Christopher Weir Maier. CORE576: An Exploration of the Ultra-Structure Notational System for Systems Biology Research. A Master's Paper for the M.S. in I.S. degree. April, 2006. 92 pages. Advisor: Bradley Hemminger

Tools for managing and interacting with biological data must be able to cope with the dynamic, complex, and information-rich nature of biological research. The Ultra-Structure theory, developed by Jeffrey Long, proposes a new notational paradigm of rules that is specifically designed to cope with complex systems. The approach, which has been applied successfully to the intricacies of business management, may prove useful in the context of biological information systems.

A prototype of an Ultra-Structure system for biology, dubbed CORE576, was developed in Java and PostgreSQL to explore this proposition in the context of a operating mass spectrometry systems biology laboratory. Examples of the system at work are given, and future research directions are given.

Headings:

Notational Systems – Ultra-Structure

Information Systems – Design

Databases – Biological

Bioinformatics – Systems Biology

Bioinformatics – Proteomics

CORE576: An Exploration of the Ultra-Structure Notational System for Systems Biology Research

by Christopher Weir Maier

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science

Chapel Hill, North Carolina
April, 2006

Approved by:

Bradley Hemminger

Contents

Acknowledgments

I would like to thank Morgan Giddings for the privilege of working with her lab, for introducing me to the fascinating world of mass spectrometry, and for all her helpful suggestions over the course of this work. I would also like to thank Brad Hemminger for being a great teacher and advisor. Many thanks are especially due to Jeff Long for his patient guidance in helping me learn the concepts of his Ultra-Structure system.

Most importantly, I'd like to thank my wife Dee for all the love and support she has provided, particularly over these past two years of graduate school. Whatever success I have had has been in no small part due to her.

Chapter 1

Introduction and Motivation

1.1 Biology as an Information Science

Biology is rapidly transforming itself into an information-based science. In its beginnings, biology was largely an exercise in cataloguing and classification. With the discovery of the genetic basis of inheritance by Mendel, biology became an experimental pursuit. The elucidation of the role of DNA in genetics ushered in the age of molecular biology, with researchers dissecting biological pathways to discern the function of their components. Researchers could spend months and years determining functions of single proteins, but modern biology does not have this luxury. Advances in instrumentation and analysis technologies, as well as computational techniques, have opened up new avenues of investigation for the biologist (Hood 2002) by generating an ever-increasing amount of detailed biological information.

Modern biological research is quite familiar with the "information overload" commonly pointed to by information scientists in recent years. Medical text mining experts regularly tout the exponentially increasing number of research articles indexed by Medline, just as genetics researchers tout a similarly exponential growth in the number of sequenced genes in GenBank. The completion of the draft human genome in 2001 heralded the entry into a "systems biology" era, in which complex

experiments capable of examining broad aspects of cellular function (e.g. analyzing the expression of all of a cell's genes, as opposed to only a handful) are commonplace. The thought of a single experiment generating hundreds of megabytes of data, let alone gigabytes, while unthinkable in years past, is rather unsurprising today. No human can effectively assimilate and integrate such a staggering volume of data without assistance from computerized systems. Thus, an important line of work in the research community centers on devising solutions to the problems this glut of information presents. Research articles describing the creation of task- and research-specific databases, as well as analysis packages and evolving community-driven data standards, are frequently seen. In order to be of any use, biological data must be stored and organized in such a way that it can be easily accessed and queried; information that cannot be accessed may as well not exist.

1.2 Project Context

This work was performed in the systems biology laboratory of Morgan Giddings in the Department of Microbiology and Immunology at UNC. Her lab carries out both "dry lab" computational research centering on algorithm design, modeling, and data analysis, as well as traditional "wet lab" investigations focusing on understanding and elucidating mechanisms underlying the evolution of antibiotic resistance in bacteria.

Specifically, members of the Giddings lab are currently looking at mechanisms of resistance in the bacteria E. coli to the antibiotic streptomycin. They do this by comparing unmutated bacteria (called "wild type" or "WT") to bacteria that have been grown in conditions that select for streptomycin resistance (dubbed "SmR"). Additionally, a third bacterial strain, "SmRC", is used. Generally, mutated bacteria grow at a reduced rate when compared to their unmutated counterparts; SmRC is a strain of the SmR mutant that has been selected for increased or "compensated"

growth. As streptomycin interferes with protein synthesis, mutations in the protein components of the ribosomal complex (which is responsible for protein synthesis) impart resistance to the drug. Thus, by comparing ribosomal proteins from each of these three bacterial strains, the lab aims to determine the nature of the mutations that give rise to streptomycin resistance. The hope is that this knowledge will shed light on antibacterial resistance mechanisms in general.

Understanding the mutations amounts to characterizing the proteins of the ribosomal complex, which entails a cataloging of any mutations, truncations, and post-translational modifications that have taken place; the essential question being asked is "what makes the mutated proteins different from their normal counterparts?" To answer this question the lab takes a mass spectrometry (MS)-based proteomics approach (see Appendix B for a broad overview of techniques and terminology). By combining data from so-called "top-down" and "bottom-up" MS techniques, a more complete and accurate characterization of proteins in their cellular milieu is possible.

The power of this approach has been demonstrated by the Giddings lab and collaborators at the Oak Ridge National Laboratory (VerBerkmoes et al. 2002; Strader et al. 2004; Connelly et al. 2006). However, like many of the "-omics" approaches in modern biological research, this approach generates large amounts of data. Researchers would like to have an automated way to analyze the information, but no such system exists; data is compiled and cross-referenced manually through the use of spreadsheets. Also contributing to the manual nature of the task is the fact that the outputs of many analysis programs used by the lab are in a variety of formats that are not immediately machine-readable; HTML, plain text, and even Excel spreadsheets. This has many drawbacks, as one might expect. Being a manual process, it is tedious and error-prone. The data is multidimensional and does not always fit nicely into the tabular format dictated by the spreadsheet model, which hinders complex analysis. With each new experiment, all previous data should ideally be checked for

correlations; as the data grows, this becomes quite a daunting task. The addition of a new research technique to the laboratory's repertoire is another complicating factor; even if there was an information system in place, the chances that it could easily incorporate new data for which it was not designed without significant overhead in terms of both data model and software redesign are slim indeed. Thus, the information problem faced by the Giddings lab is two-fold; the information must be better organized, taken from a collection of files and put into some form of organized and flexible data repostiory, and new tools to facilitate the analysis of these data in an automated way must be developed.

This is not a problem that is limited to the Giddings laboratory, however; it is a situation that must be dealt with by all systems biology research groups. Since systems biology encompasses a multitude of research approaches and techniques, and since integration of information from these various techniques is an ultimate goal for the research community, a versatile and general information system that can be used by investigators regardless of their particular research approach would be an ideal solution. Naturally, this is a challenging proposition, given the diversity and complexity of the overall research endeavor. However, the Ultra-Structure notation for complex systems, developed by Jeffrey Long, has the potential to provide a solution. To this end, the Giddings lab, in collaboration with Bradley Hemminger of the UNC School of Information and Library Science, are pursuing the creation of an Ultra-Structure system for biological research. Initial exploratory efforts, undertaken with the assistance of Long, resulted in a small prototype system, implemented in Microsoft Access and Visual Basic. This work describes additional development efforts for the system.

Chapter 2

Ultra-Structure

2.1 Introduction

Our knowledge of biology is pushing at the limits of reductionist science. The complexity we see in biological systems creates obstacles to further progress. One such obstacle concerns the computer systems we use in biological research. The problems associated with software engineering are well-known (Brooks Jr. 1995). To be robust, computer systems must be able to cope gracefully with change. Object-oriented development, for example, allows for the development of specialized, orthologous software components that can be assembled into working systems. However, maintenance in the face of changing requirements can still be daunting. When changing requirements fall into the realm of domain knowledge, updates can be particularly problematic, as they require both software experts and domain experts. To take an example from the business world, imagine the problem of updating a complex financial package following a massive rewrite of the tax code.

The Ultra-Structure approach addresses this problem by representing data and algorithms as "rules" stored in a database. These rules are formally defined, human-readable rules that system operators are able to modify, in order to change how the system operates. The only traditional software components are the so-called

"animation procedures," which are general purpose software methods that operate in a general way on these rules in order to generate the behavior of the system. Because the animation procedures are general, they should not need to be changed to introduce new behavior and functionality; this is achieved through the addition of rules to the system.

These rules are conceived as a new kind of notation system. Notation concerns the symbol languages humans have developed to convey information. In mathematics, we have various symbols, such as $\int$, $\sum$, π, $\sqrt{}$, and ∂, not to mention numbers themselves, all of which have very specific meanings, and can be chained together to create "sentences" in the mathematical language. In chemistry, we refer to molecules with formulæ such as C_6H_6, $C_6H_{12}O_6$, and the like, as well as via structural diagrams. In music we have the staff notation, chord symbols, and tablature. Other notations exist as well, including money and time. Written language itself is a notational device, one which has been developed in many ways by many cultures throughout history.

New notational systems arise to solve some problem, creating abstractions that enable us to manipulate systems more efficiently. For example, consider the move from Roman numerals to Arabic, which introduced the abstraction of "zero"; this simple concept revolutionized math and science. In psycholinguistics, the Sapir-Whorf hypothesis states that the language that people use influences how they see the world. Jeffrey Long takes a similar stand in asserting that our notational models determine the advances that can be made in various pursuits from science to the arts; how far could science and mathematics have progressed before the idea of "zero"? According to Long, our current notational devices are ill-suited to dealing with complex and dynamic systems, which accounts for a large part of our difficulty in truly understanding and exerting control over such systems (Long 1999a). In response to this, he has developed the new notational system of Ultra-Structure, which is specifically designed to cope with this kind of complexity.

