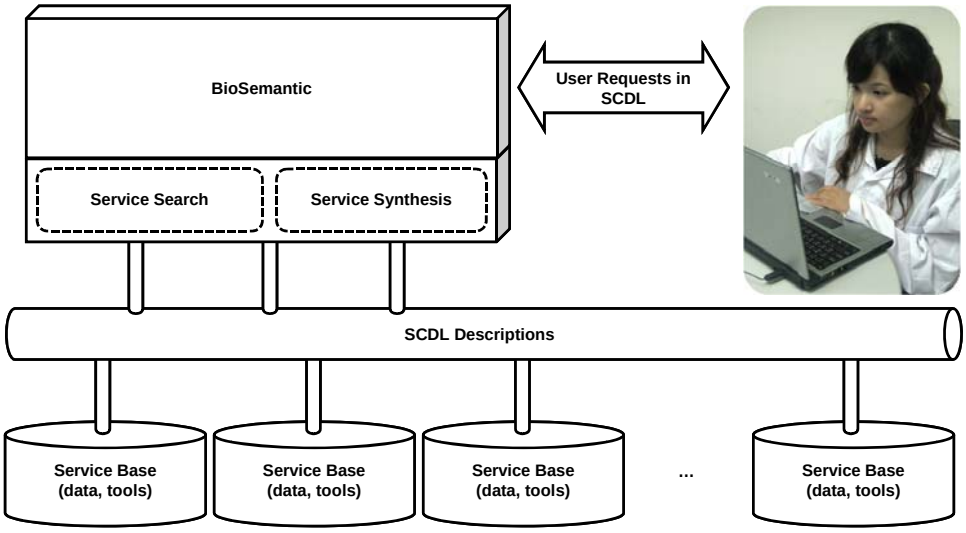

Figure 1. Service framework of BioSemantic System.

The current system relies on SemanticObjects™ [19] as the core technology, which is a development environment that provides an object relational layer on top of relational data sources that could assist designers generate a global schema to capture the semantics

of compound objects. Objects are defined within a global schema and wrapped by Java classes. Data are stored in different data sources and manipulated by SemanticObjects™ transparently without depending on further data sources. The global schema is mapped to local data sources by a mapping module. Using the Objects Designer, the user can declare object classes as well as define their operations and behaviors. The data associated with actual objects are stored in the data sources. An SNL (Structured Natural Language) parser is also provided to allow the user to compose their queries in SCDL, using Web Tools. Hence, a solution developed in SemanticObjects™ is extensible and user programmable based on SCDL. We envision that the system being used as follows. Users will define the problem by composing an SCDL query or an SCDL program. The SCDL request is parsed into a set of queries in SemanticObjects after service search and service synthesis are done.

3. Semantic Capability Description Language

Semantic Capability Description Language (SCDL) is an SQL-like description language that may be utilized to describe the functionality and capability of a database driven web service, with an objective to support automatic service composition. The syntax of SCDL for a web service *WS* is similar to that of SQL, as expressed in the following generic form:

SELECT outputs (O_1,…, O_m), aggregated-outputs ($f_1(A_1)$,…, $f_d(A_d)$)
FROM inputs (I_1,…, I_m), variables (R_1,…, R_n), other variables (S_1,…., S_k)
WHERE p(inputs, outputs, other variables)
GROUP BY (H_1,…, H_j)

where O_1,…, O_m are output objects; $f_1(A_1)$,…, $f_d(A_d)$ are possible aggregation functions, I_1,…, I_m are input objects; R_1,…, R_n are some range variables; S_1,…., S_k are sets that may be derived from the inputs and the range variables; H_1,…, H_j are the variables based on which to group the output objects; and p(inputs, outputs, other variables) is a formula that describes the relationships among the inputs, the outputs and the variables. Like SQL-99, SCDL allows variables to be typed, and it allows a function to be included as a condition in the WHERE clause. A major difference between SCDL and SQL is that SCDL allows "exponential variables", where the domain of an exponential variable could be the set of all subsets of an existing set, and variations of exponential variables to represent biological variables (We will see some examples later in the paper). The corresponding algebraic expression is as follows:

(1)

Note that while an SCDL expression *may* be executable, in practice it is often not realistic to do so. The language is utilized for the purpose of service search/synthesis only. By comparing the capability of a service expressed in SCDL and a query in SNL (that can be converted into SCDL), a match (one-to-one mapping) may be determined. In order to be

more flexible, the present system also accommodates a matching mechanism that maps a query into multiple services (one-to-many mapping). In the following sections we shall illustrate the use of SCDL for describing some typical biomedical applications.

