Peer-to-Peer Experimentation in Protein Structure Prediction: an Architecture, Experiment and Initial Results

Xueping Quan¹, Chris Walton², Dietlind L. Gerloff¹, Joanna L. Sharman¹, and Dave Robertson²*

¹ Institute of Structural and Molecular Biology, University of Edinburgh, UK
² School of Informatics, University of Edinburgh, UK

Abstract. Peer-to-peer approaches offer some direct solutions to modularity and scaling properties in large scale distributed systems but their role in supporting precise experimental analysis in bioinformatics has not been explored closely in practical settings. We describe a method by which precision in experimental process can be maintained within a peer-to-peer architecture and show how this can support experiments. As an example we show how our system is used to analyse real data of relevance to the structural bioinformatics community. Comparative models of yeast protein structures from three individual resources were analysed for consistency between them. We created a new resource containing only model fragments supported by agreement between the methods. Resources of this kind provide small sets of likely accurate predictions for non-expert users and are of interest in applied bioinformatics research.

1 Introduction: Peer-to-Peer Experimentation

In Section 4 we describe novel results obtained for a specific experiment that concerns consistency in protein structure prediction. When read by itself, Section 4 is a novel piece of analysis with a result of interest to part of the bioinformatics community. The broader novelty of this paper, however, is the way in which this result is obtained and in particular the peer-to-peer architecture used to obtain it. A peer-to-peer architecture is one in which computation is distributed across processors and in which none of the processors has an overarching coordinating role - hence coordination for a given task must be achieved via communication between processors. Section 1.1 gives our perspective on scientific experimentation as a peer-to-peer activity. Section 1.2 summarises the way in which we tackle the crucial issue of maintaining the integrity of experiments in a peer-to-peer setting. With this general approach in place, we then describe (in Section 2) its use to implement the specific experiment of Section 4. Section 3 summarises the general-purpose program used to conduct our experiments, and Section 5 concludes by summarising the broader system of which this is the first part.

* The research described in this paper was funded through a grant from the European Union Sixth Framework Programme, Information Society Technologies, sponsoring the Open Knowledge Project (www.openk.org).
1.1 Scientists as Peers in a Web Community

When conducting experimental studies (or amassing information to support experimental studies) from Internet sources, each scientist (or group) may adopt a variety of roles as information providers, consumers or modifiers. Often these roles are narrowly specific, as for example the role one adopts when canvassing trusted sources for information about specific proteins and applying assessment metrics to these that are appropriate to a particular style of experimentation. In science, the roles we adopt and the specific ways in which we discharge the obligations of these roles are fundamental to establishing peer groups of “like minded” scientists in pursuit of related goals by compatible means.

The need to be precise about such obligations is strongly felt in traditional science - hence the use of rigid conventions for description of experimental method and monitoring of its execution via laboratory notebooks, enabling experiments to be monitored, replicated and re-used. Analogous structure is beginning to emerge in Internet based science. For example the structure of Web service composition in Taverna [1] provides a record of the associations between services when using these to manipulate scientific data. Like Taverna, we describe interactions as process models. Unlike Taverna, our process models are part of a system for peer-to-peer communication in which process models describing complementary roles in experimentation are shared between peers as a means of communicating and coordinating experiments.

1.2 Scientific Coordination as Peer-to-Peer Communication

Traditionally, peer-to-peer systems have not focused on the issue of maintaining the integrity of complex, flexible processes that span groups of interacting peers. Engineering solutions have polarised into those which are highly centralised (co-ordinating interactions through a server) versus those which rely entirely on the sophistication (and coordinated engineering) of peers to obtain reliable processes through emergent behaviours. It is, however, possible to have a distributed, de-centralised, interaction guided by a shared, mobile model of interaction. To support this we have developed a specification language, based on a process calculus, that can describe interactions between peers and (since the language is executable in the tradition of declarative programming) can be deployed to control interactions. The language is called the Lightweight Coordination Calculus (LCC) in recognition of our aim to produce the most easily applied formal language for this engineering task.

The LCC Language Space limitations prohibit detailed discussion of LCC, its semantics or of the mechanisms used to deploy it. For these, the reader is referred to [2]. In this paper we explain enough of LCC to take us through the bioinformatics experiment of subsequent sections.

Broadly speaking, an interaction model (or, for scientists, an experimental protocol) in LCC is a set of clauses, each of which defines how a role in the interaction must be performed. Roles are described by the type of role and an
identifier for the individual peer undertaking that role, which we write formally as \(a(R\ell, I\text{den})\). The definition of performance of a role is constructed using combinations of the sequence operator (‘\(\text{then}\)’) or choice operator (‘\(\text{or}\)’) to connect messages and changes of role. Messages are either outgoing to another peer in a given role (‘\(\Rightarrow\)’) or incoming from another peer in a given role (‘\(\Leftarrow\)’). Message input/output or change of role can be governed by a constraint defined using the normal logical operators for conjunction, disjunction and negation. Notice that there is no commitment to the system of logic through which constraints are solved - on the contrary we expect different peers to operate different constraint solvers. We explain below why we take this view in the context of peer to peer experimentation.

A conversation among a group of peers can be described as a collection of dialogue sequences between peers. The speech acts conveying information between peers are performed only by sending and receiving messages. For example, suppose a dialogue allows peer \(a(r1,a1)\) to broadcast a message \(m1\) to peers \(a(r2,a2)\) and \(a(r3,a3)\) and wait for a reply from both of these. Peer \(a(r2,a2)\) is expected to reply with message \(m2\) to peer \(a(r1,a1)\). Peer \(a(r3,a3)\) is expected to do the same. Assuming that each peer operates sequentially (an assumption that is not essential to this paper but helpful for the purpose of example) the sets of possible dialogue sequences we wish to allow for the three peers in this example are as given below, where \(M_o \Rightarrow A_r\) denotes a message, \(M_o\), sent to peer \(A_r\), and \(M_i \Leftarrow A_i\) denotes a message, \(M_i\), received from peer \(A_i\).