The best introduction to Ultra-Structure is the seminal ACM paper from 1995 (Long and Denning 1995); interested readers are strongly urged to give this paper a thorough reading. Apart from this reference, however, there is only a very small body of available literature on Ultra-Structure (Long 1999b; Shostko 1999; Overgard 1999). Long is writing a book on Ultra-Structure, but it is not yet finished (Long 2005). Since the approach is relatively unknown, and this project is fully built on its ideas, a review will be given here, drawn from these resources, as well as the experience of working with the CORE576 system.

2.2 The Power of Rules

Ultra-Structure is a rule-based notation and information system model that grew out of ideas based on Chomsky's transformational grammars. Ultra-Structure takes the position that all complex systems are the result of the interactions of processes, and that regardless of their complexity these processes can be described using relatively simple rules. This idea of emergence is echoed by systems theory researchers (Weinberg 2001; Laszlo 1996) and can be seen in Conway's famous Game of Life, wherein surprisingly complex and life-like behaviors come about in a simulation of artificial lifeforms based on only three simple rules (Gardner 1970). In Ultra-Structure, rules can interact with each other in a variety of ways, triggering different behavior based on the context in which they are processed. Rules can be defined in a hierarchical manner, meaning that individual rules explicitly covering each contingency need not be created, as more general rules can subsume these specific cases. This economy of system specification can reveal the higher level essential components of the system in a way that other representations may not be able. Knowledge of these essential components is a prerequisite for true system mastery and understanding, and may also reveal hidden connections between seemingly different systems.

In Ultra-Structure, rules have a canonical form consisting of a series of one or more *factors* and a series of zero or more *considerations*. Factors specify conditions that must be met in order for a rule to be examined during processing, and considerations can specify actions that may be taken. To a first-degree approximation, factors can be thought of as forming the antecedent of an if-then statement, with considerations forming the consequent; this analogy is only approximate, however. If input to the system matches the factors for a particular rule, the action implied by the considerations is not *necessarily* carried out. They are called "considerations" because many rules may have factors that match a given input (depending on various context-dependent transformations that may subsequently take place; see Section 2.6). The system then "considers" the right-hand sides of the matching rules to determine the course of action. For instance, the considerations of one rule may be overridden or modified by another matching rule. Actions and scenarios denoted in considerations are not guaranteed to occur; rather, they are acceptable possible actions whose executions are contingent on the overall state of the system's rule base.

2.3 Deep, Middle, and Surface Structures

Ultra-Structure organizes the rules that generate the complex behaviors of a system in a hierarchical fashion. This hierarchy is three-tiered, consisting of the deep, middle, and surface structures, each level being built on top of the ones preceding it. These levels provide a helpful and straightforward way of thinking about complex systems, and offer a means by which to link related systems together.

Ultra-Structure theory postulates that all members of a given class of systems will share a common underlying structure. For example, while basketball and chess are clearly quite different kinds of games, they are at their core "Games," and as such share some fundamental organization that is responsible for their "game-ness."

Similarly, all businesses, be they large multinational corporations or local family-owned hardware stores, share features by virtue of their membership in the class of "Businesses." In Ultra-Structure these class-wide similarities must somehow relate to rules; they are the *kinds* of rules that appear in the systems of a given class. The deep structure of a class of systems is thus defined as the formal specification of families of rules, as well as general software methods that operate on these rule families.

It is not uncommon in biological research that databases and software be tailored to a specific research organism, technique, or research regime. For instance, researchers studying the *Escherichia coli* bacterium will likely have different information storage and processing needs than those studying other model organisms, such as the yeast *Saccharomyces cerevisiae*, the nematode *Caenorhabditis elegans*, or the mouse. Similarly, researchers using microarray technology must deal with different information infrastructures than those utilizing mass spectrometry. A survey of scholarly bioinformatics journals regularly turns up application notes for such specialized systems (Prickett et al. 2006; Mao et al. 2005; Jacques et al. 2005). Integration of data across this variety of resources and formats is a complicated affair, and is an active area of research (Baker et al. 1999; Wilkinson and Links 2002). These specific systems all have their own unique structure; the database schema for one project is generally not able to be replaced with the schema from another and still be expected to work, for example. If a combined schema were to be developed to store information from several different research approaches, it would likely become quite large and cumbersome, making it difficult to query and modify. Some combined schemata have been developed (see for example Section 4.5), but for relatively similar data sets. In any event, with the pace at which new research techniques and projects come into being, any effort to consolidate data from markedly different sources would quickly run into insurmountable complexity barriers. The question that Ultra-Structure seeks to answer is "Is there, in some sense, a 'universal schema' that can be used for these

different systems?" The conjecture is that yes, there is, and it is based on the fact that all these bioinformatics resources exist within the common realm of biological research. It is the deep structure of biological research.

In practice, since Ultra-Structure systems are generally implemented in the context of a relational database, the rule definiton component of the deep structure is composed of a set of tables which specify the definitions of the rules on which the system operates. These tables are known as *ruleforms*, and are discussed more fully in Section 2.4. Additionally, *animation procedures*, the software components of the deep structure, are discussed in Section 2.7.

The middle structure of a system consists of all the rules that define a specific system. To continue with the games analogy, baseball and chess are both "Games," and so share the same deep structure; however, they are very different games, governed by different rules. All the Ultra-Structure rules that describe the various rules and regulations of baseball ("three strikes and you're out," "a game consists of nine innings," "bases are located 90 feet apart,") and chess ("rooks may move any number of spaces horizontally or vertically," "pawns that make it across the board are promoted," "a requirement for castling is that the king and the rook involved have not yet moved") comprise the middle structure. While the middle structure can contain these "rulebook" rules, it can also contain other types of higher-level rules, such as those that govern strategy formation. In practical terms, the middle structure consists of the contents of the database tables that define the rules.

The surface structure is the easiest to grasp, as it is the readily seen manifestation of the system. Continuing with the example of games, the surface structure of baseball and chess can be seen in the playing of a game of baseball or chess. The unfolding of the final game of the World Series, or the endgame played between two Grandmasters comes about through the application, observance, and utilization of

the particular rules in the middle structure. As such, the surface structure is not explicitly stored anywhere in an Ultra-Structure system, but arises out of the use of that system, from the animation and interaction of the rules that define it.

2.4 Ruleforms

As a notational system, Ultra-Structure offers the idea of ruleforms, which are collections or classes of formally equivalent rules. In other words, the rules in a ruleform all share the same structure. Ruleforms are generally presented in a tabular form, the structure of which is dictated by the ruleform. The contents of the table are the rules, one per row. When implemented in a working system, a ruleform is conveniently described by a relational database table, with each tuple of the table representing a single rule. The primary and foreign key integrity constraints of modern relational database systems are also beneficial, and allow for rapid matching and selection of rules; a primary key consists of a ruleform's factors, and foreign keys allow linking of information in one ruleform to another.

It is thought that complex systems can be described by relatively small numbers of ruleforms; there may be many thousands of rules needed to full describe any given system, but these rules will fall into a small number of ruleforms, generally less than 50, based on experience (Long and Denning 1995). The utility of ruleforms lies in the fact that all rules that belong to a ruleform are formally equivalent, meaning they can be operated on in an equivalent manner. This allows semantically similar information to be treated similarly, a situation that may not occur in more traditional information modeling approaches. An illustrative example concerns the notion of Locations (Long and Denning 1995). In a traditional relational database approach, items as disparate as phone numbers, email addresses, and physical locations (of, say, items in a warehouse inventory) would occupy separate tables. In an Ultra-Structure

system, these data might all be viewed semantically as being *Locations*; an email address is a person's location in "email space", whereas a phone number is a location in "phone space". Treating semantically similar items in the same manner is a powerful idea, and can help to reveal the underlying nature or meaning of the system. The full version of this idea, called the Ruleform Hypothesis, is quoted from (Long and Denning 1995):

> **Ruleform Hypothesis.** *Complex system structures and behaviors are generated by not-necessarily-complex processes; these processes are generated by the animation of operating rules. Operating rules can be grouped into a small number of classes, whose form is prescribed by ruleforms. While the operating rules of a system change over time, the ruleforms remain constant. A well-designed collection of ruleforms can anticipate all logically possible operating rules that might apply to the system and constitutes the deep structure of the system.*

A corollary to this is known as the CORE Hypothesis:

> **CORE Hypothesis.** *There exist complex operating rule engines, or COREs, consisting of $\leq$ 50 ruleforms, that are sufficient to represent all rules found among systems sharing broad family resemblances, for example, all corporations. Their definitive deep structure will be permanent, unchanging, and robust for all members of the family, whose differences in manifest structures and behaviors will be represented entirely as differences in operating rules. The animation procedures for each engine will be relatively simple compared to current applications, requiring less than 100,000 lines of code in a third-generation language.*

Clearly, the Ultra-Structure approach carries with it significant philosophical considerations that relational databases (i.e. those designed with traditional entity-relation modeling approaches), for example, simply do not have. This is in fact one of the motivations for the current work; the discovery of a stable underlying structure to biological information would be of incredible use to researchers. It could unify diverse research fields and facilitate discoveries at the interfaces of these fields, discoveries that are difficult if not impossible to make with current information practices.

While each Ultra-Structure CORE will likely have a number of distinct ruleforms, reflecting the unique features of the class of systems it represents, there exist several general kinds of ruleforms that commonly appear in Ultra-Structure systems. Each has a characteristic structure and usage pattern. These are described below.

2.4.1 Existential Ruleforms

Ruleforms with one factor are known as existential ruleforms. Rules of this form declare the existence of some entity of interest. An example is shown in Table A.1. Several existential ruleforms may, and generally do, exist in an Ultra-Structure system, specifying all the different kinds of entities that the system concerns itself with. For example, the business-oriented CORE650 system (see Section 2.8) has existential ruleforms for Products, Locations, Agencies, among others, whereas the CORE576 system includes BioEntities, Resources, Attributes, and BioEvents.

The single factor of existential ruleforms specifies a unique name for the entity. While existential ruleforms (and ruleforms in general) are not required to have considerations, existential ruleforms generally do specify some number of considerations that will contain additional information about an entity, as well as provide metadata about individual rules, such as when the rule was last updated, and by whom.

