J. Pestic. Sci., **31**(3), 282–288 (2006)

Review

(Special Topic)

G-language System as a platform for large-scale analysis of high-throughput omics data

Kazuharu ARAKAWA[†] and Masaru TOMITA*

Institute for Advanced Biosciences, Keio University, Fujisawa, Kanagawa 252–8520, Japan
[†] *Japan Society for the Promotion of Science*

(Received June 13, 2006)

The advent of high-throughput measurement technologies has resulted in the rapid accumulation of "omics" information including genome, transcriptome, proteome, and metabolome data. This increase in data acquisition has lead to a demand for an efficient computational platform for *in silico* analysis. The G-language software suite provides a comprehensive workbench for large-scale omics research and systems biology. The suite includes a bioinformatics research framework G-language Genome Analysis Environment, which contains a Gene Prediction Accuracy Classification benchmarking tool for the quantification of the sensitivity of genome informatics analysis methods to genome annotation completeness. Omics data processed in this environment can be visualized with KEGG-based pathway mapping web service, and Genome-based Modeling System enables automatic prototyping of metabolic pathway models from the genome. The software suite covers various domains of omics, with the goal of integrating all of these data for research into systems biology.

Keywords: bioinformatics, omics, G-language System, software tools, analysis pipelines, systems biology, post-genome.

Introduction

Molecular biology has experienced an unprecedented explosion of available data over the last decade with the introduction of high-throughput experimental technologies in the fields of genomics, transcriptomics, proteomics, and metabolomics, collectively termed with other-omes as "omics." The wealth of available biological data is best exemplified by the 858 databases listed in the Nucleic Acids Research Molecular Biology Database Collection 2006[1)] and over 2000 genome projects listed in the Genomes OnLine Database[2)] that are growing at an exponential rate (see the graph in http://www.ncbi.nih.gov/Genbank/genbankstats.html). Computational approaches and bioinformatics have already proven to be a successful and indispensable counterpart in molecular biology to utilize the huge masses of information, both in the hypothesis-free and hypothesis-generating processes.[3)] The first approach was primarily required in the genome projects such as in filtering and pre-processing of large-scale experimental data and for functional and structural characterization of genes and other molecular components. The latter is more crucial in the post-genome era, which compensates the research cycle of molecular biology by generating hypotheses that are difficult to formulate with only human intuition through the use of *in silico* data mining of the variety of complex omics data.

Despite of the existence of a myriad of established bioinformatics software tools for specific tasks, a combination of the software tools in a pipeline or a workflow interlinking the available tools and databases is required in order to perform bioinformatics research, just as bench biologists require experimental protocols integrating numerous procedures and apparatuses. For example, even the trivial task of homology searching consists of a workflow of querying databases for a sequence, retrieving the sequence record, running a BLAST[4)] search for the obtained sequence, parsing the output, and filtering the result to retrieve desired matches. EnsEMBL is a perfect example of a bioinformatics pipeline in action, combining numerous software tools and databases to achieve genome annotation in automation.[5)] Therefore a meta-level

* To whom correspondence should be addressed.
E-mail: mt@sfc.keio.ac.jp

software platform for the integration of multiple software tools and databases to develop workflows is essential for effective computational biology researches. As a platform, such an integrated workbench should be able to handle multiple database and software input and output (I/O) formats, should be equipped with rich user interfaces in the development of the workflows, and should be able to build and reuse analysis workflows. Tools for each of these purposes have become available mostly from open-source community based efforts, with Bio* Toolkit being the most successful example.[6,7] BioPerl,[8] BioJava, BioPython, and BioRuby projects hosted at the Open Bioinformatics Foundation (http://open-bio.org/, OBF) provide application programming interfaces (APIs) and libraries to handle biological databases and software tools in the popular programming languages used for bioinformatics research, Perl, Java, Python, and Ruby. Using these toolkits bioinformaticians need not worry about the variety of database formats, and can manipulate the data contents seamlessly as with any other data structures in the programming language. The European Molecular Biology Open Software Suite (EMBOSS) is another successful software package that is listed at the OBF, which is a comprehensive collection of more than 150 UNIX command line tools mostly aimed for bioinformatics sequence analysis.[9] Each program included in the EMBOSS package is accompanied by its I/O definition in Ajax Command Definitions (ACD) files, which pre-defines the required data formats and software parameters so that the command line tools can be easily interlinked to create a workflow, making the system interpret all necessary I/O specific requirements. The Taverna project provides a rich graphical user interface (GUI) to graphically create and run workflows by interconnecting available Simple Object Access Protocol (SOAP) web-services for bioinformatics.[10] Several other projects offer interfaces and means to create reusable workflows for life science researchers including Biopipe[3] and GPIPE.[11]

