Automating the Determination of Open Reading Frames in Genomic Sequences Using the Web Service Techniques -- A Case Study using SARS Coronavirus

Paul Hsueh-Min Chang, Von-Wun Soo, Tai-Yu Chen, Wei-Shen Lai, Shiun-Cheng Su and Yu-Ling Huang
Department of Computer Science, National Tsing-Hua University
101, Section 2, Kuang-Fu Road, Hsinchu, Taiwan 300, R.O.C.
pchang@cs.nthu.edu.tw, soo@cs.nthu.edu.tw, g924326@oz.nthu.edu.tw,
g926325@oz.nthu.edu.tw, d86623801@ntu.edu.tw, rebecca@ailab0.cs.nthu.edu.tw
+886-3-5715131 ext 4199

Abstract

As more and more new genome sequences were reported nowadays, analyzing the functions of a new genome sequence becomes more and more desirable and compelling. However, the determination of the functions of a genomic sequence is not an easy task. Even with several bioinformatic tools, the task is still a labor-intensive one. This is because human experts have to intervene during the processing of using these tools. For efficiency, immediacy and reduction of human labor, a system of automating the analyzing process is proposed. We take the automated determination of Open Reading Frames of a genomic sequence as the domain tasks that involve using a number of computational tools and interpreting the results returned from the tools. A service-oriented approach is taken, in which analyzing tools are wrapped as Web services and described in Semantic Web languages including OWL and OWL-S. The SARS Coronavirus genomic sequence is taken as a test case for our approaches. We are in the process of building an agent-based system for automating the tasks, in which an intelligent agent is responsible for understanding purposes of the Web services by parsing the service descriptions, and carrying out the interpretation tasks according to a workflow.

1. Introduction

Modern advancement in bioinformatics has resulted in computational tools that help analyze gene or protein sequences and match similar sequences in the database.

Biologists can run these tools to gain information about an unknown sequence and predict its functionalities before conducting actual laboratory work.

Although computational tools enable fast sequence analysis, they alone do not make the analyzing process automatic because operating the tools requires intense human attention. For example, to find out the functionalities of the genome sequence of SARS Coronavirus [1] [2], its RNA (in FASTA format) can be sliced into more than 100 open reading frames, each of which is then sent to Blastp to find out the matching protein. Running an analyzing process like this can be tedious for the human operator. Hence it is desirable to automate the process at least partially. Besides alleviating human workload, automation also speeds up the analysis. In urgent situations such as the appearance of a new disease caused by an unidentified virus, quick information about the virus is useful even if the information is not complete.

This requirement of automating analyzing processes is compatible with the recent research theme of Web services [3]. A sequence processing tool, once wrapped as a Web service, is syntactically interoperable since it can be accessed by programs written in any programming language. Moreover, a Web service can be semantically interoperable if it is described with semantic markups in languages such as OWL-S [4]. Semantic interoperability means that the usage and functionalities of the service can be understood by parsing the semantic annotations. Given biological tools wrapped as Web services and described with semantic markups, an intelligent agent can utilize the tools to carry out the processing task. More specifically, the agent looks up in the service directory for potentially useful Web services after delegated the processing goal of the biologist. The agent then search for Web services that fits in the workflow and choose among them by reading the semantic markups. Finally the agent accesses the chosen services either sequentially or in parallel

Proceedings of the Fourth IEEE Symposium on Bioinformatics and Bioengineering (BIBE'04)
0-7695-2173-8/04 $ 20.00 © 2004 IEEE

according to the workflow specification, and then output the results. Furthermore, the agent can arrange the results and interpret them in a way useful for biologists by referring to domain ontologies. The construction of good domain ontologies and service descriptions relies on the knowledge and experience of biology experts.

This paper is organized as follows. Section 2 elaborates on the motivation, which is to automate the analyzing process of SARS Coronavirus. Section 3 describes the workflow of the analysis in detail. Section 4 describes the Web-service-based approach. Section 5 reports the work in progress. Section 6 concludes this paper.