\[
\begin{align*}
\text{For } a(r1,a1) &: \{m1 \Rightarrow a(r2,a2), m1 \Rightarrow a(r3,a3), m2 \Leftarrow a(r2,a2), m2 \Leftarrow a(r3,a3)\} \\
\text{For } a(r2,a2) &: \{m1 \Leftarrow a(r1,a1), m2 \Rightarrow a(r1,a1)\} \\
\text{For } a(r3,a3) &: \{m1 \Leftarrow a(r1,a1), m2 \Rightarrow a(r1,a1)\}
\end{align*}
\]

We can specify this dialogue using a notation, similar in style to CCS, to describe the permitted message passing behaviour of each peer. Each peer, \(A\), is defined by a term, \(A :: D\), where \(D\) describes the messages it is allowed to send. \(D\) can be constructed using the operators: \(D_i \text{ then } D_j\) (requiring \(D_i\) to be satisfied before \(D_j\)), \(D_i \text{ or } D_j\) (requiring a choice between \(D_i\) or \(D_j\)) or \(D_i \text{ par } D_j\) (requiring the peer to wait until \(D_i\) and \(D_j\) both are satisfied). A specification using this notation for our example is the following set of clauses:

\[
\begin{align*}
\{a(r1,a1) :: (m1 \Rightarrow a(r2,a2) \text{ then } m1 \Rightarrow a(r3,a3)) \\
\text{For } a(r2,a2) &: \{m1 \Leftarrow a(r1,a1) \text{ then } m2 \Rightarrow a(r1,a1)\} \\
\text{For } a(r3,a3) &: \{m1 \Leftarrow a(r1,a1) \text{ then } m2 \Rightarrow a(r1,a1)\}
\end{align*}
\]

The syntax of LCC is as follows, where \(\text{Term}\) is a structured term and \(\text{Constant}\) is a constant symbol assumed to be unique when identifying each peer:
Framework := \{ Clause, \ldots \}

Clause := Peer :: Def

Peer := a(Type, Id)

Def := Peer \mid Message \mid Def then Def \mid Def or Def \mid Def par Def

Message := Info \leftarrow Peer \mid Info \Rightarrow Peer

Type := Term

Id := Constant

The above defines a space of possible dialogues but this is larger than the space we typically wish to allow in practice, since dialogues normally assume constraints on the circumstances under which messages are sent or received. These constraints are of two kinds: proaction constraints and reaction constraints. Proaction constraints define the circumstances under which a message allowed by the dialogue framework is allowed to be sent. LCC makes no assumption about the mechanism for deciding whether these constraints hold in the current state of the peer, this being a matter for the engineer of that peer. Each constraint is of the form:

\[ A : (M \Rightarrow A_r) \leftarrow C_p \quad (1) \]

where \( A \) and \( A_r \) are peer descriptors (of the form \( a(\text{Type}, \text{Id}) \)); \( M \) is a message sent by \( A \) addressed to \( A_r \); and \( C_p \) is the condition for sending the message (either empty or a conjunction of sub-conditions which should hold in \( A \)). In our earlier example, to constrain peer \( a(r_1, a_1) \) to send message \( m_1 \) to peer \( a(r_2, a_2) \) when condition \( c_1 \) holds in \( a(r_1, a_1) \) we would define the proaction constraint: \( a(r_1, a_1) : (m_1 \Rightarrow a(r_2, a_2)) \leftarrow c_1 \).

Reaction constraints define what should be true in an peer following receipt of a message allowed by the dialogue framework. As for proaction constraints, LCC makes no assumption about the mechanism for ensuring that these constraints hold in the current state of the peer. Each constraint is of the form:

\[ A : (M \Leftarrow A_s) \rightarrow C_r \quad (2) \]

where \( A \) and \( A_s \) are peer descriptors (of the form \( a(\text{Type}, \text{Id}) \)); \( M \) is a message sent by \( A \) and received by \( A_s \); and \( C_r \) is the reaction upon receiving the message (either null or a conjunction of sub-conditions which \( A \) should be able to satisfy). In our earlier example, to constrain peer \( a(r_2, a_2) \) to receive message \( m_1 \) from peer \( a(r_1, a_1) \) when condition \( c_2 \) holds in \( a(r_2, a_2) \) we would define the proaction constraint: \( a(r_2, a_2) : (m_1 \Leftarrow a(r_1, a_1)) \rightarrow c_2 \).

As a LCC specification is shared among a group of peers it is essential that each peer when presented with a message from that specification can retrieve the state of the dialogue relevant to it and to that message. This is done by retaining (separately) in the specification the instances of dialogue clauses used by each peer participating in the dialogue. The principle of this is similar to unfolding in the transformation of logic programs, where we can take a clause; find a sub-goal
satisfiable by other clauses; then replace this with subgoal with the subgoals of that clause. A detailed description from a logic programming viewpoint appears in [2].

Using LCC as an Experiment Protocol Language The experiment described in this paper relies on the collation of predicted structures for yeast proteins across a number of peers and comparison of the collated data across the peers to produce a tentative assessment of the predictions. The data is filtered based on these comparisons leaving behind only predictions deemed to be reliable on these grounds. Figure 1 defines a LCC specification for our example. Notice that it is specific about the message sequencing and essential constraints on this type of interaction but it leaves flexible the choice and number of peers supplying data and the forms of data lookup and filtering - so the LCC specification is a model of a class of interactions, and we can ground it in specific peers and constraints at deployment time (as we show in Section 2).

Brief Comparison to Related Approaches Since LCC is a process calculus used in a peer to peer setting, it intersects with a variety of standards and methods. We summarise below what we believe to be the three main points of contact in bioinformatics experimentation:

Targeted systems: These are systems which adopt a peer to peer architecture but where the purpose of the peer groups is focused on a specific class of experimentation. An example is the SEED system [3] which is targeted on gene annotation. This sort of system gains accessibility (for biologists) because it is specific to that domain (although of course in time SEED may generalise). LCC differs from this approach by being a generic peer to peer coordination language in which one obtains specific forms of peer to peer coordination through description of the required interaction.