3.1 Notations and Definitions

The notations used in the rest of Section 3 are listed below:

- A DNA sequence is a string of nucleotide bases q_i, where $q_i \in NA$ = {A, T, C, G}, i = 1, …, n; $n \in Z^+$;
- An RNA sequence is a string of nucleotide bases q_i, where $q_i \in NB$ = {A, U, C, G}, i = 1, …, n; $n \in Z^+$, and each element has an attribute called *charge* and an attribute called *molecular_weight*;
- A protein sequence is a string of amino acid a_i, where $a_i \in AA$ = {F, Y, C, W, L, P, H, Q, I, M, T, N, K, S, R, V, A, D, E, G}, i= 1, …, n; $n \in Z^+$;
- The predicate *blast*(*A,B*) is true if nucleotide_sequence *A blasts* nucleotide_sequence *B;*
- λ^{NA} designates the set of all possible DNA sequences;
- λ^{NB} designates the set of all possible RNA sequences;
- ξ^{AA} designates the set of all possible protein structures;
- The function (s_u, s_v).*Similar()* calculates the structural and/or sequence similarity to compare it with a predefined threshold *t*.

3.2 Primary Sequence Analysis

Primary sequence analyses of genes and proteins represent a fundamental class of applications that are routinely performed. These analyses depend solely on the underlying nucleic acid sequences for genes, and the amino acid sequence for proteins. These analyses often cover the BLAST, alignment, and prediction of protein families, domains and functions. Presented below are several examples chosen to demonstrate the wide applicability of SCDL and BioSemantic System to primary sequence, structural analyses and alignment problems.

3.2.1 BLAST Problem

Perhaps one of the most common tasks in biological research today is that of identifying genes and proteins related or similar to a particular sequence. The task is often performed with BLAST (NCBI, http://www.ncbi.nlm.nih.gov/blast/Blast.cgi). The SCDL describing a representative query is presented below:

Example 1. Find nucleotide or amino acid sequences from a database that are similar to a given sequence.

SELECT *N*

FROM $\lambda^{NB}(input)\ s, \lambda^{NB}(input)\ s'$
WHERE $blast(s, s')$

3.2.2 Sequence Alignment Problem

Another common problem is that of aligning multiple sequences of nucleic acids and/or proteins. The objective is to identify which regions are conserved and which are different. This problem becomes complicated by the fact that there can be intervening sequences of varying lengths that play little or no functional/structural role. The SCDL describing a representative query for finding subsequence pairs that match with statistical significance is presented below:

Example 2. Given two nucleotide or amino acid sequences, align based on matching residues and minimizing mis-matches and gaps.

SELECT (s_u, s_v)
FROM λ^{NA}(input) $u \in Q, \lambda^{NA}$(input) $v \in Q, \pounds^u s_u, \pounds^v s_v, float(input)\ t$
WHERE $u \neq v$ **AND** $(s_u, s_v).Match() > t$

where u and v belong to the set Q of sequences, s_u is any subsequence that may be derived from u, and s_v is any subsequence that may be derived from v. The problem may be solved using one or more of the following services:

- Align at NCBI (http://www.ncbi.nlm.nih.gov/blast/bl2seq/wblast2.cgi);
- CLUSTALW at EBI (http://www.ebi.ac.uk/Tools/clustalw/);
- Mutalin (http://bioinfo.genopole-toulouse.prd.fr/multalin/multalin.html);
- Weblogo (http://weblogo.berkeley.edu/logo.cgi);
- STRAP (http://www.charite.de/bioinf/strap/); and
- various Regulatory Sequence Analysis Tools (http://rsat.ulb.ac.be/rsat/).

3.3 Predict Protein Families, Domains and Functions

In biological systems the structure of a macromolecule such as a protein determines its function. Much effort has gone into analyses of primary sequences to predict the structure and function of expressed proteins. This includes the prediction of protein family and domain. Several representative examples are presented below.

While it is often the case that similar primary sequences result in similar 3D protein structures, in many cases *different* primary sequences also can result in similar 3D structures. Many tools and algorithms have been generated to perform these types of analyses and predictions. Following is a representative example.

Example 3. Find all proteins from a database that share a structural similarity to a given protein.

SELECT y

FROM ξ^{AA}(input) u, ξ^{AA}(input) $v \in db$, £$^{(u)}s_u$, £$^{(v)}s_v$, £$^{(u \times db)}(u, v)$, £$^{(u \times v)}p(s_u, s_v)$,
float(input) t
WHERE (s_u, s_v).*Similar()* > t

where u is an input protein that is a member of the protein structure database db, v is an input protein that is a member of the protein structure database db, s_u is any substructure that may be derived from u, and s_v is any substructure that may be derived from v . The function (s_u, s_v).*Similar* calculates the structural similarity to compare it with a predefined threshold t. The problem may be solved using one or more of the following services:

- FSSP (http://www.chem.admu.edu.ph/~nina/rosby/fssp.htm), and
- CATH (http://www.cathdb.info/).

3.4 Microarray Analysis

Microarray experiments are routinely used to study gene expression and metabolic pathways. They are also increasingly being used to identify biomarkers and to validate drug targets, as well as to study the metabolic and potential toxicological effects of compounds in a high-throughput mode. The amount of data generated from these experiments is astronomical and new tools like the BioSemantic System are needed for data-mining and knowledge retrieval and synthesis.