So far the main contribution of bioinformatics and computational biology has primarily belonged to and focused in the genomics domain, and although successful and important, the majority of the tasks would be categorized as data processing and hypothesis-free science. However, more inductive and deductive contributions from *in silico* researches are critical in the post-genome era, as the "systems biology" approach rapidly gains momentum in the anticipations of its potential to compensate the traditional descriptive reductionism approach in the understanding of life as a complex system.[12] Systems biology is a computationally intensive discipline by nature, requiring informatics for the data mining of omics data for hypothesis generation and computational modeling and simulation for *in silico* experiments for the complex behavior of the living systems.[13] In this way, a computational framework for post-genome sciences should enable seamless integration of the multitude of layers of omics data under a uniform interface for data mining, and to aid *in silico* modeling and observation of the analysis results in the context of cellular pathways and systems.

In order to overcome these issues, the G-language software suite developed at the Institute for Advanced Biosciences of Keio University provides an integrated workbench for researchers working with complex omics data. The project was started in 2001, initially with the development of a generic workbench for bioinformatics, G-language Genome Analysis Environment (G-language GAE).[14] Currently the software suite includes a benchmarking method for genome informatics analysis designated Gene Prediction Accuracy Classification (GPAC),[15] comprehensive application suite for the analysis of cDNAs,[16] KEGG-based pathway visualization system,[17] and a tool to automatically prototype a cell-wide metabolic pathway model from the genome sequence named the GEM (Genome-based Modeling) System.[18]

G-language Genome Analysis Environment

1. System and methods

G-language GAE is an integrated analysis and development environment for bioinformatics, with three main aims: (1) to construct an integrated environment for the development of analysis software, (2) to systematically accumulate existing analysis software, methodologies, and their results to avoid redundancy of efforts, and (3) to construct analysis workflows for frequent batch tasks. Therefore, from the user's point of view, G-language GAE may be thought of as a set of software libraries in the Perl programming language for database manipulation and genome informatics, or as a bioinformatics application equipped with over 200 tools and a GUI, or as a set of tools and interfaces for the agile development of bioinformatics software.

As a set of Perl libraries, G-language GAE consists of three main layers: I/O layer, application layer, and interface layer (see Fig. 1 for overall system architecture). The I/O modules provide APIs for common database access and manipulation for GenBank, EMBL, Fasta, Swiss-Prot, and other formats supported by BioPerl. A single gateway class mediates all I/O, and the types of databases and corresponding classes required are automatically interpreted. Data stored as an instance of the gateway class has uniform structure and can be handled with the same interfaces regardless of the data type. The application layer contains over 200 analysis applications using the data obtained through the I/O layer, such as genome sequence analyses and APIs for common bioinformatics software tools. We describe detailed examples in the following sections, and complete listing is available at our web site (http://www.g-language.org). Each analysis program in the application layer is implemented and provided as a native Perl function, most of which accept the data structure given by the I/O layer as the first argument. Using both of the I/O layer and the application layer, the interface layer contains APIs for the creation of graphical output and interfaces. As a set of software libraries, G-language GAE retains high level of compati-