In nearly every existential ruleform, there will exist a few "special" entities that each merit additional discussion. The first, referred to as "Top Node", is used in Network ruleforms, which are discussed in the next section. The other special entities are generally named "ORIGINAL" and "ANY". These will come into play in so-called Metarules, which will be discussed further in Section 2.6.

An idea of the kinds of entities that are found in an existential ruleform of the CORE576 system is discussed in Section 3.3.1.

2.4.2 Network Ruleforms

Ruleforms whose factors refer to two existential rules, as well as a relation code (known in Ultra-Structure parlance as a *relcode*) are network ruleforms (see Table A.2 for examples). Rules in these ruleforms define semantic networks, linking the two entities (nodes) with the directed, labeled edge specified by the relcode. Relcodes can define many kinds of relationships: taxonomic relationships are accomplished with relcodes such as "IS-A" and "INCLUDES"; associative relationships can also be formed, linking objects in different taxonomic branches, creating a network. These networks are also used to create groupings and classes of similar objects. In supplying this grouping and classification facility, Network ruleforms supply much of the power of the Ultra-Structure approach. The information contained in Network ruleforms can also be used to logically deduce new rules based on these grouping and classification features; see Section 2.5.

The basic Network rule takes the form of <***Parent Relcode Child***> where *Parent* and *Child* denote the two entities being related, and *Relcode* is the name of the relationship; for example <***Serine IS-A Amino Acid***>. The factors of a Network ruleform are *Parent* and *Relcode*, while *Child* will be included with the considerations. In addition to *Child*, other considerations of note include *Is_Original*. This consideration is used in network propagation (discussed in Section 2.5), and has the value of "TRUE" for rules that have been entered by a user or data import process, and "FALSE" for deduced rules. If an entity can participate in several relationships of the same type, then an additional factor, called a *Sequence Number*, can be used to distinguish between instances (and maintain primary key constraints when implemented in a relational database system). The ruleform must also be augmented if the relationship depends on some other factor or factors, such as time.

As mentioned above, the "Top Node" entity plays an important role in Net-

work ruleforms. This special entity is conventionally used in Ultra-Structure systems to denote the archetypal, most basic primitive entity (or superclass, if you will) of an existential ruleform. It is thus utilized in Network ruleforms as a "root" of the semantic network, an entity that all other entities are eventually connected to. "Top Node" serves as a convenient entry point into Network ruleforms. While this entity is conventionally named "Top Node", it can be called anything. System designers may want to name it after the containing existential ruleform to yield rules that "read" easier; certainly the rule <***Bacteria IS-A BioEntity***> is more readable and comprehensible than <***Bacteria IS-A Top Node***>. If a particular existential ruleform has no accompanying Network ruleform, a "Top Node" entity is not needed.

2.4.3 Relcodes

A relcode (short for "relationship code") functions as the name of one of the various relationships that may exist between entities in the system; they can be thought of as the labels on the edges of the Network ruleform semantic networks. Relcodes are defined in their own existential ruleform, the considerations of which help to define the behavior of the relcode. Important considerations that appear to have a place in Relcode ruleforms in general (i.e. regardless of the particular CORE they appear in), include *Contra*, *Is_Transitive*, and *Is_Preferred*.

The *Contra* consideration defines the relcode to use in the "opposite" relationship. An example will clarify: if the *Contra* of "IS-A" is "INCLUDES", then from the rule <***Serine IS-A Amino Acid***>, we can deduce that <***Amino Acid INCLUDES Serine***>. If "IS-A" has "INCLUDES" as its *Contra*, then "INCLUDES" must have "IS-A" as its *Contra*; to be otherwise would result in erroneous deductions. Note that a relcode may have itself as its own *Contra*. Refer to Section 2.5.1 for more discussion on the use of *Contra*.

The *Is_Preferred* consideration is a boolean flag that indicates, among the (at most) two relcodes linked through the *Contra* consideration, which one is to be used for original rules. In other words, network rules that are input by users or data import processes should be in the form dictated by the preferred relationship. Therefore, any rules that use the non-preferred directionality will be deduced rules (though some deduced rules will use the preferred form). Such a distinction will be useful in future graphical user interfaces, presenting users with only valid choices when creating new rules. Additionally, the current network propagation algorithm utilizes this flag in its processing; see Section 2.5.5.

The *Is_Transitive* consideration, also a boolean flag, indicates (appropriately enough) whether a relationship is transitive in nature. This, of course, is essential to the network propagation algorithm; see Section 2.5.

Relcodes themselves can even be organized into networks, as has been done in the CORE650 Ultra-Structure system, and may be done in the CORE576 system (see Section 5.3).

Just as existential ruleforms have "special" entries, so to does the Relcodes ruleform. Its special entry, dubbed "SAME", plays a special role in MetaProtocol rules, which are discussed in Section 2.6.

2.5 Inference on Networks

Network ruleforms, establishing semantic networks linking entities in various ways, provide a rich knowledgebase from which to make logical inferences. These inferences serve at least two purposes. First, they allow the system user to enter a relatively small number of rules — establishing a "skeleton" network, so to speak — and then automatically "fill in" the rest of the information. One does not need to enter a rules

that state that <***Serine IS-A Amino Acid***> and <***Serine IS-A Molecule***> (as well as similar rules for alanine, proline, glycine,...); all that needs be entered is <***Serine IS-A Amino Acid***> and <***Amino Acid IS-A Molecule***>, and logical inference takes care of the rest. Network inference can also be useful in checking the validity and consistency of entered rules; unexpectedly deduced rules may indicate incorrectly entered data. Alternatively, unexpected links may signal previously unknown information. These links may be unknown because they are heretofore unconsidered, or simply because they are lost in an overwhelming volume of information. It is hoped that this system can help address these last concerns.

A collection of animation procedures, dubbed the Network Propagation Engine (NPE) was written to perform this inference on Network ruleforms. There are several forms of inference that these procedures carry out, each of which will be described in turn.

2.5.1 Contra Relationships

Every rule in an Ultra-Structure Network ruleform has at least one additional rule that can be inferred from it. When given a rule stating <***Serine IS-A Amino Acid***>, we can immediately deduce that <***Amino Acid INCLUDES Serine***>; that is, the notion of "Amino Acid" includes the entity "Serine". In order to formally deduce this, the information needed (apart that contained in the rule itself) is the name of the relationship that is the "opposite" of the "IS-A" relationship. This information, is contained as a consideration (the *Contra*) in the rule that defines the "IS-A" relcode. Thus, the animation procedure responsible for inferring these contra relationships simply inspects the relcode ruleform to determine the appropriate contra relationship and then uses that to create a new rule in which the left hand side and right hand side of the original rule are interchanged.

2.5.2 Transitive Deductions

The next form of deduction is a straightforward deduction on the rules of a Network ruleform that takes advantage of the transitive nature of certain relationship types. If two network rules exist, such that both have the same relcode, and that the right hand side of one rule is the same as the left hand side of the second, then a new rule can be deduced, linking the left hand side of the first rule with the right hand side of the second. More simply, if <***A EQUALS B***>, and <***B EQUALS C***>, then <***A EQUALS C***>. This deduction is only possible if the "EQUALS" relationship is transitive, which of course it is. Again, to make this deduction possible from a formal computation point of view, the only information outside of the two rules that is needed is whether or not the relcode in question is transitive or not. This is stored as a consideration in the *Relcodes* ruleform.

In practice, the NPE scans the Network ruleforms, considering each existing rule in turn. For each rule that has a transitive *Relcode*, additional rules are selected where the *Parent* of the second rule is the same as the *Child* of the first rule, and both rules share the same transitive *Relcode*. The new rule is then constructed as described above.

2.5.3 Property Inheritance

The final kind of deduction carried out by the NPE is, strictly speaking, a superset of the previously described transitive deductions. This form, which may be considered as a kind of "property inheritance," relaxes the restriction that the relcodes used by two rules are identical, while introducing new restrictions on the kinds of rules that will be considered as input for the deduction (see Section 2.5.5). It allows for two rules of the form <***A r1 B***> and <***B r2 C***> to be used to deduce <***A r2 C***>.

Long and Denning (1995) state that network rules "declare relationships of the form *Class A, with respect to relationship type R, is a member of Class B.*" Viewed in this light, Network ruleforms not only form semantic networks, but also define class hierarchies. Network propagation can thus implement a kind of inheritance functionality. This is not inheritance in the polymorphic, object-oriented sense of the term, however, because there is no way for children to override parents.

An example of this kind of inference can be seen in the original Ultra-Structure paper (Long and Denning 1995), dealing with a Network of "Locations." Some of the example rules given contained information such as <***37 Bret Harte Terrace CITY Washington***> (stating that the street address "37 Bret Harte Terrace" is in the city of "Washington") and <***Washington STATE D.C.***> (stating that the city of "Washington" is located in the "state" of "D.C."). Based on a property inheritance kind of inference, the rule <***37 Bret Harte Terrace STATE D.C.***> can readily be inferred, by virtue of the fact that the "CITY" and "STATE" relcodes are declared transitive. Note that the standard transitive deduction, in which both relcodes are identical, would not be able to deduce this new rule.

This situation reveals important information about the concepts of relcodes and network ruleforms. In general, as a path is traced from an entity through the network to the "Top Node", each connection has a unique name that reflects the nature of that connection. This will come into play in the discussion of animation procedures in Section 2.7.

2.5.4 Comment on Deductions

The currently implemented deduction procedures operate either on a single rule or on a pair of rules. Deductions that ultimately require more rules as input can still be performed, however; entering the rules <***A EQUALS B***>, <***B EQUALS C***>,

and <***C EQUALS D***> still allows the rule <***A EQUALS D***> to be deduced. Such a deduction proceeds stepwise, first deducing that <***A EQUALS C***>, and then coupling that rule with <***C EQUALS D***> to achieve the desired result. In practice, the various forms of deduction described are carried out on the collection of rules in a Network ruleform, with a counter keeping track of how many new rules have been deduced in a single pass; deduction continues until no new rules are deduced by any method.