2. Background and motivation

The analysis of SARS Coronavirus from [1] is taken as the topic of our initial study because of its public familiarity due to the SARS outbreak in the year of 2003. Since SARS Coronavirus has a shorter genome sequence than germs or more complicated life forms, the processing steps are relatively simple. We take this analysis as a starting point; once it can be successfully automated, the approach and techniques can be extended to apply on complicated biological analyses such as those on E. coli.

Thus, the motivation of this work is to reconstruct the analysis in [1]. The first step is to understand the workflow of the analysis, which involvers a workflow consisting of sequence processing tools such as ORF finder, Blastp and Pfam. Then an agent-based system is build to perform the analysis automatically. However, instead of being restricted to a single purpose, the agent-based system must be flexible and extensible. By being flexible the system can accept different goals from the biologist if the goal is compatible with the capabilities of the analyzing tools. By being extensible the system can be enhanced with additional tools and knowledge without need to change the system architecture. The Web service approach is introduced to satisfy these two desirable properties. Information about how to operate a tool and what the tool does should be distributed to the description of each service rather than hard coded within the agent code. The agent must be able to consult an external biological ontology, written in OWL [5], to understand the meanings of the service descriptions, written in OWL-S.

The service descriptions and ontology enables reasoning about the capabilities of the tools. Thus this architecture is extensible because new processing tools can be wrapped as Web services and added to the system, and flexible because a Web service can be used in various circumstances that change with the processing goal.

3. Workflow of the SARS genome analysis

The goal of the SARS analysis is to identify potentially functional subsequences of SARS Coronavirus, and to annotate the subsequences with additional characters such as protein domain. Table 1 lists all sequence processing tools used in this analysis. Figure 1 illustrates the workflow of the analysis. The analysis can be divided into five steps, each of which is described in detail in the following sections.

