To appear in Proceedings of the Fourth International Workshop on Cooperative Information Agents, Lecture Notes in AI, Springer-Verlag, 2000.

Applying Agents to Bioinformatics in GeneWeaver

K. Bryson[1], M. Luck[1], M. Joy[1], and D.T. Jones[2]

[1] Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK
{bryson, mikeluck, M.S.Joy}@dcs.warwick.ac.uk
http://www.dcs.warwick.ac.uk/geneweaver/
[2] Department of Biological Sciences, Brunel University, Uxbridge, UB8 3PH, UK

Abstract. Recent years have seen dramatic and sustained growth in the amount of genomic data being generated, including in late 1999 the first complete sequence of a human chromosome. The challenge now faced by biological scientists is to make sense of this vast amount of accumulated and accumulating data. Fortunately, numerous databases are provided as resources containing relevant data, and there are similarly many available programs that analyse this data and attempt to understand it. However, the key problem in analysing this genomic data is how to integrate the software and primary databases in a flexible and robust way. The wide range of available programs conform to very different input, output and processing requirements, typically with little consideration given to issues of integration, and in many cases with only token efforts made in the direction of usability. In this paper, we introduce the problem domain and describe GeneWeaver, a multi-agent system for genome analysis. We explain the suitability of the information agent paradigm to the problem domain, focus on the problem of incorporating different existing analysis tools, and describe progress to date.

1 Introduction

One of the most important and pressing challenges faced by present-day biological scientists is to move beyond the task of genomic data collection in the sequencing of DNA, and to make sense of that data so that it may be used, for example, in the development of therapies to address critical genetic disorders. The raw data has been accumulating at an unprecedented pace, and a range of computational tools and techniques have been developed by bioinformaticians, targetted at the problems of storing and analysing that data. In this sense much has already been achieved, but these tools usually require expert manual direction and control, imposing huge restrictions on the rate of progress. Essentially, however, the problems involved are familiar from other domains — vast amounts of data and information, existing programs and databases, complex interactions, distributed control — pointing strongly to the adoption of a multi-agent approach.

In this paper, we describe the development of a multi-agent system that is being applied to the very real and demanding problems of genome analysis and protein structure prediction. The vast quantities of data being rapidly generated by various sequencing efforts, the global distribution of available but remote databases that are continually updated, the existence of numerous analysis programs to be applied to sequence data in pursuit of determining gene structure and function, all point to the suitability of an agent-based approach. We begin with an introduction to the problem domain, outlining some basic biology, and explaining how it leads to the current situation in which systems such as the one we describe here are vital. Then we introduce the GeneWeaver agent community, a multi-agent systems for just this task, describing the agents involved, and the agent architecture. In all this, we aim to provide a view of the overall problem and outline all aspects of the system, rather than addressing any particular aspect, such as the databases providing the data, for example.

2 Genome Analysis and Protein Structure Prediction

The complete sequencing of the first human chromosome represents a significant milestone in the Human Genome Project, a 15-year effort to sequence the 3-billion DNA basepairs present in the human genome and to locate the estimated 100,000 or more human genes. A "rough draft" of the complete human genome will be available by spring 2000 [7], a year ahead of schedule. This draft will provide details of about 90% of the human genome with remaining gaps being filled in over the subsequent three years.

One of the key problems that such a vast amount of data presents is that of recognising the position, function and regulation of different genes. By performing various analysis techniques, such as locating similar (or *homologous*) genes in other species, these problems may be solved. Basic methods are now well developed including those for database searching to reveal similar sequences (similarity searching), comparing sequences (sequence alignment) and detection of patterns in sequences (motif searching). The key problem in the analysis of genome data is to integrate this software and primary sequence databases in a flexible and robust way. It has been stated in the biological community that there is a critical need for a "widely accepted, robust and continuously updated suite of sequence analysis methods integrated into a coherent and efficient prediction system" [6].