This method of deduction has some issues, which may have to be addressed in the system; see Section 5.3.

2.5.5 Requirements of Manually Entered Network Rules

The described propagation techniques require that the rules that are manually entered into the Network ruleforms obey certain restrictions in order to ensure complete and accurate rule propagation. A key requirement is that these manually entered rules must describe a skeleton network; that is, the rules must describe some path from every node to the "root" of the network. Network propagation is a powerful technique, but it can only operate on the data it is given; if there is no data that can be used to establish connections, then none will be able to be made.

Another requirement is that network rules that are manually entered should use only the "preferred" relcodes. What this means in practice is that propagation of new rules will take place in one direction, either from the "root" of the network out to the perimeter, or vice versa. Currently in the CORE576 system, preferred relcodes create rules in a specific-to-general form, such as <***Serine IS-A Amino Acid***>, resulting in a deduction that proceeds from the network perimeter to the center. Rules handling the reverse navigation are generated by the contra deduction discussed in Section 2.5.1.

2.6 Protocols and Metarules

Thus far, the ruleform types described have mainly been concerned with the storage of data, yet a key feature of an Ultra-Structure system is the encoding of processes as data. This information is stored in Protocol and Metarule ruleforms. A protocol ruleform defines the various steps needed to perform some action, as well as the sequencing of those steps. When rules in this kind of ruleform are activated (by matching their factors), their considerations, defining various actions, are inspected for possible execution. Metarule ruleforms, on the other hand, contain rules on how to interpret rules. When some input comes into the system, Metarules are responsible for determining which Protocol rules should be examined to carry out some desired action or series of actions. Long and Denning (1995) present a helpful example from the CORE650 system, wherein a request for a product order (the input) is guided through a rather complex series of processing steps, defined by the contents of metarule and protocol ruleforms. What initially entered the system as a simple request to ship a product to a customer triggered several subsidiary processes, such as credit checking, discount application, billing, inventory picking, and shipping. When one considers the various constraints on ordering — the status of the customer as well as their location, to name just two — the task of writing software to appropriately handle these becomes quite difficult. By encoding these constraints as rules, the task becomes much easier. A Metarule ruleform will thus contain rules that can be used to transform system input in a variety of ways to facilitate further processing.

The factors of a Metaprotocol ruleform will define some condition under which the rule should be examined for further processing. The considerations will contain relcodes that are used to navigate relevant network ruleforms to create a "masked" input that in turn is used as a key into a Protocol ruleform. In the CORE576 system, for instance, if some processing task that took the name of a (*BioEntity*) chemical

molecule as input (say "Water") needed to know something about the molecule's polarity in order to properly guide process execution, a Metarule might contain the relcode "POLARITY". This would mean that the the *BioEntity Network* ruleform would be searched for a rule that had factors of $<$***Water POLARITY***$>$, which would find the consideration "Polar Molecule". Thus, "Water" will have been masked with "POLARITY" to become "Polar Molecule". In cases when a particular input should not be masked, but used as-is to inspect the appropriate Protocol ruleform, the special relcode "SAME" is used.

As stated earlier, Protocol ruleforms define the ordering of various processing steps in the system. The factors of these ruleforms are generally set up to reflect the "output" of their corresponding Metarule ruleforms. A mapped input set obtained from a Metarule thus forms a key by which to select rules from the Protocol ruleform. Considerations of these ruleforms then define actions to execute. In the CORE650 business system, this might entail a work order being written to a department's work queue, from which human workers would draw their tasks. Alternatively, it could trigger some autonomous computer program to perform a task, such as calculating sales tax. In the Protocol ruleforms that currently exist in the CORE576 system, triggered actions generally involve reflectively invoked Java methods.

In Section 2.4.1, the special Existential ruleform values of "ANY" and "ORIGINAL" and their utility in Protocol ruleforms were mentioned. If a particular rule needs no particular value for a given factor, then the signal value of "ANY" may be used, indicating that any value is acceptable to cause inspection of the rule. The value of "ORIGINAL" is utilized in the considerations of a Protocol rule. Values found in considerations are used to invoke software methods, and are commonly used as parameters. "ORIGINAL" indicates that the original, unmapped input value should be used for a particular consideration, instead of the value obtained following Metarule processing.

This processing is more easily grasped with an example, which is given in the context of the CORE576 system in Section 3.5.

2.7 Animation Procedures

In order for the rules defined in a system to have any kind of effects, they must be interpreted and acted upon. In Ultra-Structure, small software methods known as "animation procedures" perform this function. These methods are designed to operate in a general manner on ruleforms rather than on individual rules. In this way, the coded software may remain stable in the face of changing system requirements. By being coded to ruleforms, they will be able to properly manipulate any rules contained in those ruleforms. The purpose of animation procedures is thus to separate control logic from "world knowledge;" all information specific to the system itself is contained in rules while the animation procedures define things like which ruleforms to inspect, and in which order. As such, the animation procedures are kept general; a well-coded animation procedure should not need to be altered in order to make some specific processing possible. It is possible, however, that in the course of processing, some external software may be invoked.

Animation procedures can be coded in whatever programming language that is desired; there is nothing "special" about them from a software engineering perspective.

2.8 Previous Uses of Ultra-Structure

Long has investigated the Ultra-Structure system through the implementation of several COREs. The most mature CORE, CORE650[1], concerns business-related operations, such as ordering and billing. This CORE has been used by several companies,

to impressive effect (Long and Denning 1995). Long has implemented additional COREs for the Australian government to track shipments from the United States and Europe, and for a declassification project under the auspices of the United States Department of Energy (Jeffrey Long, personal communication). Long has also developed smaller, experimental COREs in the fields of Artificial Life, Games, Music, and Legal Systems. The current work is an extension and updating of a preliminary exploration of a Biology CORE.

Notes

[1]The number 650 is the Dewey Decimal Classification for "Management and Auxiliary Services". Similarly, 576 is the classification code for "Genetics and Evolution". This is the standard naming schema for COREs.

Chapter 3

CORE576: Ultra-Structure for Biology

The Ultra-Structure notational system offers a novel vantage point from which to think about complex systems. Certainly some of the most complex systems humans seek to understand are biological ones; millions of years of evolution have crafted intricate and elaborate networks of interacting chemicals, molecules, cells, organs, organisms, and populations. Teasing out exactly what goes on in these systems is a monumental challenge, one that biological researchers have been tackling for centuries. Managing the scientific data that comes from this research has always been difficult, but it has become particularly thorny in recent years due to the sheer volume of data our modern research approaches are capable of generating.

The CORE576 system is a work-in-progress application of the Ultra-Structure approach to modeling and managing biological information. It is hoped that Ultra-Structure can offer relief to researchers struggling to make sense of their data, and at the same time provide insights into the complexities of biology, something that perhaps only a new notational abstraction may do.

3.1 Background

The current implementation of the CORE576 system has its roots in an early prototype system designed by Jeff Long for the Giddings Lab in 2003. This system was based on Microsoft Access, chosen due to the rapid prototyping capabilities its graphical user interface provides. Ruleforms were created as relations in the database, with factors constituting primary keys, and integrity constraints to link ruleform universals (Ultra-Structure terminology for the components of a ruleform; "attributes" in relational theory) to their parent ruleforms. Animation procedures for the system were coded in Visual Basic, and user interfaces were created as Access forms. This prototype system exhibited interesting features. For instance, it was capable of simulating the translation of a protein sequence from a corresponding DNA sequence. Rules governing the translation process (e.g. "DNA is translated in codons of length three," "The codon 'ATG' encodes the amino acid 'Methionine'," etc.) were entered into the database, and animation procedures were coded that acted on these rules to carry out the process of translation. In addition to this, the system could calculate the masses of chemical compounds: rules stating the composition of a molecule, say serine, were entered, as well as rules declaring information such as the atomic weight of various biologically important elements. Animation procedures were then responsible for calculating the final mass based on these rules. An interesting aspect of the system was that molecules could be defined, not just in terms of which and how many atoms of each element they were composed of, but in terms of larger functional groups, such as amino groups (NH_3) or carboxyl groups (COOH). This enabled large compounds to be built up piece-by-piece, something that would be helpful, for example, in calculating the theoretical mass of a protein, based on its amino acid sequence.

Due to lack of resources, little work was done on CORE576 in the following

years. The contents of the database had been migrated to MySQL, and a basic web-accessible front end was created, but little work was done on the animation procedures. A few Perl scripts had been written, but they were mainly *ad hoc* single-use scripts for loading data, and not animation procedures *per se*. As such, the "definitive" version of the system remained the Access prototype. This was the state of of the CORE576 system at the beginning of the current work.

3.2 Current Implementation

The current implementation of the CORE576 system has been migrated from Microsoft Access and Visual Basic to PostgreSQL (version 8.0.3) and Java (1.5 JDK). PostgreSQL was chosen over Access due to its more advanced capabilities, its availability on a broad number of computer platforms, as well as its open source license; it was chosen over MySQL due to PostgreSQL's support for a greater subset of the SQL standard. Using Java to code the animation procedures also removes platform restrictions, as the Java Virtual Machine is widely available. The SQL used to create the database is relatively portable, but utilizes a few PostgreSQL-specific extensions for convenience, namely the extensions to the `VARCHAR` and `NUMERIC` types that eliminate the need to specify a maximum length or precision, respectively. By allowing unrestricted `VARCHAR`s, genetic and protein sequences can be stored without concern that they will be erroneously truncated (any genomic information to be stored in the system will likely utilize a special table of `CLOB` data), and unrestricted `NUMERIC` datatypes allow both integer and decimal numeric information to be stored an manipulated exactly. This simplifies data storage and also prevents errors being introduced in the data import and export processes. Not all decimal values can be represented exactly in binary; without the `NUMERIC` datatype, a value imported as "0.1" will be stored as something slightly different, possibly leading to erroneous calculations and

user confusion. The system thus accepts what it is given and returns *exactly* the same; processing methods may subsequently use floating point or integer arithmetic as appropriate, but the database itself is neutral.