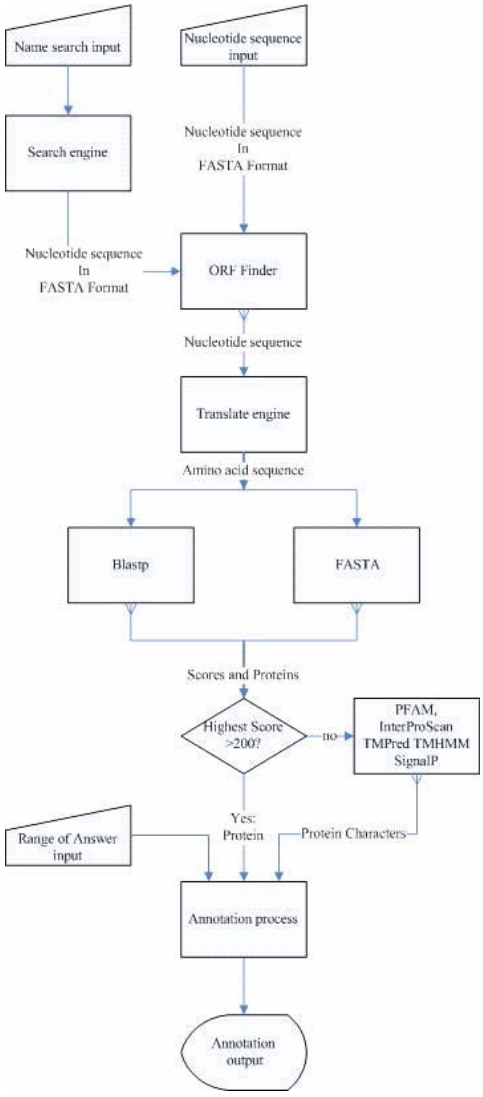

Figure 1. The workflow of the genome sequence analysis of SARS Coronavirus

Table 1. The tools used in the genomic sequence analysis for finding ORF

Tool	Input	Function	Link
ORF Finder	Nucleotide sequence	Find open reading frames.	http://www.ncbi.nih.gov/gorf/gorf.html
Transeq	Nucleotide sequences	Translate nucleotide sequences to amino acid sequences	http://www.ebi.ac.uk/emboss/transeq/
Blastp	Amino acid sequence	Compare amino acid sequences to known proteins.	http://www.ncbi.nlm.nih.gov/BLAST/Blast.cgi
Fasta	Amino acid sequence	Same as Blastp	http://www.ebi.ac.uk/fasta33/
Pfam	Amino acid sequence	Compare amino acid sequences to known proteins	http://www.sanger.ac.uk/Software/Pfam/search.shtml
TMpred	Amino acid sequence	Suggest models for transmembrane topology	http://www.ch.embnet.org/software/TMPRED_form.html
TMHMM	Amino acid sequence	Predict transmembrane helices in proteins	http://www.cbs.dtu.dk/services/TMHMM/
SignalP	Amino acid sequence	Indicate the cleavage site of a protein	http://www.cbs.dtu.dk/services/SignalP-2.0/
InterProScan	Amino acid sequence	Same as Pfam.	http://www.ebi.ac.uk/InterProScan/

Step 1 Predict potential ORF subsequences.

A biologist may have a genome sequence in hand or can find a sequence through the search engine on NCBI web site. Then sequence must be in FASTA format before sent to the ORF finder. The ORF finder slices the sequence into potentially functional subsequences known as open reading frames.

Step 2 Translate the subsequences

The open reading frames are nucleotide sequences, but further analyses require amino acid sequences as inputs. Therefore the open reading frames must be translated to amino acid sequences. Each translated subsequence is a potential protein. The Transeq translation engine does the job.

Step 3 Search for similar proteins

The translated subsequences are then passed to Blastp and FASTA to perform sequence similarity search. Both Blastp and FASTA compare the input sequences with the known proteins in their data banks and return a list of proteins which is similar to the input sequence. The protein lists are sorted in descending order according to the scores computed by Blastp and FASTA. In general, we care more about the protein with the highest score.

Thus the translated subsequences can be divided into three groups according to the highest score in the protein list of each subsequence. The first group contains subsequences with a highest score which is larger than 200. They are believed to have a correct match, and therefore their importance is clear. The second group contains subsequences with highest scores between 200 and 50. Deciding whether the protein with the highest score is a correct match is not easy. Subsequences with highest scores lower than 50 are believed to be have an incorrect match. It means that either the open reading frames in this group are non-functional or that they represent a new protein that the data banks do not contain.

Step 4 Gather characters of the those proteins

Since whether the subsequences belonging to the second and the third group represents a protein is not certain, other tools including Pfam, InterProScan, SignalP, TMPred and TMHMM can be used to gather more characteristics about these subsequences that may or may not be proteins. Pfam and InterProScan can be used to determine the domain of a protein. TMPred and TMHMM can predict if a protein is a membrane protein. SignalP can indicate the cleavage site of a protein. Characteristics of the low score subsequences can give the biologists more clues so that the biologisits can make the better decision.

Step 5 Annotate and process output

A user may need different output for different purposes. In this step we first integrate the outputs of step 3 and step 4 into a complete output. Then we process the output according to the biologist? need. For example, a biologist may need the subsequences in the first group if she wants to find out the functionalities of a new genome sequence.

However another biologist may only need the subsequences in the second and the third group if she wants to find a new protein.

4. Automation approach

This section describes our approach of automating genome sequence analysis described in the previous section.

4.1. System architecture

The architectural overview is illustrated in Figure 2. Our system currently relies on existing Web-based tools such as those on the NCBI Web server. Although all tools required to analyze SARS Coronavirus can be found on the Web servers, they have to be accessed by human through a Web browser interface. Thus we wrap these tools as Web services on our Web server to allow programmatic access with SOAP protocol. Each Web service consists of a Web client that communicates with the tool website using plain HTTP Get/Post protocol. A Web client also contains a parser that parses the returned results (often in HTML) and transforms them to structured data. The Web service wrapper then wraps the Web client, so that any requests sent to the Web service wrapper will be transferred to the Web client and then redirected to the tool website for the results. Note that a Web service wrapper can do additional processing before and after the request is sent to the tool website. For example, the ORF finder Web service takes a parameter that specifies whether the input sequence is originated from a DNA virus, an RNA+ virus or an RNA- virus. This parameter helps the Web service filtering unnecessary strands. For example, SARS Coronavirus is an RNA+ virus, thus only the three sense strands are necessary. The ORF finder Web service parses the result returned from the NCBI ORF finder tool and returns only the necessary open reading frames to the requester. The Web services are then described with usage information, both syntactically in WSDL [6] and semantically in OWL-S. We will discuss more on the method to describe services in the next subsection.