2.1 Function and Structure Determination

As indicated above, the rate at which this primary genetic data is being produced at present, is extremely rapid, and increasing. There is consequently a huge amount of information that is freely available across the Internet, typically stored in flat file databases. The problem of working out what each gene does, especially in light of the potential benefits of doing so, is therefore correspondingly pressing, and one which is meriting much attention from various scientific communities.

At present, the process of identifying genes and predicting the structure of the encoded proteins is fairly labour-intensive, made worse by requiring some expert knowledge. However, the steps involved in this process are all computer-based tasks: scanning sequence databases for similar sequences, collecting the matching sequences, constructing alignments of the sequences, and trying to infer the function of the sequence from annotations of the matched proteins (for which the function is already known). Predicting the three-dimensional structure of the proteins requires analyses of the collected sequence data by a range of different programs, the results of which sometimes disagree

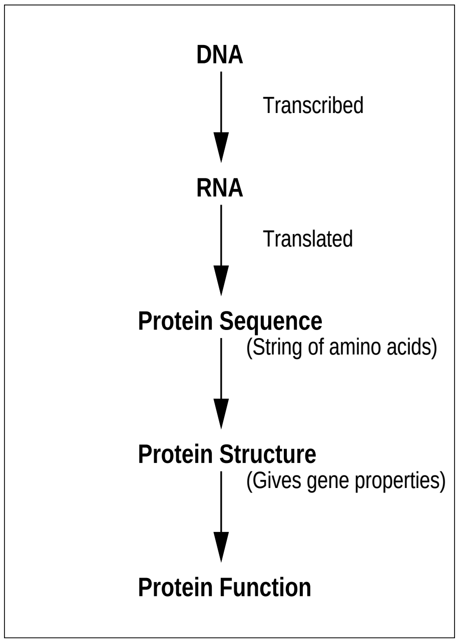

Fig. 1. From DNA to protein function

and some form of resolution needs to occur. The stages in getting to protein function from DNA are shown in Figure 1.

Now, like the primary data, some of the programs are accessible only over the Internet — either by electronic mail or the WWW (and increasingly the latter). This requires the different sources of information and the different programs to be managed effectively. Many tools are available to perform these tasks, but they are typically standalone programs that are not integrated with each other and require expert users to perform each stage manually and combine them in appropriate ways. For example, the process of trying to find a matching sequence might result in turning up an annotated gene, but the annotations include a lot of spurious information as well as the important functional information. The problem here is distilling this relevant information, which is not at all difficult for an expert, but which might prove problematic for a less experienced user. With the amount of data that is being generated, this kind of expertise is critical. For example, a very small entry in the SWISSPROT database is shown in Figure 2, in which the various lines are of greater or lesser significance. This entry is for the baboon equivalent of the protein which causes mad cow disease and other neurodegenerative disorders.

2.2 Sequence Databases

Although the principle of combining different methods and different information sources seems simple, in practice there are a number of difficulties that must be faced. Each primary data source (the primary sequence data banks and the structure data banks, for example) encode their information in different ways, not only in terms of the basic formatting, but also in the terminology used. For example, different keyword sets are used for each data bank so that understanding and interpreting the data (both of primary data sources and the results from analysis programs) is not a trivial task.

Primary databases are provided externally, under the control of a third-party who may change them at any time. In many cases, the data is largely unstructured, and is often available in the form of flat files. The methods of delivery vary, but have included email transfer of data, FTP downloads and, more recently, retrieval through the WWW. Each primary database consists of a number of sequence files, which are text files formatted in a particular sequence format. A sequence file usually contains data for a number of protein sequence entries, with each such sequence entry providing a description of the protein and its amino acid sequence.

For example, the Protein Data Bank (PDB) [5], which was established in 1971, is an international repository for the processing and distribution of experimentally-determined structure data that is growing dramatically. Similarly, SWISSPROT is a curated protein sequence databases with a high level of annotation of protein function [3]. Many other such sequence databases are also available (eg. the Protein Information Resource (PIR) [4]), but we will not provide an exhaustive list here.