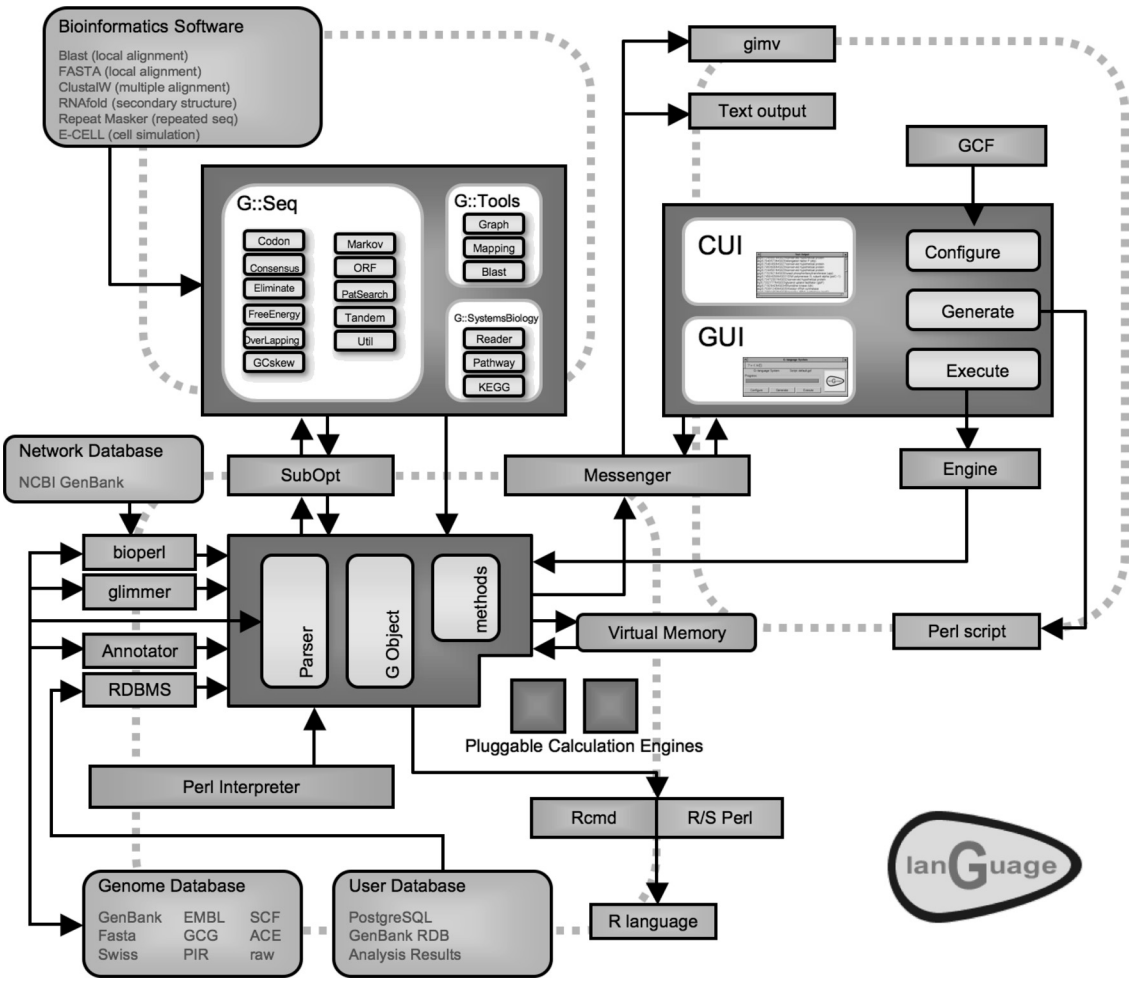

Fig. 1. System architecture of G-language GAE. The API is comprised of three layers: I/O layer for general manipulation of databases compatible with BioPerl, application layer containing over 200 analysis programs mainly for genome informatics, and interface layer for the creation of user interfaces and graphics.

bility with BioPerl. As described above, G-language GAE employs BioPerl in the I/O layer to support a variety of database formats, and therefore the BioPerl sequence object can be readily converted to G-language GAE type object by supplying the object to the gateway class constructor. The G-language GAE type object may also be reversely converted to a BioPerl sequence object by calling the conversion method from the instance. Moreover, all functions in the application layer that take sequence data as the first argument also directly accept BioPerl sequence object.

In addition to the library interface, G-language GAE is accessible as a command-line interpreter and as a GUI application. The command-line interpreter is an interactive shell that processes Perl line-by-line, equipped with basic UNIX shell functions such as command history, basic input editing, tab-completion for file names and functions in the application layer, and the execution of standard UNIX commands. All variables used during a session can be saved when quitting the interpreter, so that the users can start the next session with a consistent workspace. Since a significant fraction of bioinformatics research involves trial and error processes to search for the best procedures to solve a particular problem, this interface is especially suitable for the rapid testing of ideas with immediate responses. The entire procedure can be logged and exported as a working Perl script, therefore the trial and error processes directly results in a reusable piece of program that can be built upon. G-language GAE is easily pluggable with Perl scripts, where the system dynamically loads and provides subroutines as native functions, when they are present in the scripts deposited inside the plug-in folder. Thus scripts written using G-language GAE library interface can be immediately used from the interpreter.