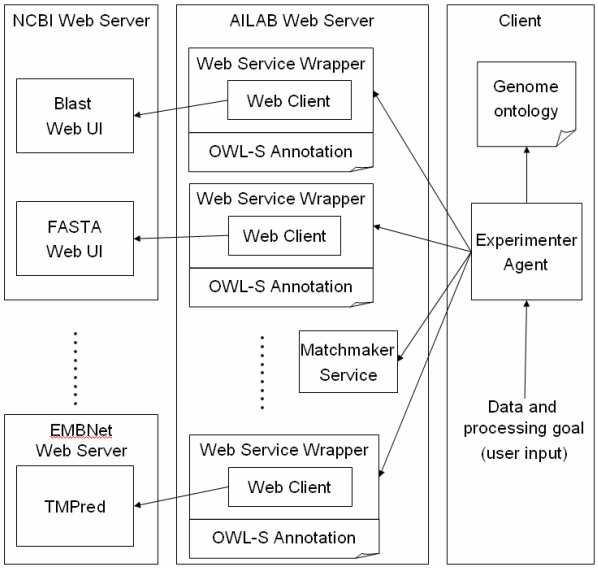

Figure 2. Overview of the system architecture based on Web services.

The experimenter agent on the client machine receives the data, which is the name of SARS Coronavirus in this example, and the processing goal, which is to annotate the open reading frames with matching proteins and other related information. The processing goal is written in a high-level goal language that allows biologists to specify the goals without going into great details. After receiving the processing goal and data, the experimenter agent contacts the matchmaker service to discover services needed for the goal. The matchmaker returns a list of URLs, through which the experimenter agent retrieves the description of each potentially useful Web service. The experimenter agent consults the genome ontology for unknown concepts in the OWL-S service descriptions to understand the function of each Web service. Then it can compose a workflow and execute that workflow by contacting each Web services.

4.2. Service description

The basic description of Web services is WSDL, which describes the syntax of the interface exposed by the service, including the service URI, operation names, and the name and datatype of each parameter. A client program can follow the instructions in the WSDL document to invoke the Web service. A WSDL document includes necessary but not sufficient information about a sequence processing tool wrapped as a Web service, because WSDL describes how to invoke an operation, but not the purpose and functionality of the operation. Take the ORF finder Web service as an example. WSDL describes the input to the ORF finder as a string and the output as an array of strings. An agent can follow the WSDL document to invoke the service, but it does not know that the input is a DNA sequence and the output is the collection of open reading frames in the sequences.

OWL-S, the W3C successor of DAML-S, fills the semantic gap. OWL-S describes a service as a collection of processes. Each atomic process is associated with an operation in WSDL and is characterized by its inputs, outputs, preconditions and effects. The inputs and outputs of a service are described with concepts in the supporting genome ontology, as depicted in Figure 3. Thus, sequence processing tools including ORF finder, Blastp, FASTA, TMHMM and TMPred are all atomic processes. However we observed that some tools are similar in functionalities. For example, Blastp and FASTA might perform differently in different circumstances but their purposes are similar. So is the case between Pfam and InterProScan. Thus, specifying an abstract process template for a set of similar tools is

preferable to describing each tool in an ad-hoc way. The process template can be created as Simple Process classes in OWL-S. Figure 4 shows a simple process for protein similarity search, which is the task to be carried out by Blastp and FASTA. Other process templates include the sequence lookup by name, the translation from DNA to protein, the finding of ORF, the protein domain analysis and the membrane analysis.