2.3 Analysis Tools

The tools used to make sense of this partially structured and globally distributed genomic data apply particular techniques to try to identify the structure or function of specific sequences. A variety of such techniques are used in a range of available programs, including similarity searches of greater or lesser sensitivity (eg. BLAST [1]), sequence alignment (eg. CLUSTALW [18]), motif searching (eg. PROSITE [10]), secondary structure prediction (eg. PSIPRED [12]), fold recognition (eg. GENTHREADER [13]), and so on. For example, BLAST (Basic Local Alignment Search Tool) is a set of rapid similarity search programs that explore all available sequence databases [1]. The search can be performed by a remote server through a web interface resulting in graphical output, or alternatively can be carried out locally through a command line interface resulting in text output.

Now, some of the methods take longer to run than others, while some provide more accurate or *confident* results than others. In consequence, it is often necessary to use more than one of these tools depending on the results at any particular stage. While a high confidence is desirable, time-consuming methods should be avoided if their results are not needed.

Although these tools already exist, they are largely independent of each other. Encapsulating them as *calculation* agents provides a way to integrate their operation in support of appropriate combinations of methods to generate confident results, and integrate them with, and apply them to, the accumulating sequence databases. Each relevant tool can be encapsulated in an agent wrapper so that applications such as BLAST can become independent agents in the GeneWeaver community.

```
ID   PRIO_THEGE     STANDARD;      PRT;   238 AA.
AC   Q95270;
DT   01-NOV-1997 (Rel. 35, Created)
DT   01-NOV-1997 (Rel. 35, Last sequence update)
DT   01-NOV-1997 (Rel. 35, Last annotation update)
DE   MAJOR PRION PROTEIN PRECURSOR (PRP) (PRP27-30) (PRP33-35C) (FRAGMENT).
GN   PRNP OR PRP.
OS   Theropithecus gelada (Gelada baboon).
OC   Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Mammalia;
OC   Eutheria; Primates; Catarrhini; Cercopithecidae; Cercopithecinae;
OC   Theropithecus.
RN   [1]
RP   SEQUENCE FROM N.A.
RA   DER KUYL A.C., DEKKER J.T., GOUDSMIT J.;
RL   Submitted (NOV-1996) to the EMBL/GenBank/DDBJ databases.
CC   -!- FUNCTION: THE FUNCTION OF PRP IS NOT KNOWN. PRP IS ENCODED IN THE
CC       HOST GENOME AND IS EXPRESSED BOTH IN NORMAL AND INFECTED CELLS.
CC   -!- SUBUNIT: PRP HAS A TENDENCY TO AGGREGATE YIELDING POLYMERS CALLED
CC       "RODS".
CC   -!- SUBCELLULAR LOCATION: ATTACHED TO THE MEMBRANE BY A GPI-ANCHOR.
CC   -!- DISEASE: PRP IS FOUND IN HIGH QUANTITY IN THE BRAIN OF HUMANS AND
CC       ANIMALS INFECTED WITH THE DEGENERATIVE NEUROLOGICAL DISEASES KURU,
CC       CREUTZFELDT-JAKOB DISEASE (CJD), GERSTMANN-STRAUSSLER SYNDROME
CC       (GSS), SCRAPIE, BOVINE SPONGIFORM ENCEPHALOPATHY (BSE),
CC       TRANSMISSIBLE MINK ENCEPHALOPATHY (TME), ETC.
CC   -!- SIMILARITY: BELONGS TO THE PRION FAMILY.
DR   EMBL; U75383; AAB50630.1; -.
DR   HSSP; P04925; 1AG2.
DR   PROSITE; PS00291; PRION_1; 1.
DR   PROSITE; PS00706; PRION_2; 1.
DR   PFAM; PF00377; prion; 1.
KW   Prion; Brain; Glycoprotein; GPI-anchor; Repeat; Signal.
FT   NON_TER       1      1
FT   SIGNAL       <1     15       BY SIMILARITY.
FT   CHAIN        16   >238       MAJOR PRION PROTEIN.
FT   DISULFID    164    199       BY SIMILARITY.
FT   CARBOHYD    166    166       POTENTIAL.
FT   CARBOHYD    182    182       POTENTIAL.
FT   DOMAIN       44     83       4 X 8 AA TANDEM REPEATS OF P-H-G-G-G-W-G-
FT                                Q.
FT   REPEAT       44     52       1.
FT   REPEAT       53     60       2.
FT   REPEAT       61     68       3.
FT   REPEAT       69     76       4.
FT   NON_TER     238    238
SQ   SEQUENCE   238 AA;  26104 MW;  3E0A3951 CRC32;
     MLVLFVATWS DLGLCKKRPK PGGWNTGGSR YPGQGSPGGN RYPPQGGGGW GQPHGGGWGQ
     PHGGGWGQPH GGGWGQGGGT HNQWHKPSKP KTSMKHMAGA AAAGAVVGGL GGYMLGSAMS
     RPLIHFGNDY EDRYYRENMY RYPNQVYYRP VDQYSNQNNF VHDCVNITIK QHTVTTTTKG
     ENFTETDVKM MERVVEQMCI TQYQKESQAY YQRGSSIVLF SSPPVILLIS FLIFLIVG
//
```