3.3 Ruleforms

Each Ultra-Structure CORE consists of both animation procedures and a distinct set of ruleforms that are specific to the CORE's target domain. Although a complete CORE576 is not yet achieved — as the CORE Hypothesis is non-falsiable, it will be impossible to definitively say that one has been achieved — several ruleforms are now defined (though they are certainly open to modification as work progresses). Their current incarnations are described below, with example rules found in Appendix A.

3.3.1 BioEntities

The *BioEntities* ruleform is one of the key ruleforms of the CORE576 system. It is an existential ruleform, declaring the existence of the various entities and (groupings of those entities) that will participate in biological processes and their analyses. For example, amino acids, the building blocks of proteins, are declared in this ruleform, yielding rules with factors such as "Serine", "Alanine", and "Tryptophan". Similarly, the classes of amino acids ("Polar Amino Acids", "Charged Amino Acids", etc.) are also defined, which facilitates the creation of animation procedures that apply not to individual rules, but to classes of rules. Additional entities defined in the *BioEntities* ruleform would include proteins, chemical elements and their isotopes, bacterial strains, genes, peptide mass lists (e.g. output from a mass spectrometry analysis), and results from various computational analyses (such as Mascot, GFS, and PROCLAME; see Section 3.4 below, as well as Appendix B).

As an existential ruleform, *BioEntities* has only one factor; namely, the name of the entity. In the current incarnation of CORE576, the considerations are mainly metadata-related, including a free-text description, usage notes, last-updated date, and the *Resource* responsible for adding the rule to the system (see Section 3.3.5 on *Resources* below).

3.3.2 BioEntity Network

The *BioEntity Network* is another vital ruleform in the CORE576 system. As a network ruleform, it defines a semantic network relating *BioEntities* to one another; each rule connects two *BioEntities* (referred to by the labels *Parent* and *Child*) with a named link (the *Relcode*). Based on the relationships encoded in the rules of this ruleform, new rules can be generated by logical inference, as detailed in Section 2.5. While this can be used to automatically "fill in" trivial rules (such as <***E. coli ribosomal protein S22 IS-A Molecule***>, deduced from the rules <***Protein IS-A Molecule***>, and <***E. coli ribosomal protein S22 IS-A Protein***>), it can be put to more powerful uses, such as determining all the measured peptides that are common to a set of liquid chromatography fractions.

Currently the *BioEntity Network* has the Network factors *Parent*, *Relcode*, and *Seq_Nbr* (for "Sequence Number"). An additional factor *Resource*, which functions to attribute the declaration of the rule to some *Resource*, enables several competing hypotheses to be present in the network simultaneously.

3.3.3 BioEvents

The contents of this existential ruleform describe various processes that the system is aware of. This can include biological processes inside a cell, processes the system carries out, and even experiments performed by members of the lab.

As an existential ruleform, the sole factor is the name of the *BioEvent*. Current considerations for this ruleform are limited to metadata (such as a *Description*), but will likely be expanded as this ruleform is further investigated.

3.3.4 BioEvents Network

As might be expected, the *BioEvents Network* ruleform defines groupings and connections among *BioEvents*. The chief use of this ruleform is currently the organization of individual experiments into larger research campaigns. For example, groups may be formed collecting all experiments performed in support of a particular research grant aim, or all experiments performed by a particular lab memeber, or all experiments performed on a particular piece of equipment.

The factors of this ruleform are the standard *Parent*, *Relcode*, and *Seq_Nbr*, while the considerations include *Child* and *Is_Original*. See Table A.4.

3.3.5 Resources

In the CORE576, the *Resources* ruleform includes such entities as laboratory members, standards bodies (such as IUPAC, the International Union of Pure and Applied Chemistry), biological textbooks, computer programs, and the like; a *Resource* is thus some source of information, an entity that declares that a rule entered into the system is true.

The sole factor of the *Resources* ruleform is the name of the *Resource*. The current non-metadata consideration is *Credibility*, an integer that ranges from 0 to 10 and indicates the trustworthiness of the *Resource*, with 10 being the most trustworthy. A standards body may be assigned a credibility of 10, whereas the network inference software may be assigned a credibility of 0 or 1, to reflect the provisional nature of

inferred rules.

3.3.6 Resources Network

Like all Network ruleforms, the *Resources Network* defines relationships between *Resources*. The most common relationship types that currently exist between *Resources* are simple grouping ones, such as those that declare computer programs to be members of the group "Software".

The structure of this ruleform is identical to that of *BioEntity Network*, with the obvious difference that the linked entities come from the *Resources* ruleform rather than *BioEntities*.

3.3.7 Relcodes

The *Relcodes* ruleform, another vital ruleform in the CORE576, defines the various relationships that can exist in each of the Network ruleforms of the system. Its current form is as described earlier in Section 2.4.3.

3.3.8 Attributes

Often, the entities that are used in the CORE576 system will have various pieces of relevant information that must be recorded and tracked. In the mass spectrometry applications, for instance, the mass of a protein or peptide is most certainly a piece of data to be tracked. These attributes are not themselves entities, but are attached to entities. Thus, the *Attributes* ruleform is an existential ruleform that defines the kinds of attributes that will be considered in the system. Entries would include "Atomic Number", "Monoisotopic Mass", "Sequence Start Position" (for example, to locate a peptide fragment within the context of a parent protein's primary amino

acid sequence), and the like.

3.3.9 Attribute Network

Just as it can be helpful to have groupings of *BioEntities*, so too can it be helpful to have groupings of *Attributes*. An example of such a grouping used in the current system involves "Masses". Protein mass spectroscopists deal with several kinds of mass measurements: monoisotopic mass, average mass, nominal mass, most abundant isotopic mass, and so on. These are all instances of "Mass", and network rules can express this fact.

The factors of this ruleform are the standard Network ruleform factors of *Parent*, *Relcode*, and *Seq_Nbr*, with non-metadata considerations of *Child* and *Is_Original*. An example is shown in Table A.9.

3.3.10 BioEntity Attributes

This ruleform contains rules that specify the attributes a particular *BioEntity* has, as well as the values of those attributes. For instance, this is the ruleform that would contain the knowledge that the element carbon has an average mass of 12.0107.

It is not enough to say that a particular *BioEntity* has a given value for a given *Attribute* unconditionally and for all time, however. Rules in this ruleform may be qualified with a *Resource* in order to, among other things, denote the credibility of the rule, as well as accommodate alternative hypotheses. An additional factor is *BioEvent* which denotes the process by which this attribute observation was obtained. There are two non-metadata considerations for this ruleform, only one of which can be non-null. These considerations hold either a textual string or a number, depending on the *Attribute*. An example is given in Table A.10.

3.3.11 BioEntity Attribute Authorization

All the attributes that the CORE576 system is capable of tracking are declared in the *Attributes* ruleform, but there is nothing in that ruleform that constrains the entities to which the attributes may be applied. For instance, the attribute "Monoisotopic Mass" is applicable for chemical elements such as carbon and hydrogen, as well as for proteins and peptides, but is nonsensical when applied to, say, E. coli bacteria. In order to define which *BioEntity / Attribute* pairings are acceptable, the Ultra-Structure notion of an "authorization rule" must be invoked. Long and Denning (1995) state that authorization rules are responsible for declaring "what inputs are allowable, authorized, or expected." Thus, to acknowledge the fact that chemical elements can have a monoisotopic mass, a rule to that effect must exist in this ruleform; the lack of a corresponding rule for E. coli bacteria indicates the inappropriateness of that pairing. Note, however, that an authorization rule does not mean that the *Bioentity must* have an attribute, just that it *may*. For instance, an experimental protein, as a protein, is allowed to have a monoisotopic mass, but that mass may not have been measured or calculated yet.

Since a given *BioEntity* may potentially have several valid attributes, and all *BioEntities* in a class (say, all "Proteins") can have the same attributes, there would be a very large number of redundant rules in this ruleform if there were no Network ruleforms. For instance, knowing that all proteins can have a monoisotopic mass, we would like to avoid having to explicitly state an individual rule for every protein in the system (Protein A, Protein B, Protein C, etc.) that repeats this fact.

Additionally, clusters of related attributes may exist. The *Attribute Network* provides a means for formally grouping these related concepts. Rules could be entered into that ruleform, creating a notion of "Protein applicable mass attributes." The *BioEntity Attribute Authorization* ruleform could thus contain a single rule to the

effect that all proteins can have any and all of the attributes in the "Protein applicable mass attributes" group. This economy of rules is one of the strengths of the Ultra-Structure notational approach.

This ruleform is used to ensure that only valid attributes are entered into the system. In a future graphical interface, it will be used, for example, in screens for users to manually enter rules; the user will only be given valid choices of attributes to enter for a chosen *BioEntity*.

Currently, the factors of the ruleform include *BioEntity* and *Seq_Nbr* (because a given *BioEntity* may have several authorized attributes). The considerations include the *Attribute*, along with metadata fields.

3.3.12 BioEntity Network Attributes

Just as *BioEntities* can have *Attributes*, so too can *BioEntity Network* connections. This situation arises when additional information about the two connected entities needs to be tracked, and when attaching this information to one of the entities alone does not capture the true situation. An example of this is seen in the import of Mascot data (see Section 3.4.2); the predicted amino acid sequence of a peptide ion only makes sense when you have both a peptide ion as well as a protein that it is predicted to have come from. Without the context provided by the protein, one cannot say, for example, that a peptide ion represents the sequence "ITEVK" or "IVTEK;" both have the same mass, so additional information must be provided.