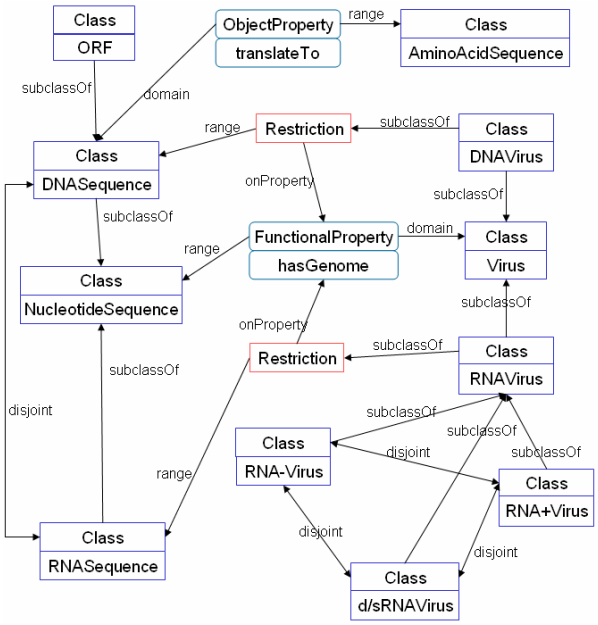

Figure 3. Part of the supporting genome ontology in OWL.

The Protein Similarity Search simple process in Figure 4. is an abstract process for protein similarity search. The input is an amino acid sequence, which is defined as the AminoAcidSequence class in the genome ontology of Figure 3. The output is an array of (matching protein, score) pairs. The two atomic processes, Blastp and FASTA, are the concrete implementation of the simple process. In other words, they *realize* the Protein Similarity Search simple process. The matchmaker service uses the simple process to match the experimenter agent request. That is, when the experimenter agent requests for a tool for performing protein similarity search, the matchmaker service returns the Web services implementing the Protein Similarity Search simple process. In this case they are Blastp and FASTA, but there might be more if the system is added with new Web services with similar functionalities.

```
<process:SimpleProcess rdf:ID="ProteinSimilaritySearch">
  <process:hasPrecondition rdf:resource="#OrfFound"/>
  <process:hasEffect rdf:resource="#SequenceCompared">
  <process:hasInput>
    <process:Input rdf:ID="ProteinMatchInput">
      <process:parameterType rdf:resource=
                        "&genome;#AminoAcidSequence"/>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:Output rdf:ID="ProteinMatchOutput">
      <process:parameterType rdf:resource=
                        "#ProteinMatchResult"/>
    </process:Output>
  </process:hasOutput>
</process:SimpleProcess>
<owl:Class rdf:ID="ProteinMatchResult"/>
<owl:Class rdf:ID="ProteinMatchResultItem"/>
<owl:ObjectProperty rdf:ID="hasItem">
  <rdfs:domain rdf:resource="#ProteinMatchResult"/>
  <rdfs:range rdf:resource="#ProteinMatchResultItem"/>
</owl:ObjectProperty>
<owl:FunctionalProperty rdf:ID="matchingProtein">
  <rdfs:domain rdf:resource="#ProteinMatchResultItem"/>
  <rdfs:range rdf:resource="&genome;#AminoAcidSequence"/>
</owl:FunctionalProperty>
<owl:DatatypeProperty rdf:ID="score">
  <rdfs:domain rdf:resource="#ProteinMatchResultItem"/>
  <rdfs:range rdf:resource="&xsd;#integer"/>
</owl:DatatypeProperty>
<process:AtomicProcess rdf:ID="Blastp">
  <process:realizes rdf:resource="#ProteinSimilaritySearch"/>
</process:AtomicProcess>
<process:AtomicProcess rdf:ID="FASTA">
  <process:realizes rdf:resource="#ProteinSimilaritySearch"/>
  ……
</process:AtomicProcess>
```

Figure 4. Part of the OWL-S document for describing Blastp and FASTA.

4.3. Service workflow

A workflow is specified as an OWL-S composite process. A composite process contains several simple or composite processes and provides control constructs. Note that a composite process must not contain any atomic process unless the atomic process has been described by the simple process because the workflow specification is abstract. The abstract workflow specification becomes a concrete workflow if the simple processes in the specification are realized by atomic processes. A workflow can specify that its constituent processes must run sequentially or in parallel, or specify complicated control constructs such as repeat-until loops can. For example, Blastp and FASTA can be run simultaneously in the workflow of SARS analysis. The workflow also involves a loop, because an open reading frame (which is a nucleotide sequence) should be translated to an amino acid sequence, which is then sent to Blastp and FASTA and optionally to Pfam and other tools before the next open reading frame can be processed.