Example 4. Given a set of microarray data *c*, select genes that are significantly over expressed.

SELECT *g*
FROM setof-microarray(input) *MA*, *MA c*, float(input) *upper_threshold*
GROUP BY c.gene
HAVING *avg(c.value)* > *upper_threshold*

Example 5. Given a set of microarray data *c*, select genes that are significantly under expressed.

SELECT g
FROM setof-microarray(input) *MA*, *MA c*, float(input) l*ower_threshold*
GROUP BY *c*.gene
HAVING *avg(c.value)* < *lower_threshold*

3.5 Drug Discovery

According to current estimates, it takes about $1.3 billion and 12~15 years to bring a new drug to market. The reason is that there are many incredible difficulties at every step of the discovery and development process. In recent years, computational approaches have been successfully applied to enhance efficiency and productivity. Again, we see this as an

area where SCDL can be utilized to define complex problems and will ultimately lead to new tools and computational approaches

Perhaps the most fundamental problem in drug discovery is to find compounds that have similar structures or substructures to each other. These can be out of very large databases often > 10^5~10^6. There are many software products available to perform these queries.

Example 6. Find compounds from a data source that contain one or more substructures that are similar to a given substructure.

SELECT *c*
FROM ξ^{AA}(input) *s*, ξ^{AA}(input) $c \in db$, $£^c s_c$, float(input) *t*
WHERE (*s*, s_c).*Similar()* > *t*

where *c* is an input compound from the compound database *db*, $£^c$ denotes all possible substructures that may be derived from *c*, and s_c is any member of $£^c$. Given a substructure *s*, the function (*s*, s_c).*Similar()* calculates a score of similarity between s and s_c that is to be compared with a predefined threshold *t*.

4. Complex Queries

In many situations a complex analysis cannot be accomplished by a single query. Instead the use of multiple queries and actions involving some workflow are required. Simple control structures can be added to SCDL to connect several queries. For example, the following SCDL program segments may be composed and executed during a user session (via a higher level interface.)

Example 7. Given a cellular pathway *p* and a set of microarray data *c*, highlight the genes that are over expressed and the genes that are under-expressed in different colors.

SELECT *g*
FROM *setof-microarray(input) MA, MA c, float(input) upper_threshold*
GROUP BY c.gene
HAVING *avg(c.value) > upper_threshold*

CALL the result G^{over}

SELECT g
FROM *setof-microarray(input) MA, MA c, float(input) lower_threshold*
GROUP BY *c*.gene
HAVING avg(*c*.value) < *lower_threshold*

CALL the result G^{under}

Color (*p*, *g*, *color*)
FROM *pathway(input) p, Ψ(p) g*
WHERE if (*g* in G^{under}) *color is blue* else if (*g* in G^{over}) *color is yellow*

In the above, the function *Ψ(p)* returns the set of genes included in the pathway *p*.

Example 8. The risk of developing Alzheimer's Disease (AD) increases and the risk of developing Huntington Disease (HD) decreases as the average tangle density in the front cortex decreases. This hypothesis may be verified by deriving, for example, the following dataset:

Front-density	AD Possibility	HD Possibility
0-20	100	80
20-39	95	90
40-59	90	100
….	…	…

The following SCDL program segment can realize the above query composed by the user at run time:

Let *tdfc* be the tangle density in the front cortex.

SELECT *s*.AVG(*tdfc*)
FROM patient *s*

Call the result set_0;

SELECT *s.MAX(AVG(tdfc))*
FROM set_0 s

Call the result $tdfc_{max}$ // max tangle density in the front cortex

SELECT *s.MAX(AVG(tdfc))*
FROM set_0 s

Call the result $tdfc_{min}$ // min tangle density in the front cortex
$a = tdfc_{min}$

while (a < $tdfc_{max}$)
{
    $a = a + 20$;
    SELECT *s*.AVG(*tdfc*)
    FROM patient *s*
    WHERE $a - 20 < s$.AVG(*tdfc*) < a;

    Call the result set_1;

    SELECT *s*.AVG(*tdfc*)
    FROM patient *s*
    WHERE *s*.diagnosis("AD") and $a - 20 < s$.AVG(*tdfc*) < a;

    Call the result set_2;
    Calculate the ratio of set_2 and set_1;

Call the result rat_1;
Add (a, rat_1) to $result_1$;
}

$a = tdfc_{min}$;

while ($a < tdfc_{max}$)
{
$a = a + 20$;
SELECT s.AVG(*tdfc*)
FROM patient s
WHERE $a - 20 < s$.AVG(*tdfc*) $< a$;

Call the result set_3;

FROM patient s
WHERE s.diagnosis("HD") and $a - 20 < s$.AVG(*tdfc*) $< a$;

Call the result set_4;
Calculate the ratio of set_4 and set_3;
Call the result rat_2;
Add (a, rat2) to $result_2$;
}