Fig. 2. An example SWISSPROT database entry

3 The GeneWeaver Agent Community

GeneWeaver is a multi-agent system aimed at addressing many of the problems in the domain of genome analysis and protein structure prediction, as discussed above. It comprises a community of agents that interact with each other, each performing some distinct task, in an effort to automate the processes involved in, for example, determining gene function. Agents in the system can be concerned with management of the primary databases, performing sequence analyses using existing tools, or with storing and presenting resulting information. The important point to note is that the system does not offer new methods for performing these tasks, but organises existing ones for the most effective and flexible operation. This section provides an overview of the system through the agents within it.

Figure 3 illustrates the overall perspective of GeneWeaver in that it contains the different classes of agents and shows how they inter-relate. At the left side, PDB Agent, Swiss Agent and PIR Agent all manage the primary sequence databases indicated by their names, and interact with the Protein NRDB (non-redundant database) Agent, which combines their data. At the right edge of the figure, the calculation agents (including the Blast Agent and the Clustal Agent that perform specific analysis tasks) attempt to annotate sequences in the database using relevant programs, again indicated by their names. At the top right, an expert calculation agent can combine the skills of the other calculation agents using expert knowledge encoded in plans. In this case it can use the Blast Agent to find similar proteins and then use the Clustal Agent to accurately compare the proteins obtained. (For clarity, these interactions are not shown in the diagram). Finally, at the top, the results generated by the system must be externally accessible, and this functionality is achieved by the Genome Agent. At each point of external interaction, agents typically receive and provide information via the WWW.

There are five types of agent present in the GeneWeaver community.

– *Broker agents*, which are not shown in Figure 3 since they are *facilitators* rather than points of functionality, are needed to register information about other agents in the community. They are similar in spirit to the notions discussed, for example, by Foss [9] and Wiederhold [20], but are very limited in functionality because of the constrained domain. (With a more sophisticated domain, however, this functionality might be correspondingly enhanced, to include more complex matchmaking, for example [14].)
– *Primary database agents* are needed to manage remote primary sequence databases, and keep the data contained in them up-to-date and in a format that allows other agents to query that data.
– *Non-redundant database agents* construct and maintain non-redundant databases from the data managed by other primary database agents in the community.
– *Calculation agents* encapsulate some pre-existing methods or tools for analysis of sequence data. They attempt to determine the structure or function of a protein sequence. Some calculation agents have domain-specific expert knowledge encoded as plans that enable them to carry out expert tasks using the other calculation agents.
– *Genome agents* are responsible for managing the genomic information for a particular organism.