The workflow for the SARS analysis is currently specified manually; nevertheless it still allows a certain degree of flexibility. First, the agent can make decisions at some points of the workflow. For example, it can decide how to combine and interpret the results

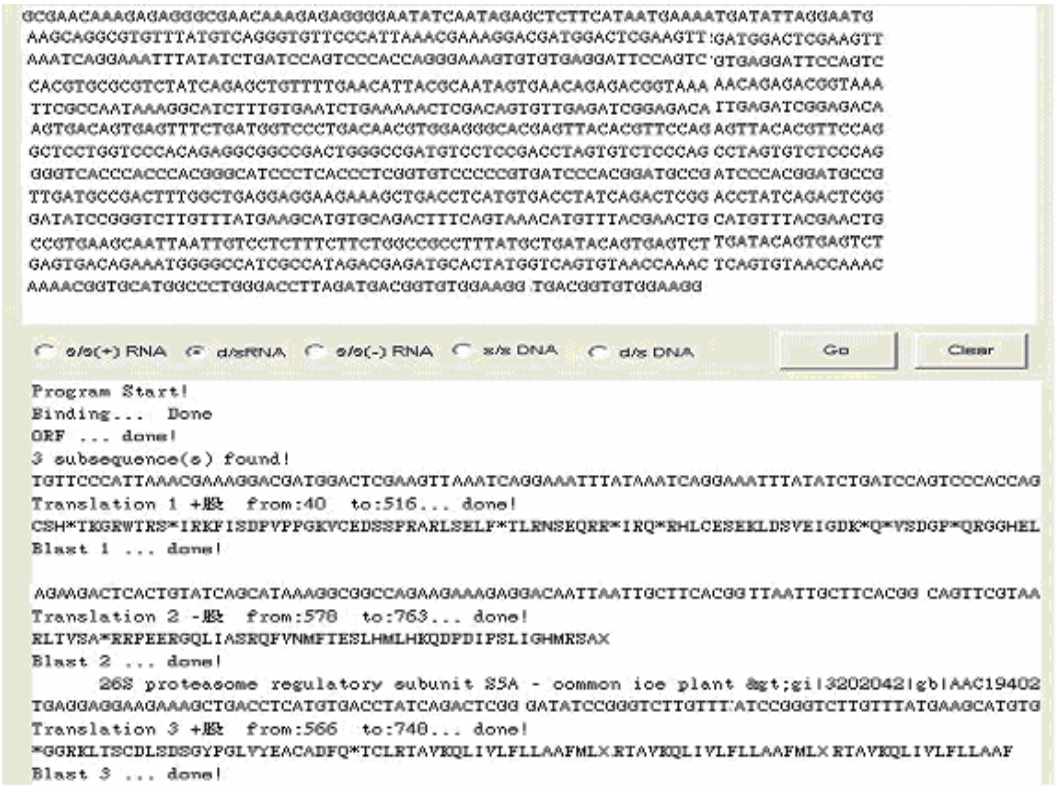

Figure 5. The inputs and outputs of the partial workflow.

returned from Blastp and FASTA. Second, if a goal is only a subgoal of the super-goal then the processing of subgoal of the analysis in [1] can be mapped into a partial workflow. For example, the agent system can bypass Blast and FASTA and proceed to run TMPred and TMHMM directly if the biologist only wants to find out whether the DNA sequence to be analyzed has a subsequence that can be translated to a protein that has membrane properties.

5. Implementation status

We have wrapped all nine genome sequence analysis tools in Table 1 as Web services. The generic method to wrap a tool involves three steps.

Step 1: Write a Web client program to access the tool on the tool website. The client program automatically fills the forms on the tool webpage, submit the form through HTTP Get or Post, and get the results returned through HTTP Put protocol.

Step 2: Write a parser program that parses the results returned from the tool website to extract useful data and store the data structurally.

Step 3: Wrap the Web client and the parser as a Web service and deploy the Web service to Apache Tomcat servlet container.