Just as *BioEntity Attributes* rules are contextualized with a *Resource* and a *BioEvent*, so too are rules in this ruleform. Thus the factors for a *BioEntity Network Attribute* rule include *Resource*, *BioEvent*, *Attribute*, *Seq_Nbr*, as well as all the factors of the *BioEntity Network* rule the *Attribute* is being applied to. Similar to *BioEntity Attributes*, the considerations in this ruleform can accomodate text string

or numeric values for the *Attribute*. See Table A.12.

3.3.13 BioEntity Aliases

In the real world, a *BioEntity* may be known by many names or labels. For instance, the element carbon is also known by the symbol "C", and the amino acid threonine is known by the codes "Thr" and "T". Additionally, depending on the context, the same label can refer to several different BioEntities. When discussing amino acids, the label "C" is interpreted as referring to cysteine, but in the context of examining a genetic sequence it refers to the nucleotide cytosine. The ability to resolve these multiple names back to one "canonical" label — in our case, the label that acts as a key into the *BioEntities* ruleform — is essential, and the *BioEntities Aliases* ruleform presents a way to accomplish this goal.

The *BioEntity Aliases* ruleform (see example as Table A.13) currently contains three factors. The first, *Alias*, is the alias that is being resolved. This factor references the *Aliases* ruleform, which is currently a bare-bones ruleform, having only a single factor (the *Name* of the alias) and no considerations. The second, *Sense*, refers to a valid *BioEntity* and is the context in which to evaluate the alias, as described above. This takes advantage of the grouping and classification capabilities of the *BioEntity Network* ruleform; in the example given above, resolving the alias "C" to cysteine requires a context of "Amino Acid", which is itself a *BioEntity*. The final factor, *Seq_Nbr*, is added to allow a *Alias* to refer to potentially several *BioEntities* in the same sense. This appears counterintuitive, and may perhaps be dropped in the future, but the current way of handling references to proteins from other databases necessitates this. Specifically, results from the Mascot program contain a protein database accession number as well as a textual name for the protein. While the accession number is unique, the name is not necessarily so; there are 690 E. coli

proteins with the name "Hypothetical protein.- Escherichia coli." in the MSDB (the database underlying the Mascot program), as well as a number of other duplicately-named proteins. Thus, in order to handle information obtained from Mascot, the CORE576 system currently refers to proteins by using their accession number, with the name serving as an alias. Other data sources likely have similar issues.

The single consideration for this ruleform is *BioEntity*, which is the *BioEntity* the *Alias* refers to.

3.3.14 Attribute Metarules

This ruleform, along with *Attribute Protocol*, defined in the following section, are provisional in the sense that they were created to perform a specific task in the system; these ruleforms may or may not exist in future incarnations of the system, depending on the evolution and consolidation of the ruleforms. As such, they are somewhat particular to the task at hand, yet still retain some generality, as will be discussed.

This ruleform supports the functionality described in Section 3.5. The example ruleform is seen in Table A.14. As a metarule ruleform, this ruleform contains information on how to process other rules. In particular, this ruleform guides the processing of rules in the *Attribute Protocol* ruleform. The factors of this rule, *Event* and *Seq_Nbr* indicate which *BioEvent* the rules apply to. The considerations — *BioEntity1* and *Attribute1* — have *Relcodes* as their values, and are used to transform a *BioEntity* and *Attribute*, respectively, for examination of the *Attribute Protocol* ruleform. These rules thus determine for a given input, which rules of *Attribute Protocol* will be inspected.

As stated earlier, this ruleform is a provisional one. It is intended to support functions that create, delete, and manipulate various *BioEntity Attributes*.

3.3.15 Attribute Protocol

This ruleform is a companion to *Attribute Metarules*. After system input has been transformed by an *Attribute Metarule*, the transformed input is used to select rules from this ruleform. As such, the factors of this ruleform mirror the form of the metarule ruleform: *Event* corresponds to *Event* from *Attribute Metarules*, and *BioEntity1* and *Attribute1* correspond to the transformed values from those ruleforms (this is clarified by the example presented in Section 3.5). If processing requires multiple steps, *Seq_Nbr* will determine the ordering of that processing. The considerations of the ruleform specify the action to be taken when a rule's factors are matched. In this case, *Method* and *Class* define the Java class and method that should be invoked to perform a task, and *BioEntity2* and *Attribute2* specify the inputs for that method. As discussed in Section 2.6, a value of "ORIGINAL" in these considerations indicates that the unmapped, original input values be passed. Otherwise, the value of the considerations are passed as is to the specified software method. An example of this ruleform may be seen in Table A.15.

The ruleform as currently designed specifies Java methods to execute via the Java Reflection API. The specified method should be a static class method, as there is no way to specify a particular instance object. The ruleform would likely need to be modified if another programming language were used.

3.3.16 Transformation Metarules

This ruleform is paired with the *Transformation Protocol*; the two ruleforms combine to guide processes that transform *BioEntities* in some fashion. Currently these ruleforms are used to simulate protein translation, as well as RNA transcription. These processes are described in Section 3.6.

This ruleform is very similar to *Attribute Metarules*; it will be interesting to see if there is a way to consolidate the functions of these ruleforms into one more general ruleform. The ruleform has *BioEvent* and *Seq_Nbr* as its two factors, and *Resource* and *BioEntity* as its considerations. Each consideration value will be a *Relcode* used to transform some input in the appropriate Network ruleform. In this case, *Resource* and *BioEntity* act as inputs to be mapped. Mapped values will be used to inspect the rules in the *Transformation Protocol* ruleform.

An example is seen in Table A.16. As mentioned in Section 2.6, the special relcode value of "SAME" indicates that no mapping is to take place.

3.3.17 Transformation Protocol

This ruleform contains processing steps for transformation steps, currently including things like protein translation.

The ruleform has three factors: *BioEvent*, *Seq_Nbr*, and *Resource*. Its considerations include *BioEntity* and *Relcode*, which refer to entities declared in those two ruleforms. In essence, this ruleform contains rules that say how to select a specific mapping from the BioEntity Network, but guided by the *Transformation Metarules* ruleform.

An example is seen in Table A.17. The example detailed in Section 3.6 will make the usage of this ruleform and *Transformation Metarules* clear.

3.4 Data Import

In order to be useful as an investigative tool, it is necessary that laboratory data be imported into the systems ruleforms. Collections of Java classes were written for each of the several sources detailed below to aid in the import process. In general, the

data is read into an object-oriented data structure, which can then be passed to an importer class, which then generates the rules in the appropriate ruleforms necessary to represent the data. The entire data sets are first transformed to an in-memory data structure because some generated rules will require knowledge of the data set as a whole, which may not be available if a line-by-line interpretation of the data were performed. Additionally, such data structures can potentially be re-used in other applications.

3.4.1 OBO Import

The Open Biomedical Ontologies group (Open Biomedical Ontologies 2006) has defined a common format for representing a number of popular ontologies in the biomedical domain, including the Gene Ontology (Gene Ontology Consortium 2000). Thus, to import the Gene Ontology into the system, a parser and importer was written to the OBO format specifications, enabling import of any other OBO ontology.

Mapping from the Gene Ontology to CORE576 ruleforms is rather straightforward. The GO contains three sub-ontologies: `biological_process`, `cellular_component`, and `molecular_function`. Processes and functions are entered into the *BioEvents* ruleform, while components are entered into the *BioEntities* ruleform. The ontological links (the "IS-A" and "PART-OF" relationships) are entered into the appropriate Network ruleform.

3.4.2 Mascot Import

The Mascot program (Perkins et al. 1999) is a powerful tool for matching peptides to the proteins from which they are derived, provided that sequence information for those proteins exist in sequence databases. The UNC Mass Spectrometry Core Facility offers Mascot searching as a service to campus research labs. Though the

Mascot program is capable of generating output in a variety of formats, the UNC facility provides results in an Excel spreadsheet format only. To import this data into the CORE576 system, a collection of Java classes were written which convert the data (exported from Excel as a tab-delimited text file for ease of parsing) into an object-oriented data structure which is then used for rule generation.

In general, the Mascot results concern a collection of peptide ions, generated from a single biological sample, whose mass has been experimentally measured by a mass spectrometer. For each sample, a number of "hits" are presented, indicating that an identifying match has been made between some subset of the peptide ions and a protein in the MSDB (Pappin and Perkins 2005). Each hit contains information concerning the protein and the peptides that are predicted to match, including a number of pieces of contextual information, such as the predicted sequence of the peptide ions, as well as statistical confidence values. For each sample, a *BioEntity* is created, as well as one for each of the peptide ions generated from that sample. A rule in the *BioEntity Attributes* ruleform is created to store the observed mass of each peptide ion. Rules in the *BioEntity Network* ruleform are generated linking all peptide ions to the sample from which the are derived. Additional *BioEntity* rules are generated for each hit, with additional *BioEntity Network* rules linking hits to samples, as well as peptide ions to the hits they participate in. If an existential rule for the matched protein does not already exist, a new one is created in the *BioEntities* ruleform, with *BioEntity Network* rules being generated to link hit entities and peptide ion entities to the matched protein entity. To accommodate the information that is dependent on the match, rules are entered into the *BioEntity Network Attributes* ruleform.

3.4.3 GFS Import

The Genome-based Peptide Fingerprint Scanner (Giddings et al. 2003), or GFS, is a system developed in the Giddings lab that functions like Mascot to identify proteins based on mass spectrometry data, but without the restriction that protein information reside in any database.

Since GFS is available via the internet (`http://gfs.unc.edu`), the current standard output of GFS is an HTML document. Migration to XML output is underway, but an XML schema to represent GFS results is not yet complete. As an intermediate solution, an importer was written to parse a generically XML-formatted dump of the internal GFS results data structure. Generation of this XML is a language feature of the Apple Objective-C frameworks in which GFS is written.

As GFS presents results very similar to those of Mascot, the mapping process for GFS data is similar as well. Instead of matching peptides to proteins, GFS matches peptides to genomic sequences, so *BioEntity* rules creating these sequences are created instead of rules for proteins. Other rules are generated in the *BioEntity Network* and *BioEntity Network Attributes* as necessary.