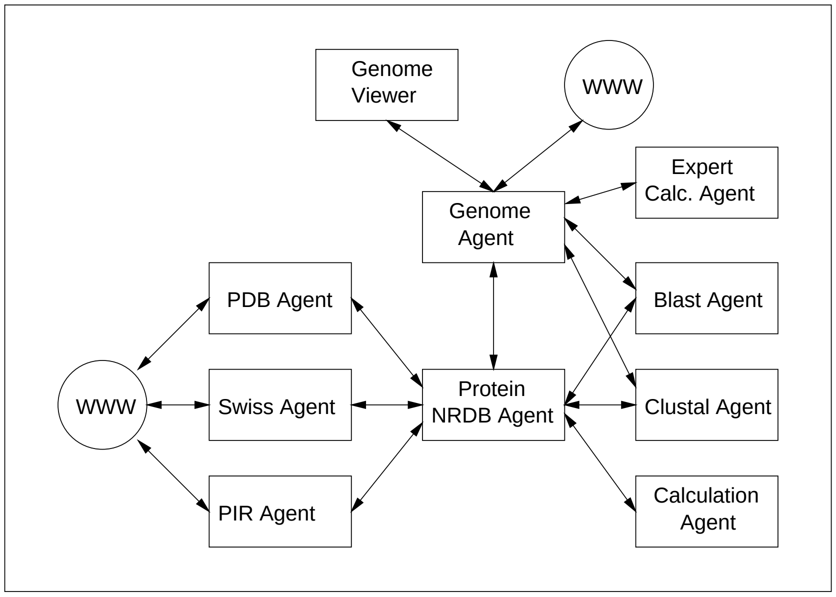

Fig. 3. GeneWeaver agent community

4 Architecture

Each agent in the GeneWeaver community shares a common architecture that is inspired by, and draws on, a number of existing agent architectures, such as [11], but in a far more limited and simplified way. An agent contains a number of internal modules together with either an external persistent data store that is used for the storage of data it manipulates, or the analysis program used to predict function. In this section, we describe the generic modules that comprise the architecture illustrated by Figure 4, which forms the basis of each agent. In essence, everything revolves around the central *control module*, which is given direction by the *motivation module* through particular goals, and then decides how best to achieve those goals. It can either take action itself through its *action module*, or request assistance from another agent through its *interaction module* which, in turn, uses the *communications module* for the mechanics of the interaction. The *meta-store* simply provides a repository for local information such as the skills of other agents. Each of the modules making up the architecture is considered in detail below.

Neither the *data store* nor the *analysis tools* are regarded as a part of the agent but they are used by agents either to store persistent data they are working with or to analyse the data. Various different types of data (such as a protein sequence, a sequence file, etc.) exist within the GeneWeaver system, and an interface to the data store allows data to be added, deleted, replaced, updated and queried. In addition, although we might envisage an agent wrapper around an analysis tool in a slightly different visual representation, the calculation agents interface with these existing programs in a similar fashion. In