The GUI of G-language GAE provides intuitive access to the analysis programs implemented in the application layer without writing a single line of code and is aimed for biologists that are not familiar with command-line interface and programming. Here the users can select and connect the analysis programs and create a workflow with customizable

parameters provided for each of the programs, basically by clicking through the available options. Workflow created with this interface can be readily exported as a Perl script for more thorough control using the library interface. However, the most novel feature of this GUI is its ability to load any Perl scripts with subroutines and to convert it into a GUI-based program. The system identifies the subroutines, interprets the I/O for that subroutine, and makes the subroutines controllable from the coherent GUI. In terms of software engineering, development of the GUI is one of the most time and effort consuming processes, and this GUI converter implemented with G-language GAE reduces this effort, therefore enabling bioinformatics software developers to send a Perl script to biologists that are not accustomed to programming and let them use the programs from the GUI. In this way, all three interchangeable interfaces of G-language GAE work seamlessly together to create a solid framework for a bioinformatics workbench.

2. Applications of G-language GAE in omics research

G-language GAE was initially targeted for genome sequence analysis and therefore the system is especially strong for this purpose. The GUI initially loads a workflow named Bacteria Analysis System, which is a collection of 25 analysis programs suitable for comprehensive investigation of complete bacterial genomes, with functions for the calculation of codon usage and its bias, for the detection of conserved sequences around the coding regions by several informatics measures such as entropy and information content, for the prediction of replication origin and terminus, calculation of GC content and skews, identification of palindromes and tandem repeats, and visualization of the genome as a map of genes. For example, Chen *et al.*[19] calculated the transition of GC content throughout the genome to observe the correlation of gene location with regions having relatively low or high GC contents, in their report of the complete sequencing of a large virulence plasmid pLVPK in *Kebsiella pneumoniae* CG43. Suzuki *et al.* proposes novel methods for the statistical analyses of synonymous codon usage bias[20,21] by comparing existing methods including the codon adaptation index (CAI),[22] the predicted expression level for characterizing predicted highly expressed genes (PHX),[23] the codon bias index (CBI),[24] the intrinsic codon deviation index (ICDI),[25] and the effective number of codons (N_c)[26] that are all implemented in the application layer.

Several works have already taken advantage of the ability of G-language GAE to formulate research workflows. Sato *et al.*[27] have developed an *in silico* analysis pipeline for the comprehensive detection of candidate genes that undergo stop codon readthrough event that produces extended proteins in eukaryotes, especially focusing on the presence of protein motifs or conserved domains in the 3′ untranslated regions. Yachie *et al.*[28] predicted non-coding antisense RNA genes in *Escherichia coli* genome using Gapped Markov Model Index (GMMI), and experimentally confirmed 12 transcripts. Both workflows implemented upon the G-language GAE are available upon request from the authors of the above works. In the pathway visualization tool[17] described below, systematic microarray results of 125 comprehensive deletion mutant strains of *E. coli*[29] were normalized, filtered, and clustered using the programs in the application layer, and visualized upon KEGG pathway maps.[30] Several other workflows aimed for the analysis of cDNAs containing pipelines for the detection of translation initiation/termination signals, multivariate analysis of codon usage, comparative study of amino acid composition, comparative homology-based modeling of the structures of product proteins, prediction of alternative splice forms, and metabolic pathway reconstruction and alignment have been packaged and distributed.[16]

3. Availability

G-language GAE is freely available with the entire set of source code under the open-source GNU General Public License Vesion 2, at our web site (http://www.g-language.org) with documentations about the software. Software packages are available for Windows, Linux, and MacOS X, but we also recommend using Knoppix for Bio (KNOB) live Linux CD available at http://knob.sourceforge.jp/. The KNOB project organized by Itoshi Nikaido is developing a free fully functional Linux distribution that boots and runs completely from the CD-ROM in almost all personal computers without affecting the existing operating systems and is bundled with numerous bioinformatics software packages such as EMBOSS, BioPerl, BioPython, BioRuby, BLAST, and G-language GAE. Using KNOB, there is no need to install additional software packages and the users can use G-language GAE together with other bioinformatics tools.