3.4.4 PROCLAME Import

Also developed in the Giddings lab, the PROtein CLeavage And Modification Engine, or PROCLAME, is an algorithm for determining putative post-translational modifications based on intact protein mass measurements (Holmes and Giddings 2004).

Similar to the GFS output, an XML dump of the internal PROCLAME results data is the input to the importer. This work is currently ongoing, and may require either an extension of existing ruleforms or the creation of a new one. The reason for this is that while the above described data sets can be modeled with binary Network

rules, PROCLAME data appears to require ternary rules. An experimental protein has its whole mass measured, and is then linked to a known protein in the context of some number of post-translational modifications. The post-translational modification is currently conceived as a *BioEntity* in and of itself, so we have three *BioEntities* that are related to each other in a way that is not decomposable into binary Network rules.

3.5 Example Use: Mass Calculation

While the prototype CORE576 system could calculate masses of chemical compounds, its capabilities were rather limited. Only the average mass of molecules was able to be calculated, whereas researchers (particularly mass spectroscopists) are generally interested in a number of different masses, as mentioned above. In the current CORE576, nominal, monoisotopic, and average masses are able to be calculated, for both molecules (such as water and proteins) and for elements. To support this, the system contains rules specifying the masses and abundances of all chemical isotopes, imported from the National Institute for Standards and Technology's (NIST) website; all other information is directly calculable from these data.

To facilitate the calculation of these masses for molecules, the grouping capabilities of the *BioEntity Network* ruleform were leveraged. Two classifications of molecules were made: "Base Mass Type" and "Composite Mass Type". If a molecule belongs to the class "Base Mass Type", its molecular composition is defined in terms of the numbers of atoms of each of its component elements. For example, a molecule of water would be defined as having two atoms of hydrogen and one of oxygen, while a molecule of the amino acid glycine would be defined as having two atoms of carbon, two of oxygen, one of nitrogen, and five of hydrogen. The class is called "Base Mass Type" because instances of these molecules will be used to assemble instances

of "Composite Mass Type". For instance, instead of having to define the composition of an amino acid *residue* (an amino acid with a loss of the equivalent of one molecule of water) in terms of atom counts (which would largely be a duplication of the definition of the amino acid molecule), one can state rules that say, e.g., a glycine residue has a glycine molecule as its "base", with a water molecule subtracted. The mass of the glycine residue can then be calculated by subtracting the mass of a single water molecule from that of the intact glycine molecule.

To calculate masses of elements, a different approach needs to be taken. Masses of elements depend on the masses of their isotopes, as well as each isotope's relative abundance in nature. For instance, a monoisotopic mass of an element is the mass of its most abundant isotope, while the average mass is the sum of the masses of all the element's isotopes, weighted by their relative abundances. Thus, depending on what kind of mass calculation is requested, a different algorithm is required. It may seem odd to want to calculate the mass of an element when such information is readily available, as is the case on the NIST website. Periodically, IUPAC will release updated measurements of isotopic weights and / or abundances (due to more accurate measurement technologies, for example). In such a case, the new data can be entered into the system, which can then compute any elemental mass measurements that depend on that data.

An envisioned use for this mass calculation facility would include automatic protein mass calculation. Upon entering the sequence of a new protein to the system, a mass calculation event could be triggered, calculating average, monoisotopic, and nominal mass, as well as most abundant isotopic mass, an important measurement for mass spectroscopists (to be implemented at a later date). Additionally, a common task in the Giddings lab is to correlate experimentally measured peptide masses with the masses of theoretical peptides; these theoretical peptides could be generated automatically from the new protein sequence, and each of their masses could subsequently

be generated.

Small software methods implementing algorithms for calculating the various masses were written. To trigger the calculation of a mass, three parameters need to be input into the system; the name of a *BioEvent* ("Calculate Attribute"), the name of the *BioEntity* to operate on ("Water," "Carbon," "Serine Residue," etc.), and the name of the *Attribute* to calculate ("Monoisotopic Mass," "Average Mass," etc.); for the purposes of illustration, assume the parameters "Calculate Attribute," "Serine," and "Average Mass." These three parameters are used to drive the processing of the request. Initially, the *BioEvent* is used as a key to search the *Attribute Metarules* ruleform, which can be seen in Table A.14. The two rules for "Calculate Attribute" seen there would be returned. The contents of the *BioEntity1* and *Attribute1* fields are not a *BioEntity* and *Attribute*, respectively, but rather relcodes used to navigate the *BioEntity Network* and *Attribute Network* ruleforms in order to generalize the *BioEntity* and *Attribute* request parameters. First, the metaprotocol rule indicated by $Seq\_Nbr = 1$ is considered. The *BioEntity Network* ruleform is consulted, where a rule stating that "Serine" with respect to the relcode "MoleculeType" is a member of class "Base Chemical, Molecule, Or Group". Similarly, the *Attribute Network* is consulted for the mapping of "Average Mass", which yields a rule stating that "Average Mass" with respect to the relcode "Attribute Type" is a member of class "Mass". Thus, our original set of parameters has been transformed into "Calculate Attribute," "Base Chemical, Molecule, Or Group," and "Mass."

Continuing with the processing, this new transformed set of parameters is used as a search key for the *Attribute Protocol* ruleform, seen in Table A.15. A matching rule is found, which indicates that its considerations should now be examined. The *Class* and *Method* considerations indicate which software method is to be invoked, while *BioEntity2* and *Attribute2* indicate the parameters to be passed. In this case, both values are "ORIGINAL", which indicates that the untransformed inputs (i.e.

"Serine" and "Average Mass") should be passed. The animation procedure driving this entire process will then reflectively invoke the specified method, passing it the specified parameters.

3.6 Example Use: Protein Translation Simulation

The original CORE576 prototype had the capability of translating a DNA sequence into a corresponding protein sequence, but the current system has more sophisticated capabilities in this regard.

One significant drawback of the prototype system is that the only kind of translation possible was that dictated by the universal genetic code. The code is universal in that for the vast majority of known species it specifies the mapping from three-letter DNA codon to single amino acid, but in a number of species, this code is slightly modified. Generally these alternative genetic codes are identical to the universal code with the exception of one or two altered mappings. Unfortunately, the prototype did not account for the existence of these alternative codes. Additionally, the prototype only handled the translation of DNA into protein, but not DNA transcription to RNA, or RNA to protein.

Biological sequences were also represented in a rather unconventional way. A gene would be entered into the BioEntities ruleform. Since the nucleotides that comprise the sequence of a gene are themselves BioEntities, the sequence of the gene would be encoded as a series of BioEntity Network rules, utilizing the *Seq_Nbr* factor as an index into the sequence; the fact that Gene X has cytosine as the nucleotide at the fifth position would be stored thusly: <***Gene X HasComponent 5 Cytosine***>. With increasingly long genetic sequences (such as chromosomes or whole genomes), this approach would be untenable. Currently sequences are stored conventionally as strings, e.g. "ATCGGCAT..." in the database.

To simulate the translation of a strand of DNA into protein, one must first split the target DNA sequence into codons. Unfortunately, this processing currently exists outside of the confines of the Ultra-Structure ruleforms; incorporating this "string parsing" is ongoing work. However, given a codon (in the form of a three-letter triplet, such as "ATG") and a genetic code (encoded as a *Resource*), one can begin to translate the protein. Currently, the 64 different possible codons are represented as *BioEntities*, named "ATG", "ATC", etc.

Recall that a *Resource* is some entity that is the source of a piece of information in the system. Thus a genetic code, such as "Universal Genetic Code" or "Ciliate Nuclear Genetic Code", is the source of the information that informs us which amino acid a particular codon codes for. Also recall that the "Universal Genetic Code" is in essence the code that other alternative codes are based upon; alternative codes are kinds of "exceptions" to the universal code. This "exception" idea can be encoded as a rule in the *Resources Network* ruleform: <***Ciliate Nuclear Genetic Code IS-EXCEPTION-TO Universal Genetic Code***>. With this information in place, translation can begin.

Input for this transformation will consist of the *BioEvent*, *Resource*, *BioEntity*, which will be used to select rules from the *Transformation Metarules* ruleform in much the same way as with the *Attributes Metarules* ruleform. As an example, consider the input <***Protein Translation, Ciliate Nuclear Genetic Code, ATG***>. In other words, we want to determine the amino acid that is encoded by the DNA codon "ATG" using the "Ciliate Nuclear Genetic Code".

First, inspect the *Transformation Metarules* ruleform for rules matching the *BioEvent* "Protein Translation". As seen in Table A.16, two rules match; these will be inspected in an order determined by the value of *Seq_Nbr*. The first rule states that no transformations are to be performed (indicated by the special relcode value

"SAME").

Processing is now transferred to the *Transformation Protocol* ruleform, attempting to match the factors *BioEvent* and *Resource*. As seen in Table A.17, two rules match. Both state that the *BioEntity* to be transformed should be the "ORIGINAL" one passed as input; in this case, the codon "CTG". The first matching rule specifies a *Relcode* of "Signal Encoded". This rule tells us to examine the *BioEntity Network* ruleform to find out what signal (if any) the CTG codon encodes in the Ciliate Nuclear Genetic Code (certain codons specify processing signals, such as "Begin Translation" or "Stop Translation"). As CTG encodes no special signal according to the Ciliate Nuclear Genetic Code, there will be no matching rule. The next matching rule is then inspected, requesting an inspection using the "Amino Acid Encoded" relcode instead, since it is possible for codons to encode both a signal and an amino acid.