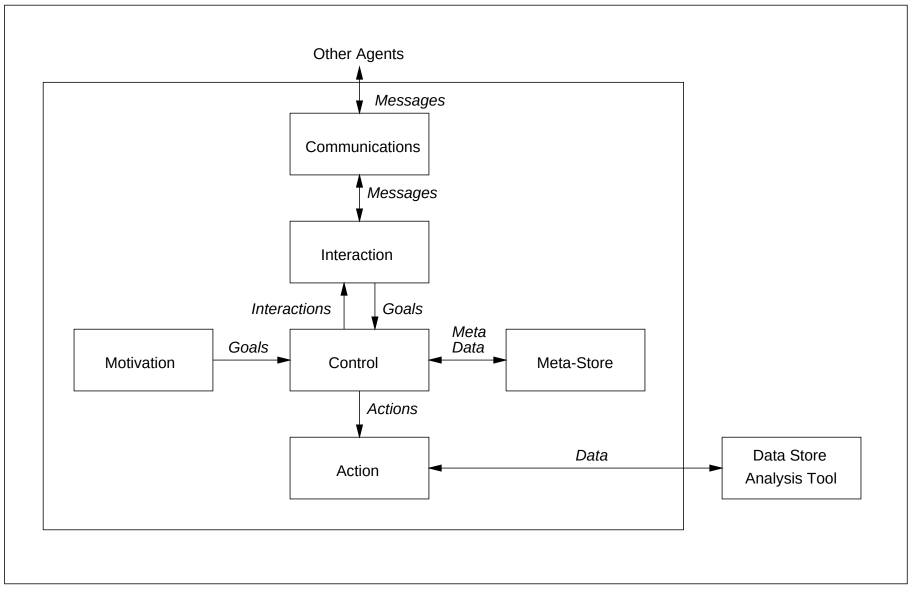

Fig. 4. GeneWeaver agent architecture

what follows, we focus on analysis rather than data management, which is considered in more detail elsewhere.

4.1 Motivation Module

Since each agent in the GeneWeaver community has different responsibilities and is required to perform different tasks, distinct high-level direction must be provided in each case to cause essentially the same architecture to function in different ways. The conceptual organisation of GeneWeaver agents thus involves the use of some high-level motivations that cause the goals and actions specific to the agent's tasks and responsibilities to be generated and performed [15, 16].

An agent is therefore initialised with the motivations required to carry out its responsibilities effectively. For example, all agents (except brokers) are motivated to register themselves with a broker agent, and will generate goals and actions to do so until they have succeeded. Similarly, a primary database agent has a motivation to cause the generation of specific goals and actions to update the primary database on a regular basis in order to ensure that it is up-to-date.

The distinction between motivations and goals is that motivations have associated intensities that cause goals to be generated. Each motivation is assigned a default intensity level, which can be modified by the *control module*, with only motivations with the highest intensity firing. For instance, the motivation to register with a broker initially defaults to a maximum intensity, but once the action has been achieved and the goal satisfied, the intensity decreases to zero. This not only allows the different motivations to be ordered in terms of priorities, it also allows intensities to be modified during execution.

4.2 Control Module

Perhaps the most important of the components of the agent architecture is the *control module*, which organises how actions should be carried out, once provided with a goal to pursue. The control module is initiated with a number of *plans* which can satisfy different goals and which consist of a number of steps specifying types of actions or interactions to be performed. It uses the meta-store for information on how particular actions should be carried out by determing whether suitable skills are available locally or whether the skills of a remote agent are required. The meta-store also provides information about types of interactions supported by other agents.

For example, a simple goal might be to find a match for a particular protein sequence. One way to satisfy this may consist of the single step of using a suitable *similarity search* method. Information contained in the meta-store is then used to locate just such a *similarity search* method, which may be one of the skills local to the agent or may require interaction with another agent. A more complicated way of accomplishing this goal may consist of two steps: a *similarity search* method may be used to find a similar sequence, which can then be used in a *sequence alignment* against the original sequence to provide more confidence in the result. Now plans can have a *quality* assigned to them, with the latter plan in the example above having a higher quality than the simpler one. They can also have an *efficiency* value associated with them, with the latter plan having a lower efficiency due to the use of more resources. Note that some plans may only be applicable in certain circumstances; for instance the latter plan requires the meta-store to identify a suitable *sequence alignment* method to use. Thus the plans have pre-conditions which need to be satisfied before they can be used. So, a suitable plan to meet a particular goal can be selected based on its quality, efficiency and any pre-conditions it requires.

4.3 Action Module

The *action module* is responsible for managing and performing *data actions* that modify the underlying data being manipulated by the agent. Such actions typically involve performing operations on some input data and may (optionally) result in some output. In this respect, they are likely to use and modify data in the agent's data store. The action module is thus critically important in terms of the agent functionality in this domain, since it is the only module that can interface to the underlying wrapped analysis program, and thus provides the only way to invoke the tool for performing an analysis. Each agent's action module is instantiated with a number of *skills* (or types of action) that the agent can perform.