Gene Prediction Accuracy Classification

Computational “dry” biology deals with biological information and data acquired by “wet” experimental biology of the organisms and organic components. In this respect, the primary data source for biology *in silico* is innately secondary data. Usually these data further undergo several additional *in silico* steps, as typified by functionally annotated sequence data that are central to omics researches. Although thorough validations are conducted throughout the data preparation processes in order to assure the quality of information, a certain order of error rate is inevitable. To name a few, functional genome annotation is prone to error in the processes of sequencing,[31] detection of the open reading frames (ORFs),[32,33] characterization of genes,[34] and curation of annotations.[35] Therefore, genome informatics methods with extremely high sensitivity are likely to be affected by the innate errors of the primary data, which could possibly result in erroneous outcomes. This is especially critical for comparative genomics, since the complete genomes from different organisms used in these studies have varying levels of annotation completeness,

and the sensitivity of the analysis method should be assured for all target genomes. GPAC test included with the G-language software suite[15] provides a means to quantitatively benchmark the sensitivity of genome informatics analysis with regard to the annotation completeness of the genomes.

GPAC firstly classifies genes into five categories according to the annotation credibility levels as the following: group 1 consisting of all genes other than hypothetical ORFs, group 2 consisting of all genes other than hypothetical and putative ORFs, group 3 consisting only of conserved ORFs as characterized by the NCBI Clusters of Orthologous Groups (COGs) database,[36] group 4 consisting only of conserved functional ORFs, and group 5 consisting of functional, putative, and conserved hypothetical ORFs. Then the analysis being benchmarked is repeated using the selected set of genes for each of the five groups and the genes eliminated by the selection, and the sensitivity of the analysis can be observed as the deviations of the results of the selected groups compared with the original result which targets all genes in the genome. The degree of deviation can be statistically quantified by the bootstrap method. Users may choose alternative classification scheme defined using the evidence codes of Gene Ontology Annotation (GOA)[37,38] instead of the five gene groups defined above.

GPAC test of the calculation of average gene length using an old version of *E. coli* genome (U00096 18-NOV-1998) revealed that this simple calculation is actually sensitive, resulting in longer average gene length above the standard deviation computed by the bootstrap test when including hypothetical genes, therefore exclusion of this set of genes is advised for comparative study. Latest *E. coli* genomes significantly reduced the number of hypothetical ORFs, and GPAC test using the latest versions do not determine the calculation of average gene length as being sensitive. In this way, GPAC test is convenient for the selection of data and for the improvement of analysis methods in the preparation of *in silico* researches.

Pathway Visualization Tool

Comprehensive omics dataset based on high-throughput measurement such as transcriptome, proteome, and metabolome provides practical information about the cell-wide activity of the layers analyzed. However, systematic interpretation and understanding of omics data is often difficult, given the huge amount of data and the intricacy of the underlying physiological network that interconnects the compositional molecules. Scientific visualization of such data with a cellular context is a potential technology to aid human understanding of complex phenomena of the whole cell with the systems biology approach. Several software tools exist for the visualization of biological data; for example, Cytoscape[39] draws biological interaction network graphs, ArrayXPath,[40] VitaPad,[41] and GenMAPP[42] maps microarray data to pathway diagrams, and BioCyc Omics Viewer[43] and KEGG API[30] provides interfaces to map given data onto the pathway databases.

In order to observe systematic properties of the whole cell, it is desirable for a visualization tool to be able to simultaneously map omics data from different layers such as transcriptome, proteome, and metabolome, and to be mapped onto familiar pathway diagrams as opposed to automatic layouts *de novo*. Pathway visualization tool of the G-language software suite[17] provided as a web service (http://www.g-language.org/data/marray/) simultaneously maps complex omics data including genes, mRNAs, proteins, and metabolites onto KEGG pathway diagrams in a single vector graphic, and it is especially advantageous when systematically observing the results from computer simulations of cellular pathway models. Given comma-delimited name-value pairs of the molecules, with common or canonical name for genes, EC number for enzymes, and KEGG compound ID for metabolites, a pathway diagram is generated as a FLASH(SWF) vector image with corresponding objects color coded. Color values are integers from 1 to 100, which represent a red to green spectrum for genes/mRNAs/proteins, and a blue to yellow spectrum for metabolites. Heteropolymeric enzymes with multiple subunits are correctly represented by subdividing the box representing the enzyme. Using this tool, transcriptome data of 38 two-component regulatory system mutants[44] of *E. coli* as well as 125 carbon metabolism mutants[29] are visualized with 104 pathway diagrams.