Peer to peer frameworks: These are computing frameworks that allow bioinformatics programmers an easy way to establish peer to peer networks. An example is the Chinook system [4] which allows bioinformaticians familiar with Perl to build peer to peer applications. LCC also requires a form of programming (since LCC specifications have a computational behaviour) but LCC does not depend on any specific peer to peer framework. Instead, it standardises on the knowledge representation language used to represent the interaction process and its current state(s). This allows the same LCC specification to be used across multiple frameworks.

Scientific workflow systems: These provide high level visual interfaces by which scientists may describe processes connecting Web/Grid services. Examples are the Taverna [1] and Kepler [5] systems. Their role is analogous to CASE tools in traditional software engineering: they make it easier for engineers to construct executable code for a class of problems. Like CASE tools, their weakness is that outside of that class of problems an engineer must understand deeply how a specification is structured. Although it is possible
A data collator, $X$, accepts a request for data on a yeast protein specified by its yeast-id, $I$, from an experimenter, $E$; then $X$ changes role to being a data collector over the set, $Sp$, of data sources about which it has knowledge; then it reports back the set, $S$, of filtered data from those sources.

A data collector takes each peer identifier, $P$, in turn from the given set, $Sp$, of data sources and changes to the role of data retriever to obtain the data set, $D$, from $P$. The set $Sd$ contains each instance of $D$ obtained by data retrieval.

A data retriever requests data on yeast-id, $I$, from data source, $P$, and then awaits the report of the data set, $D$ from that source.

A data source accepts a data request and replies with the the data set obtained via lookup of its data base.

\[
\begin{align*}
a(data\_collator, X) :: \\
  &data\_request(I) \Leftarrow a(experimenter, E) \text{ then} \\
  &a(data\_collator(I, Sp, Sd), X) \Leftarrow yeast\_id(I) \land sources(Sp) \text{ then} \\
  &data\_report(I, S) \Rightarrow a(experimenter, E) \Leftarrow filter(Sd, S)
\end{align*}
\]

\[
\begin{align*}
a(data\_collator(I, Sp, Sd), X) :: \\
  & \left( \text{null \Leftarrow Sp = \[] \land Sd = \[] \text{ or} \right) \\
  & \left( a(data\_retriever(I, P, D), X) \Leftarrow Sp = [P|Rd] \land Sd = [D|Rd] \text{ then} \right) \\
  & a(data\_collator(I, Rp, Rd), X)
\end{align*}
\]

\[
\begin{align*}
a(data\_retriever(I, P, D), X) :: \\
  &data\_request(I) \Rightarrow a(data\_source, P) \text{ then} \\
  &data\_report(I, D) \Leftarrow a(data\_source, P)
\end{align*}
\]

\[
\begin{align*}
a(data\_source, P) :: \\
  &data\_request(I) \Leftarrow a(data\_retriever(I, P, D), X) \text{ then} \\
  &data\_report(I, D) \Rightarrow a(data\_retriever(I, P, D), X) \Leftarrow lookup(I, D)
\end{align*}
\]

\textbf{Fig. 1.} Example peer-to-peer interaction specified in LCC
to invent a visual language that can be generically used for specifying logic-based programs (see [6] for an example) there is no evidence that a generic visualisation is easier for human communities of practise to understand than the mathematical representation from which it originated. LCC therefore attempts to provide the leanest possible such language, assuming that the vast majority of scientists will be users of LCC specifications rather than designers (which requires very effective methods for sharing LCC specifications - a topic of the OpenKnowledge project described at www.openk.org). In the sense that one could build a high level visual interface for a subset of the LCC language, LCC is complementary to workflow CASE tools.

2 Experiment Implementation

The LCC specification, presented in the previous section, defines how the various peers will interact during the experimental process to provide the desired outcome. This specification describes how the experiment will be performed without directly identifying the peers that will be involved. To perform the actual experiment, it is necessary to supply a set of peers that match this specification, and thereby enable us to instantiate the LCC protocol. In this section, we describe how the actual experiment was performed, and outline the computational services that we constructed to accomplish this task.

From a computational point of view, our experiment is essentially a service composition task. That is, we will construct our experiment by identifying a collection of independent services, and then compose these services together dynamically to enact our experiment. In doing so, we adhere to the popular Service Oriented Architecture (SOA) paradigm, which is commonly used in Grid computing. We use our Magenta tool to perform this dynamic service composition as it was designed specifically for this purpose, as part of the OpenKnowledge project. Magenta is effectively an interpreter for LCC specifications, where the peers are defined by Web Services. We have previously demonstrated the use of Magenta to compose services in the astronomy domain [7,8]. Nonetheless, there are a number of important differences between the experiment that we perform here, and our previous astronomy experiments. Three of the key differences are summarised below:

1. Previously, we were using web services that had already been constructed for the AstroGrid project. In this case, while the necessary data is freely available on the web, there are no web services constructed to access this data. In other words, the data is accessible using HyperText Markup Language (HTML) pages intended for humans, but there are no Web Services Description Language (WSDL) interfaces and procedures to make this data available to computation entities, e.g. peers. Therefore, it was necessary to construct our own services to query and retrieve the data through form posting and screen-scraping techniques.

2. The astronomy data was all obtained from the same source and was uniform, i.e. all data was the same format and quality. Here, we are attempting to
reconcile data from three independent sources. Each of these sources has derived their data using different methods, and have classified their data in different ways. To overcome these issues, it was necessary to build our services so that they can cope with missing and incomplete data, and can present the data in a uniform way. It was also necessary to design our services so that they could place quality thresholds on the data, and exclude results which did not meet these thresholds.

3. The final difference concerns the control of the underlying services and data. In the astronomy scenario, we were closely associated with the individuals who constructed the services and gathered the data. This meant that we could ask questions about the quality and distribution of the data, and obtain advice on using the services effectively. This time, we have no such close link to the data providers, and we are simply using the data that they have made publicly available. As a result, meta-information such as the quality and coverage of the data sets had to be derived experimentally.