4.4 Interaction Module

The *interaction module* handles the higher level interaction between different agents. Several possible types of interaction exist, each of them following a particular fixed interaction protocol.

An interaction takes place between two agents: a *requester* and a *provider*. When assistance is needed by an agent, a *requester interaction* is generated by its *interaction module* and a message sent to the provider via the *communications module*. At the provider's end of the interaction, the receipt of a request for assistance (via its communications module) causes a *provider interaction* suitable for the type of interaction to be invoked in the *interaction module*. The two agents then communicate using a fixed protocol, with the respective agents adopting suitable goals required on their own side of the interaction.

The interaction module is initialised with the types of interaction it can service (or the types of provider interactions it can initiate). For example, the broker is the only agent that initiates its interaction module with a *register* interaction to service requests to *register* with it from other agents. The particular interactions an agent is able to provide to others are recorded in its meta store so that it has an accurate picture of its capabilities.

4.5 Communications Module

The mechanics of the interaction of agents in the community is achieved through message-passing communication, which is handled by the communications module. An agent communication protocol specifies both the *transport protocol* to be used (which can be one of several, including RMI and CORBA [19, 8], for example) and the communication *language* (which is currently limited to a small prototype KQML-based [17] language). In principle, each agent may have available to it multiple protocols to use in interacting with others. This is achieved by instantiating the communications module at initiation with all those protocols known to the agent (which are also recorded in the meta store so it has an accurate representation of its own abilities).

Now, the communications module for one agent interacts with communications modules of other agents, using particular transport protocols. It also passes messages on to the agent itself for interpretation and processing, as well as accepting outgoing messages to be sent out to others. In this way, one agent interacts with another through their respective communications modules.

4.6 Meta Store

The *meta-store* simply provides a repository for the information that is required by an individual agent for correct and efficient functioning. For example, the meta-data contained in this repository will enumerate the properties and capabilities of the agent, including aspects such as the protocols the agent can use, the skills that can be executed, and the agent's motivations. As other modules are instantiated on initialisation, this information is added to the *meta-store*. Thus, as the *action module* is initialised, the skills that can be performed by it are added to the repository.

The *meta-store* also provides a representation of the other agents in the community in order to determine how best to accomplish particular tasks, possibly using other agents. Information contained in it may be extended while the agent is running so that additional or newly-discovered information about itself or other agents may be included. The only significant interaction is with the *control module* which records information in the *meta-store* as appropriate, and also uses it in decision-making.

5 Conclusions

The problems faced by biological scientists in relation to the increasing amounts of genomic data being generated are becoming critical. Autonomous genome analysis that avoids the need for extensive input by domain experts, but succeeds in annotating the data with structure and function information is a key goal. Through the use of a multi-agent system in which existing databases and tools are encapsulated as independent agents, we aim to relieve the expert of this burden, and increase the throughput of genomic data analysis. In this way, we can actually use the data that has been recorded.

A similar effort on the GeneQuiz system, which generates preliminary functional annotation of protein sequences, has been applied to the analysis of sets of sequences from complete genomes, both to refine overall performance and to make new discoveries comparable to those made by human experts [2]. Though GeneQuiz has a similar motivation, and makes use of various external databases and analysis tools, its structure suggests that significant modifications may be necessary with the introduction of new databases or tools. In contrast, the agent approach taken in GeneWeaver, in which each agent is autonomous and distinct from the rest of the system, means that the community of agents can grow in line with the development of new databases and tools without adversely affecting the existing organisation.

In the GeneWeaver project to date, we have developed a prototype system reflecting the structure of the community and the individual agent architecture described above. Database agents for several external databases have been constructed, and a sample calculation agent to perform BLAST searches has been incorporated into the community. Current work aims to extend the range of calculation agents and then to assess the entire system in relation to activity of human domain experts, to identify refinements both to the architecture and the individual agent control mechanisms.