If there is an algorithm readily available in the knowledge base, a description of the problem it solves needs to be matched by the program segment (that in the simplest case is an SCDL query) so that the algorithm can be used. By comparing the functionality of an algorithm expressed in SCDL as well and a given SCDL program segment, a match may be determined. This kind of mapping is called a "one-to-one" mapping. In the BioSemantic System, we have developed a matching mechanism that maps a query program into one or multiple algorithms if exist. The BioSemantic System separates two kinds of SCDL program: *SCDL-Pclient* and *SCDL-Psystem*. An *SCDL-Pclient* is an *ad hoc* SCDL program segment submitted by the client, and an *SCDL-Psystem* is a procedure predefined by the expert that is stored in the knowledge base. Every *SCDL-Psystem* has an implementation and a specification. While the implementation may be highly procedural compiled in any programming language, the specification is declarative and written in SCDL. On the other hand, an *SCDL_Pclient* is always written in SCDL. Our mapping is always carried out between an *SCDL-Pclient* and the specification of one or more *SCDL-Psystems*. Below are some examples taken from a bio-imaging application:

SCDL-Pclient 1:
SELECT *c* **FROM** 2^{blob} *c* **WHERE** *c.satallite-like()*

SCDL-Pclient 2:
SELECT *c* **FROM** 2^{blob} *c, float(input) beta* **WHERE** *c.satallite-like()* AND *c.brightness() > beta*

SCDL-Psystem1 [satallite-like-set-solver(r, g)]:
SELECT *r* **FROM** *set-of-blob(input) g,* 2^g *r* **WHERE** *r.satallite-like()*

SCDL-Psystem2 [brighter-than-set-solver (r, g, theta)]:
SELECT *s* **FROM** *set-of-blob(input) h, float(input) f,* 2^h *s* **WHERE** *s.brightness() > f*

The BioSemantic System defines five features for each *SCDL-Pclient* and the specification of each *SCDL-Psystem*: *Inputs*, *Outputs*, *Control-Structure*, *Actions* and *DataTypes*, where *Inputs* consist of a set of input variables, *Outputs* consist of a set of output variables, *Control-Structure* is the control flow, *Actions* consist of the actions and methods used in the program, and *DataTypes* are the types of the variables used. For example, the five features from Example 8 are:

Inputs: blob
Outputs: set-of-blob
Control-Structure: NO
Actions: satallite-like, brightness, … from … where…etc.
DataTypes: blob

Now, given an *SCDL-Pclient*, we will analyze and decompose it into one or more *SCDL-Psystems* in some sequence. The analysis is done in three steps. First, a program slicing technique [20] is applied based on the program dependency graph (PDG), where a program slice consists of the parts of a program that potentially affect the values computed at some point of interest referred to as a *slicing criterion* (Here we call it *SCDL-Pslice*). Second, we will calculate the "similarity" between an *SCDL-Pslice* and an *SCDL-Psystem* based on the features {*Inputs*, *Outputs*, *Control-Structure*, *Actions*, *DataTypes*} so that if the value of "similarity" is above certain threshold, the *SCDL-Pslice* and *SCDL-Psystem* would have some matching potential. Subsequently, we will use the test cases provided with the *SCDL-Psystem* to verify whether the *SCDL-Pslice* and *SCDL-Psystem* are really matched. If so, the implementation of the corresponding *SCDL-Psystem* can be used to substitute the *SCDL-Pslice*. The core of the proposed approach therefore consists of two problems: how to slice an *SCDL-Pclient* into *SCDL-Pslice*s and how to define the similarity between an *SCDL-Pslice* and an *SCDL-Psystem.*

Program slicing, first introduced by Weiser in 1979 [20], is a decomposition technique that extracts from those statements of a program relevant to a particular computation. Slicing was first developed to facilitate debugging, but it was then found helpful in many aspects of the software development life cycle, including debugging, testing, software measurement, program comprehension, maintenance, program parallelization, etc.

For a complicated language with pointers like C (or an object-oriented language like C++ or Java), program slicing is very difficult. In most cases, it cannot be done in real-time. For SCDL, fortunately, it is possible to be done efficiently and accurately. Because, first, SCDL does not contain pointers or references, so variables can be easy to identified and traced. More importantly, SCDL is a query-based programming language, thus a lot of computational details may be hidden via object relational queries.

Given an SCDL-Pslice or SCDL-Psystem, we can build a program dependency graph (PDG) by tracing the dataflow. Every node in the PDG is a subset of the program that includes only those program elements that may affect the values of the variables used in it. Previous research in program slicing has illustrated that a node in a PDG is a semantically meaningful subset of the original program.

As an example, consider the SCDL program segment (called Pclient3 hereafter) discussed in Example 8. By tracing the variable *a*, we can find two semantically complete code segments SCDL-Pslice2 and SCDL-Pslice3. The entire SCDL program segment is also identified as a semantically complete segment, called SCDL-Pslice1. Because there is no dependency between $result_1$ and $result_2$, SCDL-Pslice2 and SCDL-Pslice3 are independent of each other. The relationships among these slices are shown in Figure 2, and the features of Pclient3 are listed in Table 1.