A diagram that illustrates the main components and services in our experiment is given in Figure 2. At the left of this diagram are the three data providers for our experiment, namely: SWISS, SAM, and ModBase. These providers all make their databases available through a standard web page (i.e. HTML) interface. We note that there is an extra pre-filtering step required for the ModBase database, and we do not operate on the ModBase dataset directly. This step is described later in this document.

To enable the various data sources to be used in our experiment, we have constructed a web service companion for each of the providers: a SWISS service, a SAM service, and a ModBase service. These companion services enable us to access the data through a standard web service (i.e. WSDL) interface. These services also provide the same abstract interface to the data sources, so that we can query them in exactly the same way. This interface corresponds to lookup(I, D) in Figure 1.

There are two additional services that we have constructed for our experiment. These services are illustrated on the right of Figure 2. The first of these is the MaxSub service, which provides a web service wrapper and WSDL interface for the MaxSub application. This application is used to perform comparisons between sequences. However, it was previously only available as a stand-alone application and could not be run over the web. The second service that we have constructed is the CYSP (Comparison of Yeast 3D Structure Predictions) service. This service is the core of our experimental process. It is responsible for querying the three data providers, invoking MaxSub to perform comparisons, and storing the results in our CYSP database. We provide our own database so that the experimental results can be reused without the need for recalculation, and for future experiment validation purposes. Our CYSP service effectively acts as a filter over the three data sources, and the interface to this service corresponds to filter(S_d, S) in Figure 1.
3 The MagentA Architecture

Our experiment is enacted by MagentA, which is at the centre of Figure 2. As previously noted, MagentA is essentially an interpreter for LCC specifications. A specification in LCC is parameterised by peer identifiers, and in our experimental scenario we have five peers: one for each of the five key web services. During execution, these peers will interact, and as a result the services that they represent will be dynamically composed. In this section we briefly outline the architecture of MagentA, and show how dynamic composition is achieved. Further details on MagentA can be found in [7, 9, 10].

The MagentA approach to dynamic composition is illustrated in Figure 3. MagentA is based entirely on existing Web Services technology, which permits third-party peers and services to be readily incorporated into the system without the need for modifications or plug-ins. MagentA is founded on the notion of an artificial social system, which is a technique from Multi-Agent Systems (MAS) used to coordinate large asynchronous systems of interacting entities. Peers are grouped together into societies, and communication only occurs within a specific society. The artificial society enables large systems to be constructed, and provides a controlled environment for interaction. To become a member of a society, a peer must agree to observe certain rules and conventions, and in return the peer will benefit from the other members of the society. The rules and conventions in an artificial social system are defined by explicit social norms.

A peer in MagentA is represented by a web service, and one or more prox-ies, which are marked A in our diagram. These proxies are responsible for the communicative processes of the peers, i.e. executing the appropriate part of the LCC protocol. Each web service encapsulates the capabilities of a specific peer.
At the core of the approach are MAS Services, which enable the proxies to interact. MAS services are web services that internally contain a peer communication environment. A single MAS Service is effectively responsible for a society of peers. All of the complex peer interactions happen within the MAS Service, while the external web services are accessed when necessary using standard remote procedure calls. The MAS service is also responsible for controlling entry and exit to the society, and for enforcing adherence to the social norms. In our approach, these social norms correspond directly to the LCC protocol that is executed. The MAS Service fulfills a similar role to the mediator or governor found in workflow-based composition systems, e.g. Taverna, though our use of societal concepts provides an alternative semantics.

Although the MAS services impose some central control on the system, the approach is still scalable. This is because a society will typically only contain a small number of peers that directly interact, even in a large system. Figure 4 shows how large systems can be constructed from multiple MAS services. These services may define different societies, or separate instances of the same society. Furthermore, a single peer can participate in multiple societies at the same time. That is, the proxies for a web service can reside inside different MAS services. Consequently, our Magenta approach can be considered as a peer-to-peer (P2P) architecture.

As previously noted, Magenta effectively performs dynamic composition of web services. An LCC protocol is executed by a collection of proxies, acting on behalf of specific peers. As the protocol is executed, the operations inside the web services will be invoked, and the results passed to different peers. The pattern of composition is determined by the flow of control through the LCC protocol. This control flow is affected by run-time events, and so the services will not necessarily
be invoked in the same way each time the protocol is executed. Thus, we have defined a protocol-based approach for dynamic service composition.

There are two main kinds of errors that can undermine dynamic service composition in MagentA. The first of these are network errors, where messages cannot be sent, or messages are not received when expected in the protocol. These kinds of errors can be handled by the timeout mechanism in LCC. Since LCC is a lightweight language, it does not directly prescribe how error recovery will be accomplished. Instead, it is up to the protocol designer to define their own recovery scheme. The timeout mechanism is the basis for the majority of such schemes (e.g. two-phase commit), and so these schemes can readily be encoded in LCC.

The second kind of error occurs when external services modify their interface definitions. This means that the service cannot be invoked, even though it is accessible on the network. If there is a choice of services with the same interface in the protocol, then the backtracking mechanism in LCC will invoke all of the alternatives in-turn. However, if there are no alternative services, then the protocol will be prematurely terminated. LCC protocols are written for specific services, and if the interface to a service changes, then the protocol must be similarly updated. We are currently investigating the use of semantic markup for services to enable such errors to be automatically handled.

The experiments described in this paper use only the basic features of MagentA and do not perform any error handling. This is because we have direct control over the web services that we use. We have constructed the five key service for our experiment: the SWISS service, the SAM service, the ModBase Service, the MaxSub service, and the CYP service. These services provide us with a uniform way to access the data sources, and to perform computation over the data. We have also used the MagentA tools to compose these services,
4 Example experiment: Consistency-checking in Protein Structure Prediction

Knowledge of a protein molecules three-dimensional structure is vital for understanding its function, targeting it for drug design, etc. The two prevalent techniques for determining the 3-D coordinates of all protein atoms with high precision are X-ray crystallography and nuclear magnetic resonance spectroscopy (NMR). However, compared with the case with which the amino acid sequences of proteins (their “1-D structures”) are deduced through the sequencing of genomes and cDNA, the effort and cost required for determining a protein 3-D structure remains tremendously high. Accordingly, a number of computational biology research groups specialise in producing structural models for proteins based on their amino acid sequences alone. Where they are accurate, predicted protein structures can provide valuable clues for biological research relating to these proteins.