The potential problem of this wrapping approach is that the input/output format and even the URL of tools can change. Among the nine tools, Blastp offers XML-formatted output which facilitates programmatic access. Other tools output the results in HTML; that means the parser must change accordingly if the layout of the output HTML changes. Fortunately, writing the parser is not difficult. The advantage of the wrapping approach is that the websites update and maintain the tools and databases regularly, and therefore we don't need to maintain a copy of the tools on our server.

We have also composed ORF finder, Transeq and Blastp Web services into a partial workflow. Given an input DNA sequence, the workflow outputs the top three proteins that are similar to the amino acid sequences translated from the input. Figure 5 shows the user interface. The input sequence may be either a DNA sequence or an RNA sequence translated to a DNA sequence because Genbank stores all nucleotide sequences in the form of DNA. Therefore, the user must specify whether the sequence comes from a DNA virus, an RNA+ virus or an RNA- virus. Since ORF finder does not recognize whether the input sequence is originated from a DNA or RNA, the parser inside the ORF finder Web service must interpret the result according to the viral type. If it is

an RNA+ virus, the three antisense strands are filtered. If it is an RNA- virus, the three sense strands are filtered, and the antisense strands are reversed and complemented before returned to the user. If it is a double strand virus, all six strands returned from ORF finder website is returned to the user with three antisense strands reversed and complemented. Each resulting open reading frame is then sent one by one to the Transeq Web service to be translated as an amino acid sequence, which is then passed to the Blastp Web service. The output shows the name of the matching proteins and their score, computed by Blastp. We have verified that the output is the same as the result gathered from manual operation.

We have written OWL-S descriptions for the ORF, Blastp and Transeq Web services. We have also developed a supporting genome ontology. The ontology can help the Web services processing user request. For example, if the user does not know that SARS Coronavirus is an RNA+ virus, the sequence search engine shown in Figure 1 can use the ontology to infer the fact, because SARS Coronavirus an instance of Coronavirus which is a subclass of RNA+ virus.

6. Related work

There are some research works that use agents to manage workflow for the bioinformatics domain. Corradini et al. [8] propose an agent-based approach to integrate bioinformatic tools. The system is composed of three layers. The top layer is called User Layer, which provides interfaces allowing users to specify user-level workflow using UML Notes. The middle layer, called the System Layer, has a System Compiler that parses the user-level workflow and decomposes it into several agent-level workflows via mapping techniques and databases that contain correlations between user-level and agent-level workflows specifications. The System Compiler also creates agents that monitor the workflows. The bottom layer called Run-Time Layer, executes all activities that are specified in the agent-level workflow by triggering corresponding agents. Although the tool integration approach of [8] has the merits of being modular and reusable, it lacks several desirable properties of our approach. First, its agents are automatically generated and thus cannot make complex decisions during workflow execution. Second, it does not adopt an open and interoperable standard such as web service standards, which enable dynamic sharing and composition of tools. In this aspect, the myGrid project [9] is an ongoing effort to establish a computation infrastructure for bioinformatics based on Web service and Grid computing technologies.

7. Conclusion and future work

Since genome sequence analysis often requires significant human efforts, a system that automates the analysis is desirable. Users of such a system do not need to manually operate each tool, nor do they need to know the details about tools. Instead of simply connecting the sequence processing tools with a program for a particular analysis, an agent-oriented approach is taken. Each sequence processing tool is wrapped as a Web service and described with agent-readable semantic annotations.

Compared to hard-coding the workflow in a program, this approach has several merits of being open and flexible. First, service descriptions allow the experimenter agent to dynamically choose the most appropriate tools for the analysis. Second, new tools can be easily added to the system if they are wrapped as Web services. Third, users are allowed to assign different processing goals to the experimenter agent who will then execute the part of the workflow needed for the goal. This paper has demonstrated the feasibility of integrated web service technologies and has successfully applied the technologies in the real tasks of automating the analysis of genomic sequences. We are currently investigating the possibility of automatically and dynamically composing a workflow according to the biologist's need.

Currently, the Web services are wrappers of web-based tools on well-known servers such as the NCBI server. Since these web servers are accessed simultaneously by biologists and other researchers, the performance of our system cannot be controlled. Nevertheless, we are looking into workflow scheduling techniques that balance the workloads of the web services while satisfying users' workflows.