Acknowledgements

The work described in this paper is supported by funding from the Bioinformatics programme of the UK's EPSRC and BBSRC.

References

1. S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lippman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.
2. M.A. Andrade, N.P. Brown, C. Leroy, S. Hoersch, A. de Daruvar, C. Reich, A. Franchini, J. Tamames, A. Valencia, C. Ouzounis, and C. Sander. Automated genome sequence analysis and annotation. *Bioinformatics*, 15(5):391–412, 1999.
3. A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1999. *Nucleic Acids Research*, 27(1):49–54, 1999.

4. W.C. Barker, J.S. Garavelliand P.B. McGarvey, C.R. Marzec, B.C. Orcutt, G.Y. Srinivasarao, L.-S.L. Yeh, R.S. Ledley, H.-W. Mewes, F. Pfeiffer, A. Tsugita, and C. Wu. The PIR-international protein sequence database. *Nucleic Acids Research*, 27(1):39–43, 1999.
5. F.C. Bernstein, T.F. Koetzle, G.J. Williams, E.E. Meyer Jr, M.D. Brice, J.R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank: a computer-based archival file for macromolecular structures. *Journal of Molecular Biology*, 112, 1977.
6. P. Bork and E.V. Koonin. Predicting functions from protein sequences: where are the bottlenecks? *Nature Genetics*, 18:313–318, 1998.
7. F.S. Collins, A. Patrinos, E. Jordan, A. Chakravarti, R. Gesteland, L. Walters, and the members of the DOE and NIH planning groups. New goals for the U.S. human genome project: 1998-2003. *Science*, 282:682–689, 1998.
8. J. Farley. *Java Distributed Computing*. O'Reilly, 1998.
9. J.D. Foss. Brokering the info-underworld. In N.R. Jennings and M. J. Wooldridge, editors, *Agent Technology: Foundations, Applications, and Markets*, pages 105–123. Springer-Verlag, 1998.
10. K. Hofmann, P. Bucher, L. Falquet, and A. Bairoch. The PROSITE database, its status in 1999. *Nucleic Acids Research*, 27(1):215–219, 1999.
11. N.R. Jennings and T. Wittig. ARCHON: Theory and practice. In *Distributed Artificial Intelligence: Theory and Praxis*, pages 179–195. ECSC, EEC, EAEC, 1992.
12. D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292:195–202, 1999.
13. D.T. Jones. GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences. *Journal of Molecular Biology*, 287:797–815, 1999.
14. D. Kuokka and L. Harada. Matchmaking for information agents. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 672–679, 1995.
15. M. Luck and M. d'Inverno. Engagement and cooperation in motivated agent modelling. In C. Zhang and D. Lukose, editors, *Distributed Artificial Intelligence Architecture and Modelling: Proceedings of the First Australian Workshop on Distributed Artificial Intelligence, Lecture Notes in Artificial Intelligence, 1087*, pages 70–84. Springer Verlag, 1996.
16. M. Luck and M. d'Inverno. Motivated behaviour for goal adoption. In *Multi-Agent Systems Theories Languages and Applications: Proceedings of the Fourth Australian Workshop on Distributed Artificial Intelligence, Lecture Notes in Artificial Intelligence, 1544*, pages 58–73. Springer Verlag, 1998.
17. J. Mayfield, Y. Labrou, and T. Finin. Evaluating KQML as an agent communication language. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents II (LNAI 1037)*, pages 347–360. Springer, 1996.
18. J.D. Thompson, D.G. Higgins, and T.J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.
19. G. Vossen. The CORBA specification for cooperation in heterogeneous information systems. In P. Kandzia and M. Klusch, editors, *Cooperative Information Agents*, pages 101–115. Springer-Verlag, 1997.
20. G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3), 1992.