Thanks to regular rounds of independent assessment using newly emerging atomic protein structures as “blind” tests at the so-called CASP [11] and CAFASP [12] experiments, protein structure prediction techniques have improved noticeably during the past decade. This is particularly true for template-directed protein structure prediction, in which the knowledge of a previously determined protein 3-D structure is used to generate a model for a different protein. If evolutionary relatedness between the two proteins can be established based on sequence similarity between a protein of interest and another protein whose 3-D structure is already known, then a model can be generated through comparative modelling. The known structure serves as a modelling template in this approach. The target protein sequence (i.e. the protein of interest) is aligned optimally onto the structural scaffold presented by the coordinate structure of this template. This typically includes the atoms making up the protein backbone, and the directionality of the side-chains (see [13] for an overview). Comparative modelling is generally considered “safe” to apply when the similarity between the sequences of target and template is sufficient to establish their alignment confidently over the whole length of the two proteins, or at least over relevant portions. However, the confidence in each individual prediction is not easily estimated at the time of the prediction. As a consequence, a substantial proportion of the models submitted for CASP/CAFASP comparative modelling targets are wrong and the degree of accuracy of their atomic coordinates varies substantially [11, 14].

In practice, biologist users of the model databases providing access to the structure predictions are often interested in a single target protein. Such users often apply a “consistency-checking” strategy to assess whether or not to trust the predicted coordinate structures for their protein of interest by comparing the models proposed by different groups/databases to each other. Where the
models agree, over the whole or a part of the protein molecule, they are deemed an approximately correct representation of the actual 3-D molecular structure of the target protein.

4.1 3-D Structural Models for Yeast Proteins

As an implemented example in which a consistency-checking experiment is undertaken we have investigated the consistency between pre-computed comparative models for the proteins encoded by the genome of the budding yeast *Saccharomyces cerevisiae*. The yeast genome sequence has been known since 1996 [15] and it is currently predicted to encode 6004 proteins. For 330 of these proteins (or fragments of them) 3-D structures for have been determined through X-ray crystallography or NMR to date. For this experiment we selected three public-domain repositories offering access to pre-computed coordinate models for yeast proteins generated by different automated methods: SWISS-MODEL [16], MODELLER (ModBase) [17], and SAM-T02/Undertaker [18]. We systematically retrieved and compared the models for all yeast proteins with to-date undetermined structures and extracted a sub-set of protein models that were “validated” by agreement between all three methods.

4.2 Data Sources

The systematic open-reading frame (ORF) names of all predicted protein-encoding genes in the yeast genome, commonly referred to as YIDs, were extracted from the Saccharomyces Genome Database (SGD) [19]. The 6004 ORFs listed in SGD on 7 June 2006 were used to query the three model databases.

**SWISS**: The SWISS-MODEL Repository [20] is a database of annotated protein structure models generated by the SWISS-MODEL [16] comparative modelling pipeline. SWISS draws the target sequences for its entries from UNIPROT (the successor of SWISS-PROT/TrEMBL). Yeast proteins are also annotated with their YIDs. Only models for proteins of unsolved structures are accessible. If the structure of a protein was already determined experimentally (through X-ray crystallography or NMR) SWISS links directly to this structure in the PDB [21].

**ModBase**: This database [22] contains comparative models generated by the program ModPipe (an integration of PSI-BLAST [23] and MODELLER [17]) based on protein sequences extracted from SWISS-PROT/TrEMBL. ModBase typically contains a large number of models for the same target protein that can be considered redundant and imposes only minimal quality standards. In order to streamline the procedure for this experiment and only work with models that have chances of being correct we downloaded all ModBase entries for yeast proteins and eliminated redundancy and extremely low-quality models from the set locally (see below). Note that, by contrast to SWISS, ModBase also contains models for proteins with crystallographically or spectroscopically determined structures (re-modelled onto themselves as templates and/or closely homologous proteins).
SAM: The Karplus group at UC Santa Cruz provides WWW-access to provisional models for all predicted yeast proteins using their combination of local-structure, hidden Markov model (HMM)-based fold recognition and ab initio prediction [18]. The methodology underlying fold recognition is similar to that of the comparative modeling in that a template of known structure is used. However, besides various technical differences, the application range targeted by fold recognition methods differs from that of the programs used by SWISS and ModBase in that the former specialize in predicting structures where target-template sequence similarity is too remote to detect by standard methods. Accordingly, this data source offers protein structure predictions for all ORFs of yeast (including some not listed as genes by the SGD database) and a confidence estimate for the corresponding target-template matches together with a collection of provisional (i.e. unrefined), often fragmented, coordinate models for each.

4.3 Processing/Pre-Filtering of Data Sets

As described above the amount of data pertaining to each YID, and its organization, differs quite dramatically between the three data sources. To ensure that the structural comparisons between the models could be run efficiently for the set of all available yeast models, we undertook a minimum of local processing and pre-filtering of the entries after retrieving them from the three databases. Note that most of this would be unnecessary in the more common situation where a biologist user is interested in comparing only the entries pertaining to a smaller set of YIDs, usually even only a single protein. (In this case, a simple cross-check of the confidence value (which is often expressed as an E-value) associated with the model and/or a search for the best model available could be incorporated in the interaction and a larger number of pair-wise comparisons would be undertaken to extract the protein model region that is supported by the different modelling methods.)