GEM System

Dynamic behaviors of the living systems arise as a result of complex nonlinear interactions of the underlying molecular components, and computational simulation is necessary in order to capture the non-intuitive outcomes. A key process in this aspect is the system-level integration and modeling based on the reservoir of knowledge accumulated by traditional reductive approaches and the recent omics data from different layers of biology. However, the majority of the tasks required during current computational modeling for systems biology requires time-consuming manual operations, resulting in a major bottleneck in systems biology research. The GEM System of the G-language software suite enables automatic prototyping of cell-wide metabolic pathway models, integrating various omics databases using the genome sequence as its primary input and references during the integration processes. The resulting model is generated in the standard Systems Biology Markup Language (SBML) format,[45] ready for simulation using software environments such as E-Cell.[46–48]

Starting from a genome flatfile, the GEM System firstly matches all genes to the product proteins or enzymes by combined method of annotation reference, homology search, and orthology search using Swiss-Prot,[49] COGs, KEGG, and WIT[50] databases. Then the inferred protein is matched to the corresponding stoichiometric reactions using KEGG and Brenda,[51] and after checking this reaction list against the KEGG reference pathway to distinguish isozymes and het-

eromeric enzymes, the pathway model is generated in SBML. A list of 90 bacterial metabolic pathway models generated by GEM System is available at http://www.g-language.org/gem/models/static.cgi with models in SBML and stoichiometric matrix, reaction and gene lists, and pathway diagrams visualizing the components and their interactions in the generated models. All models retain high accuracy compared with KEGG organism specific pathways, achieving over 90% coverage in most bacteria, and 100% coverage in *E. coli* with 1195 metabolites and 968 reactions.

Conclusions and Outlook

The advent of high-throughput measurement technologies and the introduction of omics datasets compensated the existing paradigm of molecular biology, which tended to be qualitative and segmented, to be able to quantitatively capture a comprehensive snapshot of cell-wide activity. Phenotypes of the whole cell resulting from the nonlinear interactions of the components are predominantly quantitative, and factors contributing to the phenotypes distributed among the system[52] are also principally quantitative. Moreover, quantitative reasoning extends the dimension of research in the vector of time-course progression, allowing predictive approaches and deductive simulation experiments *in silico*. Availability of comprehensive datasets also demands for holistic approaches in order to understand how the numerous components contribute to the characteristics of the system. In this respect, a systems biology approach is invaluable for omics research, and a computational framework should be generated to support the analysis throughout the necessary steps including the database I/O, preprocessing and filtering of the data and methods, data integration and mining, and computational modeling and simulation. Although the targeted layer of omics is currently limited by mainly focusing on the genome and metabolic domains, the G-language suite already encompasses most of the required processes. The generic workbench G-language GAE provides a flexible framework for bioinformatics, and methods developed upon this workbench can be benchmarked for sensitivity with GPAC. Multi-omics data sets can be visualized with the pathway mapping tool in order to understand the data within the context of cellular activities, and the GEM System coupled with the E-Cell enables computational modeling and simulation from omics information. All software systems are designed with flexible architecture and the open-source development style of the software system will allow rapid implementation for other layers of omics in the future development.

While omics definitely advances our knowledge of life sciences by enabling our observation from a macroscopic landscape view of the inner molecular components as opposed to the specific study of microscopic set of components, it should be noted that this approach primarily inherits the descriptive and hypothesis-free paradigm of traditional molecular biology, inherently emphasizing the ability to list the parts that make up a living system. Moreover, current approaches mainly target only a specific layer of omics, which should eventually shift to research that interconnects multiple layers in order to dissect how the systematic properties and design of the molecular components in the microscopic layer contribute to the characteristics that arise as the result in a more macroscopic layer. Although the integrative and constructive aspects of systems biology seem to be mainly anticipated, here an inductive approach is especially critical,[3] connecting the systematic properties identified by comprehensive data mining of microscopic and macroscopic layers at an intermediately meso-level. Hypothetical "designs" implied or deduced from this mesoscopic approach can then be tested and refined *in silico* by the simulation of mathematical models, which are then verified by laboratory experiments, enhancing the research cycle of molecular cell biology.