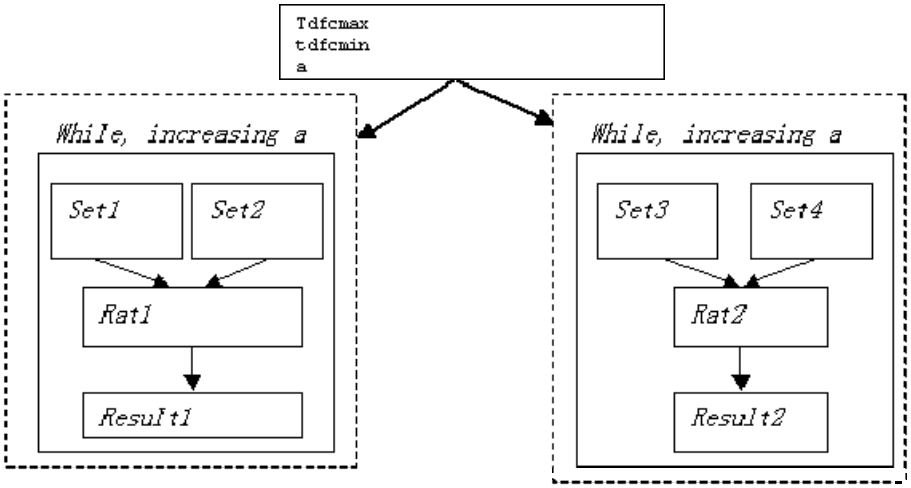

Figure 2. An example of program slicing.

We define the program similarity $S_{prog} \in [0, 1]$ as

$$S_{prog} = (\alpha S_{param} + \beta S_{control} + \gamma S_{action} + \delta S_{datatype}) / 1.111 \quad (1)$$

where S_{param} is the similarity between the inputs and outputs of the given two programs, $S_{control}$ is the similarity between the control structures of the two programs, S_{action} is the similarity between the actions in the two programs, and $S_{datatype}$ is the similarity between the data types employed in the two programs. If $S_{prog} = 1$, the two programs are exactly the same. In this study, we found the following assumptions should hold as restrictions on Equation 1.

1. S_{param} should have the highest priority in determining S_{prog}. It is quite clear that if two programs have different inputs and outputs, they basically cannot match. So here we assign $\alpha = 1$.
2. $S_{control}$ should have the second priority, followed by S_{action} and $S_{datatype}$. Thus, we have $1 > \beta > \gamma > \delta > 0$. For simplicity, we may assign $\beta = 0.1$, $\gamma = 0.01$, and $\delta = 0.001$.

We can define S_{param} as follows:

(2)

Table 1. Features of Pclient3.

	PSystem3	*PClient3:SCDL-Pslice1*	*PClient3:SCDL-Pslice2*	*PClient3:SCDL-Pslice3*
Inputs	*int, int, density, disease, set*	*int, int,set*	*int, int,set*	*int, int,set*
Outputs	*set*	*set*	*set*	*set*
Control-Structure	While(){};	While(){}; While(){}	While(){}	While(){}
Actions	show… from …; *AVG(density);* show… from …; *AVG(density);* *calculate the ratio of .. and ..;* *add… to…;*	show… from …; *AVG(tdfc);* show… from …; *AVG(tdfc);calculate the ratio of .. and ..; add… to…;* show… from …; *AVG(tdfc);* show… from …; *AVG(tdfc);calculate the ratio of .. and ..; add… to…;*	show… from …; *AVG(tdfc);* show… from …; *AVG(tdfc);calculate the ratio of .. and ..; add… to…;*	show… from …; *AVG(tdfc);* show… from …; *AVG(tdfc);calculate the ratio of .. and ..; add… to…;*
DataTypes,	*int, set, Patient*	*int Set Patient*	*int Set Patient*	*int Set Patient*

Consider an *SCDL-Pslice* that has control-structures $\{C_{A1}, C_{A2}, \ldots\}$ and an *SCDL-Psystem* that has control-structures $\{C_{B1}, C_{B2}, \ldots\}$. Note that we will combine those different control-structures having the same semantic meanings, e.g., **For each….** and **Loop…**, into one. We can define $S_{control}$ as follows:

(4)

S_{action} and $S_{datatype}$ can be calculated in a similar way. The similarity between Pclient3 and Psystem3 is calculated as follows.