ModBase: Our Magenta interface to ModBase retrieved 3448 files with model coordinates when queried with the list of YIDs. These files typically include more than one 3-D model for the same protein sequence (see above). We pre-filtered the ModBase set of models in two steps, one selecting only high-quality comparative models, and a second to eliminate redundancy. Our selection criteria for "high-quality" models were: percentage sequence identity between target and template $> 20\%$; model score $> 0.7$; E-value $< 1E-06$. The relevant values were directly accessible within the “REMARK” part of each 3-D coordinate file. Eliminating redundancy is important in cases where individual proteins are represented by multiple “high-quality” models in ModBase. As shown schematically in Figure 5, the sequence regions covered by the different models often overlap. This can occur because different template structures were chosen (correctly, or incorrectly) each giving rise to one modeled region. To allow efficient comparison of the complete set of yeast models, we made a choice as to which one model was retained if such a redundant set was encountered. This was based on clustering of the model protein sequences extracted from the 3-D
coordinate files for each YID using the program BLASTclust which is part of the BLAST suite (http://www.ncbi.nlm.nih.gov/BLAST/download.shtml). Of multiple models with > 90% pair-wise sequence identity over at least 90% (of the length of the shorter model in each comparison), only the model covering the largest sequence region was retained (Model 1 in the example in Figure 5).

Fig. 5. Schematic showing multiple redundant models for one protein. The red line represents the full-length sequence for this protein (running from the N-terminus to the C-terminus of the molecule). Blue lines represent different models (1, 2, 3, 4, and 5) covering different, overlapping and non-overlapping, regions.

By contrast to such instances of redundancy, some proteins may represented by several models without substantial overlap. Protein structures are generally easier to determine experimentally if they only encompass a small number of structural domains. Accordingly, multi-domain protein sequences are often covered using different template structures for each domain, thus giving rise to several meaningful models. These models may not overlap at all (Figure 6A), or partly (less than 90%) overlap with each other (Figure 6B). Multiple models of this kind were retained in our filtered ModBase model set, which contained 2546 models for 2280 yeast proteins.

SWISS: The majority of yeast proteins that can be found in SWISS database only have one associated 3-D model. By contrast to ModBase multiple entries in SWISS can be considered non-redundant, i.e. relate to multi-domain proteins. Moreover, stringent quality standards are imposed by the authors of the database. Our Magenta interface to SWISS queried the “Advanced Search” WWW-interface to the database (swissmodel.expasy.org/repository/smr.php?job=3) with the complete list of YIDs and retrieved 769 3-D models for 717 proteins when queried. In the case of multi-domain proteins, all available 3-D models were extracted. An additional 330 returns were crystallographically or spectroscopically determined structures extracted from the PDB database; these were disregarded in the structural comparisons.

SAM: Our Magenta interface to the SAM database of yeast models (http://www.soe.ucsc.edu/research/combio/yeast-protein-predictions/lookup.html) returned sets of 3-D models for all 6604 YIDs. The models delivered in each set are based on different templates and ordered according to SAM-T02s confidence in the underlying target-template match. Unfortunately this organization is not
Fig. 6. Schematic showing multiple non-redundant models for one multi-domain protein. The red lines represent the full-length sequence for two multi-domain proteins. Blue lines represent models covering different regions of these sequences.

suited for extracting multiple non-redundant models from the set easily in the case of multi-domain models. For the purpose of this experiment we chose to select the top model from each set, which will usually also select the model covering the largest region of the sequence. In addition we imposed a maximum E-value cut-off of 1E-03 for the target-template match. This resulted in 2211 SAM models being considered in the structural comparisons described below.

4.4 Consistency Checking

Pair-wise comparisons between the retained 3-D models from the three data sources relating to the same YID were carried out with the program MaxSub [24]. MaxSub performs sequence-dependent pair-wise comparisons between different 3-D structures (predicted models or known structures) of the same proteins, aiming to find the largest substructure over which the two structures superimpose well upon each other. It only considers the base (Cα) atoms of the protein side-chain and also ignores the details of the other backbone atoms. As a metric of the similarity of the two structures that are being compared, MaxSub computes a single score (referred to as Mscore below) ranging from 0 for a completely unmatched pair, to 1 for a perfect match. Since the values for Mscore are asymmetrical, i.e. dependent on which of the two proteins is considered to be the reference protein, we carried out the pair-wise comparisons in forward and reverse order. The distance threshold parameter was set as 3.5 Å throughout the analysis. For proteins whose 3-D structures were previously determined in the laboratory, there is no interest in a comparison between the X-ray/NMR structure and the 3-D models contained in ModBase and SAM, since the known structures may have been used as the modelling template. (This is different for newly determined structures which will be useful for evaluating the accuracy of the models, as is discussed below.) Pair-wise model comparisons were performed for all YIDs represented by at least one retained model in each of the three sets.
In total 4556 pair-wise comparisons of the remaining yeast 3-D models returned non-zero results.

Based on the pair-wise comparisons we extracted three-way “MaxSub-supported substructures”, i.e. the maximum overlap between all pair-wise matched regions for the same protein sequence. In the derivation of these substructures (illustrated schematically in Figure 7) we chose to ignore the gaps of up to 35 consecutive amino acids that were found sometimes within the regions matched in the MaxSub comparisons between two 3-D models. Such gaps were caused either by strong local deviation between the two models or missing residues in one of the models. MaxSub-supported substructures encompassing fragments of less than 45 amino acids in length were discarded to keep the number of structurally uninteresting matches (for example over only a single α-helix) as small as possible.