Acknowledgments

This research was supported in part by the Japan Society for the Promotion of Science (JSPS), and CREST of JST (Japan Science and Technology).

References

1) M. Y. Galperin: *Nucleic Acids Res.* **34**, D3–5 (2006).
2) K. Liolios, N. Tavernarakis, P. Hugenholtz and N. C. Kyrpides: *Nucleic Acids Res.* **34**, D332–334 (2006).
3) D. B. Kell and S. G. Oliver: *Bioessays* **26**, 99–105 (2004).
4) S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller and D. J. Lipman: *Nucleic Acids Res.* **25**, 3389–3402 (1997).
5) E. Birney, *et al.*: *Nucleic Acids Res.* **34**, D556–561 (2006).
6) H. Mangalam: *Brief Bioinform.* **3**, 296–302 (2002).
7) L. Stein: *Nature* **417**, 119–120 (2002).
8) J. E. Stajich, *et al.*: *Genome Res.* **12**, 1611–1618 (2002).
9) P. Rice, I. Longden and A. Bleasby: *Trends Genet.* **16**, 276–277 (2000).
10) T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat and P. Li: *Bioinformatics* **20**, 3045–3054 (2004).
11) A. Garcia Castro, S. Thoraval, L. J. Garcia and M. A. Ragan: *BMC Bioinformatics* **6**, 87 (2005).
12) H. Kitano: *Science* **295**, 1662–1664 (2002).
13) H. Kitano: *Nature* **420**, 206–210 (2002).
14) K. Arakawa, K. Mori, K. Ikeda, T. Matsuzaki, Y. Kobayashi and M. Tomita: *Bioinformatics* **19**, 305–306 (2003).
15) K. Arakawa, Y. Nakayama and M. Tomita: *In Silico Biol.* **6**, 0006 (2006).
16) K. Arakawa, H. Suzuki, K. Fujishima, K. Fujimoto, S. Ueda, M. Matsui and M. Tomita: *Genomics Proteomics Bioinformatics* **3**, 179–188 (2005).
17) K. Arakawa, N. Kono, Y. Yamada, H. Mori and M. Tomita: *In Silico Biol.* **5**, 419–423 (2005).
18) K. Arakawa, Y. Yamada, K. Shinoda, Y. Nakayama and M. Tomita: *BMC Bioinformatics* **7**, 168 (2006).
19) Y. T. Chen, H. Y. Chang, Y. C. Lai, C. C. Pan, S. F. Tsai and H. L. Peng: *Gene* **337**, 189–198 (2004).