7. References

[1] Marco A. Marra, Steven J. M. Jones, Caroline R. Astell, Robert A. Holt, Angela Brooks-Wilson, Yaron S. N. Butterfield, Jaswinder Khattra, Jennifer K. Asano, Sarah A. Barber, Susanna Y. Chan, Alison Cloutier, Shaun M. Coughlin, Doug Freeman, Noreen Girn, Obi L. Griffith, Stephen R. Leach, Michael Mayo, Helen McDonald, Stephen B. Montgomery, Pawan K. Pandoh, Anca S. Petrescu, A. Gordon Robertson, Jacqueline E. Schein, Asim Siddiqui, Duane E. Smailus, Jeff M. Stott, George S. Yang, Francis Plummer, Anton Andonov, Harvey Artsob,

Nathalie Bastien, Kathy Bernard, Timothy F. Booth, Donnie Bowness, Martin Czub, Michael Drebot, Lisa Fernando, Ramon Flick, Michael Garbutt, Michael Gray, Allen Grolla, Steven Jones, Heinz Feldmann, Adrienne Meyers, Amin Kabani, Yan Li, Susan Normand, Ute Stroher, Graham A. Tipples, Shaun Tyler, Robert Vogrig, Diane Ward, Brynn Watson, Robert C. Brunham, Mel Krajden, Martin Petric, Danuta M. Skowronski, Chris Upton, and Rachel L. Roper. The Genome Sequence of the SARS-Associated Coronavirus. *Science* 300: 1399-1404, 2003.

[2] Paul A. Rota, M. Steven Oberste, Stephan S. Monroe, W. Allan Nix, Ray Campagnoli, Joseph P. Icenogle, Silvia Peranda, Bettina Bankamp, Kaija Maher, Min-hsin Chen, Suxiong Tong, Azaibi Tamin, Luis Lowe, Michael Frace, Joseph L. DeRisi, Qi Chen, David Wang, Dean D. Erdman, Teresa C. T. Peret, Cara Burns, Thomas G. Ksiazek, Pierre E. Rollin, Anthony Sanchez, Stephanie Liffick, Brian Holloway, Josef Limor, Karen McCaustland, Melissa Olsen-Rasmussen, Ron Fouchier, Stephan G ther, Albert D. M. E. Osterhaus, Christian Drosten, Mark A. Pallansch, Larry J. Anderson, and William J. Bellini. Characterization of a Novel Coronavirus Associated with Severe Acute Respiratory Syndrome. *Science* 300: 1394-1399, 2003.

[3] W3C homepage for Web services. http://www.w3.org/2002/ws/

[4] The OWL Services Coalition. OWL-S: Semantic Markup for Web Services (Technical white paper). http://www.daml.org/services/owl-s/1.0/owl-s.html, 2003.

[5] W3C Web Ontology Working Group. http://www.w3.org/2001/sw/WebOnt/.

[6] Web Service Description Language. http://www.w3.org/2002/ws/desc/.

[7] Mikko Laukkanen and Heikki Helin. Composing Workflows of Semantic Web Services. *Proceedings of the AAMAS'2003 Workshop on Web Services and Agent-based Engineering*.

[8] F.Corradini, L.Mariani and E. Merelli, An agent-based layered middleware as tool integration, *Proceedings of the ESEC-Tool Integration Workshop*, Finland, 2003.

[9] L. Moreau, S. Miles, C.A. Goble, M. Greenwood, V. Dialani, M. Addis, N. Alpdemir, R. Cawley, D. De Roure, J. Ferris, R. Gaizauskas, K. Glover, C. Greenhalgh, M. Greenwood, P. Li, X. Liu, P. Lord, M. Luck, D. Marvin, T. Oinn, N. Paton, S. Pettifer, M. Radenkovic, A. Roberts, A. Robinson, T. Rodden, M. Senger, N. Sharman, R. Stevens, B. Warboys, P. Watson, and C. Wroe. On the Use of Agents in a BioInformatics Grid. *Network Tools and Applications in Biology* (NETTAB'2002) - Agents in Bioinformatics, Bologna, Italy, July 2002.