![Fig. 7. Schematic illustrating the derivation of three-way MaxSub-supported substructures. Two examples are shown, a single-domain protein (A) and a multi-domain protein (B). Red lines represent full-length protein sequences. Blue lines represent the pair-wise matched regions between 3-D models for these proteins. Green lines represent the resulting MaxSub-supported substructures.](image)

### 4.5 Results

The detailed results of this experiment are publicly accessible via our WWW-database CYSP (Comparison of yeast 3-D structure predictions, linked from [www.openk.org](http://www.openk.org)). The records currently relate to the yeast proteins for which at
least one model was retained in each of our sets of SWISS, ModBase, and SAM models after pre-filtering. Information is given regarding their associated 3-D model coordinates as they can be obtained from the three repositories, which regions match pair-wise between 3-D models of the same protein by different methods according to MaxSub comparison, and the Mscores attained by the matches. For proteins where three-way agreement between the methods was found, 3-D coordinates are also provided for the model fragment spanning their Max-Sub supported substructures. To the non-computational biologist looking to find an approximate 3-D structural representation of his/her yeast protein of interest, the model fragments in this new, filtered, resource are likely to be the most relevant. While there is of course no guarantee (since there always is a chance that all three methods could have erred in the same way) they would be deemed “likely correct by consensus”. This philosophy is applied widely in other areas of protein structure prediction as well, for example in secondary structure prediction, and its viability is generally supported by independently derived experimental structural information [25–27]. Attributing greater confidence to consensus predictions is certainly considered appropriate where the methods consulted are different as this was the case here.

A previous similar study by the Baker group at Washington University St. Louis compared fold predictions for yeast proteins between different fold prediction methods [28]. By contrast to our comparisons Dolinsky et al. did not carry out model superpositions but designed their SPrCY database (http://agave.wustl.edu/yeast/) for consistency checking at the template structure/fold level. Given the lower structural accuracy in general that is attained by models based on fold prediction methods (which aim primarily to identify fold resemblance to known structures in cases where no sequence similarity is detectable, and where producing a detailed model is often too difficult a problem to tackle) this is certainly justified, although it makes it impossible to directly compare our results with theirs.

We obtained 584 MaxSub-supported substructures for 545 yeast protein sequences with non-identical YIDs in this experiment. Fragments of 3-D models are most informative to the users if they span entire structural domains, or at least 3-D structurally separable parts. While some few domains are known to include less than 45 amino acids, and domain lengths spread widely, short fragments should be considered more at risk of being structurally uninformative than long fragments. As the length distribution of MaxSub-supported substructures shows (Figure 8) we would retain 136 (23\%) of the corresponding model fragments even if a minimum length of 90 amino acids were imposed, rather than the 45 amino acid cut-off we chose. Thus it seems likely that the majority of the model fragments in CYSP would be useful for investigating the local structure of the yeast proteins they represent. This notion was confirmed through visual inspection of the models and is illustrated by the three examples presented in more detail below.

The number of models yielding pair-wise matches is shown in Table 1. The number of matched models between SWISS and SAM is very similar to the
Fig. 8. Length distribution of MaxSub-supported substructures

<table>
<thead>
<tr>
<th></th>
<th>SWISS</th>
<th>ModBase</th>
<th>SAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWISS</td>
<td>459 (717)</td>
<td>649 (694)</td>
<td>685 (559)</td>
</tr>
<tr>
<td>ModBase</td>
<td>2546 (2280)</td>
<td>620 (594)</td>
<td></td>
</tr>
<tr>
<td>SAM</td>
<td></td>
<td>2211 (2211)</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Number of pair-wise matched regions between models from the three data sources (ignoring gaps). The numbers of represented yeast proteins are given in parentheses. The total number of models and proteins in each set that were considered is apparent in the diagonal. Note that comparisons were only performed for YIDs represented by at least one retained model in each filtered set.
number of three-way MaxSub-supported substructures. At first glance this coincidence may appear to reflect that only one of the SWISS-SAM matched regions is not also supported by ModBase. However, this is not necessarily true as multi-domain proteins can give rise to different numbers of pair-wise matched regions depending on which methods are compared (as is illustrated in Figure 7B). Indeed inspection of the results reveals that the relation between the pair-wise and three-way supported regions is not straightforward. Comparing the numbers of represented yeast proteins should be more informative and we note that the 545 proteins represented by the three-way MaxSub-supported substructure set in CYSP make up make up 91.8% of those represented in SWISS-ModBase; 97.5% of those in SWISS-SAM; and 91.8% of those in ModBase-SAM. While these numbers do not differ dramatically for different method-pairs, the proportion is higher for SWISS-SAM is than for the others. If one assumed that all three-way supported substructures are correct, but that there are likely to be more correct predictions in the set than the ones supported by all three methods, this could be explained by SAM being a slightly weaker predictor than the other two in this experiment. If this were the case it could be useful to look at the SWISS-ModBase set for additional (possibly correct) models. Alternatively, if one assumed that the three-way supported predictions are the only ones that are correct then this difference would indicate that SAM is the most useful method for preventing false models (which would make most sense if ModBase and SWISS were very similar methods). Neither of these assumptions can be expected to be entirely accurate (no known method guarantees that three-way supported predictions are actually correct) and forthcoming laboratory-determined protein structures will only provide an evaluation of a small number of the predictions in the near future. Moreover it would not be appropriate to derive more than very cautious conclusions based on this survey data. However, given the fact that the models provided by the SAM data source are deemed to be at “unrefined” stage, it is plausible that the first possibility is closer to the truth than the second. Our implementation makes it straightforward to perform a repeat experiment at a later stage of SAM-model refinement and/or to consult additional data sources in the future.

To illustrate the results accessible at CYSP we have selected three examples: YPL132W, YBR024W, and YLR132C. The 3-D models of YPL132W in SWISS, ModBase, and SAM were all generated based on the same template structure, 1SO9A. By contrast, different template structures were used by the each of the three model sources to model YBR024W and YLR131C (Table 2).

Pair-wise comparisons between 3-D models of YPL131W generated by SWISS, ModBase and SAM indicate that, with exception of some missing residues, the models are in perfect agreement throughout (Table 3 and Figure 9). By contrast, the three data sources only agree on the core regions of the structures predicted for YBR024W (the blue regions), and disagree otherwise (the green and red regions). Finally the three data sources disagree almost entirely on YLR131C, except over a short α-helical region (the blue regions). In this example the MaxSub-supported substructure would not be considered informative.
<table>
<thead>
<tr>
<th>YID</th>
<th>Protein Name</th>
<th>Template (E-value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YPL132W</td>
<td>CGX11_YEAST</td>
<td>1S99A (4.5E-75)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1S99A (4.5E-53)</td>
</tr>
<tr>
<td>YBR242W</td>
<td>WSC02_YEAST</td>
<td>1P7G (1.3E-76)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1SHM (1.0E-20)</td>
</tr>
<tr>
<td>YLR131W</td>
<td>CAC220_YEAST</td>
<td>1NCG (4.2E-17)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1NGB (2.9E-15)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2GL1A (4.7E-27)</td>
</tr>
</tbody>
</table>

Table 2. Three examples of proteins included in CYSP, specified by their YIDs and SWISS-PROT identifiers, with the template structures used and the E-values the three data sources attributed to their structure predictions.