PSystem3: Calculate Disease Trend
input: min, *max*, *density*, *Disease*, set_0
output: $result_1$

Let *a* be *min*;

while (a < *max*)
{
a = *a* + *delta*;
SELECT *s.AVG*(*density*)
FROM *patient s*
WHERE *a* – 20 < *s.AVG(density*) < *a*;

Call the result set_1;

SELECT s.AVG(density)
FROM *patient s*
WHERE s.diagnosis(Disease) and *a* – 20 < s.AVG(*density*) < *a*;

Call the result set_2;
Calculate the ratio of set_2 and set_1;
Call the result rat_1;
Add (*a*, rat_1) to $result_1$;
}

After program slicing, it is necessary to calculate the program similarity between <PClient3:SCDL-Pslice1, PSystem3>, <PClient3:SCDL-Pslice2, PSystem3>, etc. For example, the similarity between <PClient3:SCDL-Pslice1, PSystem3> can be computed as:

$$S_{prog} = (\alpha S_{param} + \beta S_{control} + \gamma S_{action} + \delta S_{datatype}) / 1.111$$

where $\alpha = 1$, $\beta = 0.1$, $\gamma = 0.01$, $\delta = 0.001$, and

$S_{param} = 3 + 1/7 = 0.571$
//3 inputs are the same, 1 output is the same, maximum parameter number is 6

$S_{control} = (1 + 1) / (1 + 2) = 0.667$
//Both have the control structure "while …". The only difference is their occurrences.

$S_{action} = (6 + 6) / (6 + 12) = 0.667$
//Both have the same actions but different occurrences

$S_{datatype} = 1$
//The only $S_{datatype}$ used in these two programs are *Patient* and *List*.

Therefore, $S_{prog} = (0.571 + 0.0667 + 0.00667 + 0.001) / 1.111 = 0.581$. Likewise, we can obtain the following, where the threshold is 0.6:

Similarity	Value
< *PClient3:SCDL-Pslice1, PSystem3*>	< 0.6

< *PClient3:SCDL-Pslice2, PSystem3*>	0.700
< *PClient3:SCDL-Pslice3, PSystem3*>	0.700
< *PClient1, PSystem1*>	1.000
< *PClient2, PSystem1*>	< 0.6
< *PClient1, PSystem2*>	< 0.6
< *PClient2, PSystem2*>	1.000

If we set the threshold to 0.6, the following potential program mappings can be found:

< *PClient3:SCDL-Pslice2, PSystem3*>
< *PClient3:SCDL-Pslice3, PSystem3*>
< *PClient3:SCDL-Pslice1, PSystem1*>
< *PClient3:SCDL-Pslice1, PSystem2*>

As illustrated in SCDL-Pclient2, an *SCDL-Pclient* may sometimes be an SCDL query. Because there is only one statement, so the slicing technique mentioned above cannot be applied. To address this, consider a single statement query in the following format:

SCDLQuery*:* SELECT[output+] from [input+] where [condition] (AND/OR[condition])* HAVING (condition)
condition: input.adjective() | logic composition of input.adjective()

In *SCDL-Pclient2*, for example, we can find that the verb is "SELECT", the output is "*r*", the input is "2^{blob}", the condition is "*c.satallite-like* AND c.brightness." We can *slice* the statement and extract two conditions as "c.satallite-like" and "c.brightness." Using the verb, output, and input, we can build the following table:

	verb	output	input	condition
Atomic_SCDL_Query1	Find	blob set	blob set	blob is satellite-like
Atomic_SCDL_Query2	Find	blob set	blob set	blob is brighter than theta

We call each of them an atomic SCDL query. As a result, we can find that the first query Atomic_SCDL_Query2 can be matched by SCDL-Psystem1, and the second one can be matched by SCDL-Psystem2. Finally we can take the intersection of the results returned by SCDL-Psystem1 and SCDL-Psystem2 to compute the final answers.

5. Discussions and Conclusions

In this paper, semantic capability description language (SCDL) as implemented in the BioSemantic System is presented to bridge complex biological and biomedical research problems and computational solutions. Some typical biological/biomedical services as well as their corresponding SCDL descriptions are given to demonstrate the power of SCDL in formalizing diverse problems as well as facilitating semantic retrieval.

For most "real-world" research, users will ask questions which are in essence complex queries. By comparing the functionality of an algorithm expressed in SCDL and

a given SCDL program segment, a matching mechanism is presented to perform a one-to-one mapping of a query program into one or multiple algorithms, if they exist.

In the BioSemantic System, two SCDL programs are provided for different purposes. An *SCDL-Pclient* is an *ad hoc* SCDL program segment submitted by the client; whereas an *SCDL-Psystem* is a procedure predefined by the expert that is stored in the knowledge base. To ensure a reliable matching between query programs and algorithms, an approach is presented as well to evaluate the so-called program similarity. The experimental results indicate that the proposed system including the matching mechanism provides a feasible solution to semantic computing especially for biomedicine.

It is hoped that with the addition of new SCDL definitions, the BioSemantic System will be able to address more and more complex as well as relevant biomedical research problems facilitating the ability of biomedical researchers to ultimately become more productive. As more cases and services are defined with SCDL, it is hoped that the BioSemantic System will be used for multiple purposes (besides research) that could include: medical diagnostics; patient medical records; training and education; as well as being a resource for troubleshooting and process optimization.