20) H. Suzuki, R. Saito and M. Tomita: *Gene* **335**, 19–23 (2004).
21) H. Suzuki, R. Saito and M. Tomita: *FEBS Lett.* **579**, 6499–6504 (2005).
22) P. M. Sharp and W. H. Li: *Nucleic Acids Res.* **15**, 1281–1295 (1987).
23) S. Karlin and J. Mrazek: *J. Bacteriol.* **182**, 5238–5250 (2000).
24) J. L. Bennetzen and B. D. Hall: *J. Biol. Chem.* **257**, 3026–3031 (1982).
25) M. A. Freire-Picos, M. I. Gonzalez-Siso, E. Rodriguez-Belmonte, A. M. Rodriguez-Torres, E. Ramil and M. E. Cerdan: *Gene* **139**, 43–49 (1994).
26) F. Wright: *Gene* **87**, 23–29 (1990).
27) M. Sato, H. Umeki, R. Saito, A. Kanai and M. Tomita: *Bioinformatics* **19**, 1371–1380 (2003).
28) N. Yachie, K. Numata, R. Saito, A. Kanai and M. Tomita: *Gene* **10**, 171–181 (2006).
29) T. Baba, T. Ara, M. Hasegawa, Y. Takai, Y. Okumura, M. Baba, K. A. Datsenko, M. Tomita, B. L. Wanner and H. Mori: *Mol. Syst. Biol.* **2**, 2006. 0008 (2006).
30) M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki and M. Hirakawa: *Nucleic Acids Res.* **34**, D354–357 (2006).
31) U. Bhatia, K. Robison and W. Gilbert: *Science* **276**, 1724–1725 (1997).
32) S. Audic and J. M. Claverie: *Proc. Natl. Acad. Sci. USA* **95**, 10026–10031 (1998).
33) D. Frishman, A. Mironov and M. Gelfand: *Gene* **234**, 257–265 (1999).
34) I. Iliopoulos, S. Tsoka, M. A. Andrade, P. Janssen, B. Audit, A. Tramontano, A. Valencia, C. Leroy, C. Sander and C. A. Ouzounis: *Genome Biol.* **2**, interactions0001 (2001).
35) S. E. Brenner: *Trends Genet.* **15**, 132–133 (1999).
36) R. L. Tatusov, D. A. Natale, I. V. Garkavtsev, T. A. Tatusova, U. T. Shankavaram, B. S. Rao, B. Kiryutin, M. Y. Galperin, N. D. Fedorova and E. V. Koonin: *Nucleic Acids Res.* **29**, 22–28 (2001).
37) E. Camon, M. Magrane, D. Barrell, V. Lee, E. Dimmer, J. Maslen, D. Binns, N. Harte, R. Lopez and R. Apweiler: *Nucleic Acids Res.* **32**, D262–266 (2004).
38) M. A. Harris, *et al.*: *Nucleic Acids Res.* **32**, D258–261 (2004).
39) P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski and T. Ideker: *Genome Res.* **13**, 2498–2504 (2003).
40) H. J. Chung, C. H. Park, M. R. Han, S. Lee, J. H. Ohn, J. Kim, J. Kim and J. H. Kim: *Nucleic Acids Res.* **33**, W621–626 (2005).
41) M. Holford, N. Li, P. Nadkarni and H. Zhao: *Bioinformatics* **21**, 1596–1602 (2005).
42) K. D. Dahlquist, N. Salomonis, K. Vranizan, S. C. Lawlor and B. R. Conklin: *Nat. Genet.* **31**, 19–20 (2002).
43) P. D. Karp, C. A. Ouzounis, C. Moore-Kochlacs, L. Goldovsky, P. Kaipa, D. Ahren, S. Tsoka, N. Darzentas, V. Kunin and N. Lopez-Bigas: *Nucleic Acids Res.* **33**, 6083–6089 (2005).
44) T. Oshima, H. Aiba, Y. Masuda, S. Kanaya, M. Sugiura, B. L. Wanner, H. Mori and T. Mizuno: *Mol. Microbiol.* **46**, 281–291 (2002).
45) M. Hucka, *et al.*: *Bioinformatics* **19**, 524–531 (2003).
46) K. Takahashi, K. Yugi, K. Hashimoto, Y. Yamada, C. J. F. Pickett and M. Tomita: *IEEE Intelligent Systems* **17**, 64–71 (2002).
47) M. Tomita: *Trends Biotechnol.* **19**, 205–210 (2001).
48) M. Tomita, K. Hashimoto, K. Takahashi, T. S. Shimizu, Y. Matsuzaki, F. Miyoshi, K. Saito, S. Tanida, K. Yugi, J. C. Venter and C. A. Hutchison, 3rd: *Bioinformatics* **15**, 72–84 (1999).
49) C. H. Wu, R. Apweiler, A. Bairoch, D. A. Natale, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, R. Mazumder, C. O'Donovan, N. Redaschi and B. Suzek: *Nucleic Acids Res.* **34**, D187–191 (2006).
50) R. Overbeek, N. Larsen, G. D. Pusch, M. D'Souza, E. Selkov, Jr., N. Kyrpides, M. Fonstein, N. Maltsev and E. Selkov: *Nucleic Acids Res.* **28**, 123–125 (2000).
51) I. Schomburg, A. Chang, O. Hofmann, C. Ebeling, F. Ehrentreich and D. Schomburg: *Trends Biochem. Sci.* **27**, 54–56 (2002).
52) A. Wagner: *Bioessays* **27**, 176–188 (2005).