Since it is shorter than 45 amino it was discarded and is not included in the results accessible through CYSP.

Some validation of this experiment will become possible in the future as additional structures of yeast proteins are determined in the laboratory. These newly emerging structures will allow an assessment of the accuracy of our models at least in a few cases. It will be interesting to verify, at least qualitatively, the assumption that the approach we applied extracted more accurate substructures than the models that were less well supported. To this end we are keeping track of new yeast protein structures in the PDB, and the accuracy of the corresponding models in CYSP, on www.openk.org.

5 Conclusions and Future Work

In the experiment of Section 4 we grounded the experimental protocol of Figure 1 in a specific set of services, using the MagentA system as an interpreter for the protocol. It was necessary to route all the data and analysis services through MagentA (in the way described in Figure 2) because none of the original services was equipped to interpret the protocol. We therefore incurred a small one-off cost in enabling (via WSDL and HTML) the original services to communicate with MagentA. Having done this, however, we are able to use MagentA as a proxy for the original services for any LCC protocol, so that experimenters with different ideas about how best to coordinate these (and other suitably enabled) services can implement these by altering only the protocol. Notice also that the LCC protocol is separable from the mechanisms used to interpret it, and is shareable between peers during an interaction, so we can choose whether we want a single MagentA proxy for a group of services or a separate proxy for each service (giving a more or less finely grained peer-to-peer structure).

Since MagentA is capable of interpreting any LCC protocol, we can in future add to the repertoire of protocols and thus extend the capabilities of the peer-to-peer system. For example, it is straightforward to write a LCC protocol for sharing filtered data between peers. It is also straightforward to write a LCC protocol that allows queries about specific types of protein structure to be routed between peers, thus allowing networks of peers to collate, filter and propagate results. The aim of this, ultimately, is to provide a peer network.
<table>
<thead>
<tr>
<th>YID</th>
<th>Model 1</th>
<th>Model 2</th>
<th>Missing residues</th>
<th>Pair-wise matched regions</th>
<th>MScore</th>
<th>MaxSub supported region</th>
</tr>
</thead>
<tbody>
<tr>
<td>YPL132W</td>
<td>SWISS</td>
<td>SAM</td>
<td>2</td>
<td>GLU138-GLU170, ALA172-GLY219, GLU221-PHE253</td>
<td>0.904</td>
<td>GLU138-PHE253</td>
</tr>
<tr>
<td></td>
<td>ModBase</td>
<td>SAM</td>
<td></td>
<td>VAL133-GLU170, ALA172-GLY219, GLU221-ALA255</td>
<td>0.940</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SWISS</td>
<td>ModBase</td>
<td>0</td>
<td>GLU138-PHE253</td>
<td>0.954</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SWISS</td>
<td>SAM</td>
<td>44</td>
<td>ALA120-ALA120, GLY122-PHE125, LEU127-LYS143, SER145-TYR148, SER152-HIS153, GLU160-GLU160, LEU162-ARG164, THR166-LYS175, HIS177-ILE178, ILE180-ASP203, ILE208-SER230, TYR250-GLY263, TYR266-ARG276, GLN278-ILE279</td>
<td>0.660</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SWISS</td>
<td>ModBase</td>
<td>22</td>
<td>PRO124-LYS141, LYS143-HIS153, PRO155-PRO155, GLU160-SER170, GLN181-PHE204, PRO206-PRO206, PHE248-PRO254, GLY256-LEU262, ARG264-ARG264</td>
<td>0.400</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SWISS</td>
<td>SAM</td>
<td>0</td>
<td>PRO97-ILE91, VAL95-VAL595, PRO599-ASP600, LEU602-PHE614, ARG617-GLN629</td>
<td>0.179</td>
<td></td>
</tr>
<tr>
<td>YLR131C</td>
<td>ModBase</td>
<td>SAM</td>
<td>25</td>
<td>LEU606-LEU606, CYS616-ASN611, PHE614-PHE614, ASN619-PHE637</td>
<td>0.000</td>
<td>(PRO606-GLN629)</td>
</tr>
<tr>
<td></td>
<td>SWISS</td>
<td>ModBase</td>
<td>10</td>
<td>LEU598-LEU598, CYS616-ASN611, PHE614-PHE614, ASN619-PHE637</td>
<td>0.000</td>
<td>(GlN629)</td>
</tr>
</tbody>
</table>

Table 3. This table lists information extracted from the pair-wise structural comparison results by MaxSub. The missing residue column reports how many residues in Model 1 are not found in Model 2. Reported as well-matched regions are all strictly continuous sequence fragments over which Model 1 coincided well with Model 2 after 3-D structural sequence-dependent superposition.
**Fig. 9.** Backbone representations of the MaxSub results for three proteins: YPL135W, YBR024W, and YLR131C. For each pair-wise comparison between the structures of Model 1 and Model 2 (in the order MaxSub read these models), substructures in blue are the regions in Model 1 that superimpose well onto Model 2, while the other parts of the models are shown in green (Model 1) and red (Model 2).
that, through sharing, can produce more confident predictions faster by sharing the analyses performed earlier by others. Opportunities for applying similar consistency-checking, and data sharing, strategies are found in many areas of bioinformatics. The single experiment in this paper shows the immediate benefit of this on a small scale for a specific form of analysis but to make this effective for large peer groups, where trust and provenance are (among other issues) important to the coherence of peer groups. This, although outside the scope of the current paper, is one of the central themes of the OpenKnowledge project (www.openk.org).

References