Acknowledgments

This work is supported in part under grant number NSC96-2221-E-468-011-MY3 from National Science Council, Taiwan. The views, opinions and/or findings contained in this report are those of the authors and should not be construed as an official National Science Council position, policy or decision unless so designated by other documentation.

References

[1] F. Sanger, S. Nicklen, and A. R. Coulson, DNA sequencing with chain-terminating inhibitors, *Proc. Natl Acad. Sci. USA* **74**(12) (1977) 5463-5467.

[2] P. H. Sellers, Pattern recognition in genetic sequences, *Proc. Natl Acad. Sci. USA* **76**(7) (1979) 3041.

[3] A. Sepulveda, M. Pieber, M. A. Soto, and J. C. Toha, Storage and retrieval of biomolecule sequences, *J Theor Biol.* **103**(2) (1983) 331-332.

[4] H. Peltola, H. Soderlund, and E. Ukkonen, Algorithms for the search of amino acid patterns in nucleic acid sequences, *Nucleic Acids Res.* **14**(1) (1986) 99-107.

[5] P. Gilna, L. J. Tomlinson, and C. Burks, Submission of nucleotide sequence data to GenBank, *J Gen Microbiol* **135**(7) (1989) 1779-1786.

[6] W. R. Pearson and W. Miller, Dynamic programming algorithms for biological sequence comparison, *Methods Enzymol.* **210**(1992) 575-601.

[7] J. Gollub, C. A. Ball, and G. Sherlock, The Stanford Microarray Database: a user's guide, *Methods Mol Biol.* **338**(2006) 191-208.

[8] G. R. Mishra, M. Suresh, K. Kumaran, N. Kannabiran, S. Suresh, P. Bala, K. Shivakumar, N. Anuradha, R. Reddy, T. M. Raghavan, S. Menon, G. Hanumanthu, M. Gupta, S. Upendran, S. Gupta, M. Mahesh, B. Jacob, P. Mathew, P. Chatterjee, K. S. Arun, S. Sharma, K. N. Chandrika, N. Deshpande, K. Palvankar, R. Raghavnath, R. Krishnakanth, H. Karathia, B. Rekha, R. Nayak, G. Vishnupriya, H. G. Kumar, M. Nagini, G. S. Kumar, R. Jose, P. Deepthi, S. S. Mohan, T. K. Gandhi, H. C. Harsha, K. S. Deshpande, M. Sarker, T. S. Prasad,

and A. Pandey, Human protein reference database-2006 update, *Nucleic Acids Res.* **34**(2006) D411-D414.

[9] O. Langella, M. Zivy, and J. Joets, The PROTICdb database for 2-DE proteomics, *Methods Mol Biol.* **355**(2007) 279-303.

[10] U. Stelzl, U. Worm, M. Lalowski, C. Haenig, F. H. Brembeck, H. Goehler, M. Stroedicke, M. Zenkner, A. Schoenherr, S. Koeppen, J. Timm, S. Mintzlaff, C. Abraham, N. Bock, S. Kietzmann, A. Goedde, E. Toksoz, A. Droege, S. Krobitsch, B. Korn, W. Birchmeier, H. Lehrach, and E. E. Wanker, A human protein-protein interaction network: a resource for annotating the proteome, *Cell* **122**(6) (2005) 957-968.

[11] H. M. Berman., J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, The Protein Data Bank, *Nucleic Acids Res.* **28**(2000) 235-242.

[12] D. L. Wheeler, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. Dicuccio, R. Edgar, S. Federhen, L. Y. Geer, Y. Kapustin, O. Khovayko, D. Landsman, D. J. Lipman, T. L. Madden, D. R. Maglott, J. Ostell, V. Miller, K. D. Pruitt, G. D. Schuler, E. Sequeira, S. T. Sherry, K. Sirotkin, A. Souvorov, G. Starchenko, R. L. Tatusov, T. A. Tatusova, L. Wagner, and E. Yaschenko, Database resources of the National Center for Biotechnology Information, *Nucleic Acids Res.* **33**(2005) D39-D45.

[13] R. D. Finn, J. Mistry, B. Schuster-Böckler, S. Griffiths-Jones, V. Hollich, T. Lassmann, S. Moxon, M. Marshall, A. Khanna, R. Durbin, S. R. Eddy, E. L. L. Sonnhammer, and A. Bateman, Pfam: clans, web tools and services, *Nucleic Acids Res.* **34**(2006) D247-D251.

[14] V. Matys, O. V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D. Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier, B. Lewicki-Potapov, H. Saxel, A. E. Kel, and E. Wingender, TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes, *Nucleic Acids Res.* **34**(2006) D108-110.

[15] Gene Ontology Consortium, The Gene Ontology (GO) project in 2006, *NucleicAcids Res.* **34**(2006) D322-D326.

[16] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa, KEGG: Kyoto Encyclopedia of Genes and Genomes, *Nucleic Acids Res.* **27**(1) (1999) 29-34.

[17] D. J. Mayhew, *Usability Engineering Lifecycle* (Morgan Kauffman, San Francisco, 1999).

[18] D. Hecht, R. M. Hu, R. M. Chen, J. W. Ou, C. Y. Hsu, H. Gong, K. L. Ng, H. C. W. Hsiao, J. J. P. Tsai, and P. C. Y. Sheu, BioSemantic system: applications of structured natural language to biological and biochemical research, *Proc. Workshop on Ambient Semantic Computing*, in conjunction with *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, Taichung, Taiwan, 2008, pp. 386-393.

[19] P. C. Y. Sheu and A. Kitazawa, From SemanticObjects to Semantic Software Engineering, *International Journal of Semantic Computing*, **1**(1) 2007 11-28.

[20] M. Weiser, Program Slicing, *Proceedings of 5th International Conference on Software Engineering*, San Diego, CA, Mar. 1981, pp. 439-449.

參考文獻

(與執行本計畫相關之著作發表)

Journal Papers

1. Han C.W. Hsiao*, S.H. Chen, P.C. Chang, and Jeffrey J.P. Tsai, Predicting subcellular locations of eukaryotic proteins using Bayesian and k-nearest neighbor classifiers, *Journal of Information Science and Engineering*, 24(5), September 2008, pp. 1361-1375. (SCI)
2. S. Wang, R.M. Hu, Han C. W. Hsiao, David A. Hecht, K.L. Ng, R.M. Chen, Phillip C. Y. Sheu*, and Jeffrey J. P. Tsai, Using SCDL for integrating tools and data for complex biomedical applications, *International Journal of Semantic Computing*, 2(2), June 2008, pp. 291-308.

Conference Papers

1. Alan C.H. Chen, Han C.W. Hsiao, Jeffrey J.P. Tsai, Phylogenetic analysis using nuclear-encoded mitochondrial proteins, *The 9th IEEE International Conference on Bioinformatics and Bioengineering*, Taichung, Taiwan, June 22-24, 2009, pp.374-377.
2. Charles C.N. Wang, David A. Hecht, Han C.W. Hsiao, Phillip C.Y. Sheu, Jeffrey J.P. Tsai, Describing dynamic biological systems in SPDL and SCDL, *The 9th IEEE International Conference on Bioinformatics and Bioengineering*, Taichung, Taiwan, June 22-24, 2009, pp.455-460.
3. Sean J.S. Lee, Han C.W. Hsiao, Jeffrey J.P. Tsai, Residue contact with dynamic time warping and least squares adjustment for protein structure alignment, *2009 Conference in Information Technology and Applications in Outlying Islands*, Kinmen, May 22-24, 2009, CD-ROM.
4. D. Hecht, R.M. Hu, R.M. Chen, J.W. Ou, C.Y. Hsu, H. Gong, K.L. Ng, Han C.W. Hsiao, Jeffrey J.P. Tsai, and Phillip C-Y Sheu, BioSemantic System: Applications of structured natural language to biological and biochemical research, *IEEE International Workshop on Ambient Semantic Computing in conjunction with IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, Taichung, Taiwan, June 11-13, 2008, pp. 386-393.
5. R.M. Chen, M.T. Hou, and Jeffrey J.P. Tsai, A Novel Approach for Motif Identification in Unaligned Molecular Sequences, *The 9th IEEE International Conference on Bioinformatics and Bioengineering*, Taichung, Taiwan, June 22-24, 2009, pp.378-381.
6. J.W. Ou, R.M. Hu, R.M. Chen, C.Y. Tang, Jeffrey J.P. Tsai, An Integrative Tool

for Gene Regulatory Network Reconstruction Based on Microarray Data, *The 9th IEEE International Conference on Bioinformatics and Bioengineering*, Taichung, Taiwan, June 22-24, 2009, pp.467-470.

7. J. W. Ou, R. M. Chen, R. M. Hu, and Jeffrey J.P. Tsai ,A Systematic Gene Expression Explorer Tool for Multiple and Paired Chips Analysis, *The Eleventh SDPS Transdisciplinary Conference on Integrated Systems, Design, & Process Science*, pp. 298-302, 2008.
8. C. Y. Hsu, R. M. Hu, R. M. Chen, and Jeffrey J.P. Tsai , IHCread: AN Automatic Immunohistochemistry Image Analysis Tool, *The Eleventh SDPS Transdisciplinary Conference on Integrated Systems, Design, & Process Science*, pp. 294-297, 2008.

計畫成果自評

本計畫至目前為止已經發表相關期刊論文 2 篇，國際研討會論文 8 篇，並且持續將最近的研究成果準備發表，整個計畫的進行尚很順利。另外，我們也已經建立了初步的生醫分析系統之語意整合專屬網站，我們將很快的公開發表，以提供給國內外相關研究的學者使用，分享大家的研究成果。相信本計畫成果未來對於生醫研究學者在進行研究有關各式各樣生物資訊工具的整合應用會有很大的助益。