Earlier it was mentioned that alternative genetic codes generally differ from the universal genetic code in only a small number of codon assignments. As such, it would not make much sense to essentially duplicate rules stating, for example, that both the Ciliate Nuclear Genetic Code and the Universal Genetic Code both mapped the codon CTG to the amino acid lysine. We should be able to take advantage of the rule in the *Resources Network* ruleform that says <***Ciliate Nuclear Genetic Code IS-EXCEPTION-TO Universal Genetic Code***>, which is exactly what is done. Rules are entered into the *BioEntity Network* linking each of the 64 possible codons to the appropriate signals and amino acids, as defined by the Universal Genetic Code. Thereafter, only those assignments that *differ* between the universal and alternative genetic codes are recorded as rules. Thus, to add a new alternative genetic code to the system, only two or three new *BioEntity Network* rules will generally need to be added.

With this in mind, the last inspection described (using the "Amino Acid Encoded" relcode) will most likely not match any rule — there are only three codon assignments difference between the Ciliate Nuclear Genetic Code and the Universal Genetic Code. As a result, processing will revert back to the *Transformation Metarules* ruleform, with the second of the two originally matched rules.

This second metarule, instead of requesting no mapping of the input values, specifies that the *Resource* — in our example, the "Ciliate Nuclear Genetic Code" — should be masked via the *Relcode* "IS-EXCEPTION-TO". Examination of the *Resources Network* indicates that the result of this masking will be "Universal Genetic Code".

With this new transformed input set, the *Transformation Protocol* ruleform is inspected as before, this time matching with "Universal Genetic Code" rather than the earlier value of "Ciliate Nuclear Genetic Code". Again, no "Signal Encoded" is found, but this time an "Amino Acid Encoded" *is* found, namely "Leucine". In this way, an entire genetic sequence could be translated into protein sequence, using any number of genetic codes.

Note that this transformation method is general, in that largely the same processing can be used to simulate the transcription of RNA from DNA. Instead of scanning a genetic sequence in groups of three nucleotides (i.e. codons), scanning would have to take place a single nucleotide at a time, resolving the single-letter codes to the *BioEntity* representing the nucleotide base by utilizing the *BioEntity Aliases* ruleform. However, once that was done, the processing would be essentially the same; the rules would look different, but the relationships among them would be identical to those among the protein translation rules. A tracing through this processing will not be given, but the rules governing it are given in the example ruleforms in Appendix A.

Chapter 4

Related Work

The post-genome information overload in the biological sciences has necessarily resulted in the development of methods to efficiently and effectively manage that information. This work on an Ultra-Structure for biology is the latest in this line. What follows is a survey of some of these efforts, with comparisons to Ultra-Structure.

4.1 BioNetGen

The concept of applying rule-based reasoning to biological problems is not new. A recent application, BioNetGen (Blinov et al. 2004; Faeder et al. 2005), is a specialized system for modeling cell-signaling networks. A user creates a plain-text input file, defining the existence and amounts of signaling molecules (as well as their modification states) and the equations that govern their interactions. This file is then processed by a Perl program, simulating the network of biological interactions. This results in the generation of plain-text output, including complete listings of all generated chemical species and their modification states, timecourses of protein levels, and even SBML-formatted XML output.

Compared to the Ultra-Structure methodology, this most certainly not a general tool. All components of the system are specific to cell signaling networks. Ad-

ditionally, the structure of the rules is not clearly specified with anything resembling ruleforms. The rules are expressed for the most part in mathematical language, as opposed to the natural language-like rules of Ultra-Structure. BioNetGen does, however, take a relatively small number of rules and fully propagates a network of logically consistent data that are implied by those rules. In this way, BioNetGen and Ultra-Structure are similar; both can be used to investigate and evaluate hypotheses, represented by the input rules. The CORE576 model as it currently stands is not explicitly mathematical, as BioNetGen is, but nothing in Ultra-Structure theory necessarily precludes this. Indeed, a proposed extension to the current model involves the addition of confidence values to rules, to be utilized in probabilistic inferencing. Another obvious difference between BioNetGen and CORE576 is that BioNetGen is not an information storage system, but rather a modeling and simulation system.

4.2 SPLASH

The SPLASH (systematic proteomics laboratory analysis and storage hub) system (Lo et al. 2006) is a web-accessible, XML-based proteomics information system. Its development arose from an ever-present need in the bioinformatics community to integrate information from a variety of sources into some unified architecture.

The heart of SPLASH is the PEDRo proteomics data model (Taylor et al. 2003). This data model was proposed before the MIAPE (minimal information about a proteomics experiment) standard (HUPO PSI 2005a) as a way to encode relevant information concerning a proteomics experiment. SPLASH uses a slightly modified version of this data model to accommodate the particular work that its research group does, but all information can be mapped back to the original PEDRo model, preserving compatibility for both import and export. Thus, any PEDRo-compliant data can be easily brought into the system. Data import can be interactive, through

a variety of predefined forms, or via batch mode, wherein data exported from e.g. mass spectrometers are processed by helper programs and input into the underlying database. Internally, data is stored in a conventional relational database mapping of the PEDRo data model. The system is XML-based in the sense that its inputs and outputs are XML documents, which is well-suited to its web-accessible nature; of particular interest is the generation of SVG representations of two-dimensional gel images. SPLASH also incorporates outside information from the Gene Ontology (GO) and the Kyoto Encyclopedia of Genes and Genomes (KEGG).

SPLASH is based on PEDRo; the system's main contribution is a web-accessible front-end for interacting with the data, in terms of querying, analyzing, importing, and exporting. SPLASH has a modular architecture, which currently includes components for data entry and management, as well as search and data mining. The maintainers of SPLASH stress that they have kept an eye open for evolution of community data interchange standards, such as MIAPE and mzData (HUPO PSI 2005b), in order to facilitate their seamless interchange with SPLASH repositories. As a result, the SPLASH system is robust and useful for integration and inspection of various proteomics information sources.

Both the CORE576 Ultra-Structure and SPLASH platforms aim to be a general information architecture for bioinformatics researchers. However, SPLASH is clearly crafted specifically for the proteomics community. CORE576 was developed in the context of a proteomics laboratory, but nothing in the structure of the system restricts it to only proteomics data. Indeed, one of the tenets of Ultra-Structure methodology is that the underlying architecture (the ruleforms and procedures for manipulating them) will remain the same across a class of domains. The differences come into play in the form of the rules that are entered into the system. SPLASH encodes this information into the very structure of the system, and so is limited to the proteomics community.

Although SPLASH has data mining modules, it does not intrinsically contain any kind of inferencing capabilities, which is a significant difference with CORE576.

4.3 PRISM

PRISM is an distributed information system targeted toward high-throughput proteomics research, developed by the Pacific Northwest National Laboratory to manage their LC-MS proteomics workflows (Kiebel et al. 2006). It is a large system, composed of several interconnected servers, performing specialized tasks. It has a unique modular architecture that allows a degree of flexibility in managing data processing pipelines. While the PRISM system is quite different from an Ultra-Structure system, there exist a number of interesting similarities between the two approaches.

The first of these similarities concerns the so-called "manager programs," specialized, autonomous programs that periodically query a main data server to see if there is any work for them to perform. These programs perform tasks such as preprocessing incoming data from laboratory equipment, archiving old data sets, and performing specific data analyses. These manager programs are described as state machines whose operations are based on the values of state variables associated with metadata records for the scientific data managed by the system. Tracking entities (the PRISM term for these metadata records) are organized into a hierarchy, rooted at "Campaign" and extending down to "Analysis Job," enabling the tracking of data throughout the entire experimental process, as well as the organization of data into experiments and larger encompassing research campaigns.

The manager program aspect of the PRISM architecture recalls Long's descriptions of the CORE650 business Ultra-Structure system (Long and Denning 1995). In that system, animation procedures are responsible for generating implicit work order steps, ordering them, and writing them to a Work Orders ruleform (table). The

agencies responsible for executing these various steps, be they human or computer, will monitor the Work Orders table for tasks that indicated that they are "ready" to be performed. When a task is completed by an agent, it is flagged as such; the Ultra-Structure system then determines which subsequent tasks are now ready to be executed. This cycle is repeated until all tasks have been completed.

Additionally, PRISM's tracking entities recall Ultra-Structure's concept of Network ruleforms, though in a more restricted form. PRISM maintains a strict hierarchy of entities, whereas Ultra-Structure allows for richer graphs. PRISM's tracking entities do allow for specialized handling of experimental data sets based on metadata attributes: an example given concerned the differential processing of experimental data from ^{18}O- versus ^{15}N-labeled samples. Rules associated with particular tracking entities direct their processing by the manager programs. It is unclear how these rules are implemented in the PRISM system; are they encoded into software, or are they decoupled and available to the user for ease of inspection and modification? These rules are certainly important in terms of directing processing, but it is not clear how central they are to the overall PRISM architecture. Additionally, it does not seem that PRISM's hierarchy of tracking entities allows for any form of logical inferencing, which is an important aspect of any Ultra-Structure system.

4.4 PRIDE

PRIDE (the protein identifications database) is another XML-based proteomics data repository (Martens et al. 2005). It was developed at the European Bioinformatics Institute (EBI) to address a common problem: protein identification data published in journal articles is typically available only in PDF tables, which is not readily amenable to machine processing. The PRIDE system is envisioned as a way to turn "publicly available data into publicly accessible data" by creating a machine-accessible

repository for protein identification information.

The PRIDE data format is a hierarchical, experiment-centered XML model, backed by a relational database backend. The creators of the system list this as a strength of the system; as there are many possible mappings of hierarchical data to a relational model, third parties can create relational implementations that are tailored for specific data processing and querying needs. The relational schema used by the official PRIDE system is thus a reference implementation, presumably chosen for performance in general usage cases.

PRIDE certainly fills an important role in proteomics research. It is a rather specialized role, however, and is clearly proteomics-specific. Additionally, the system is only used for storage of information; any manipulation or analysis of the data is wholly external to the system.

4.5 BioWarehouse

The BioWarehouse system (Lee et al. 2006) represents an evolution of the federated database approach taken by other projects. The warehouse approach integrates information across a number of different data sources, but does so in a local context; all data sources are replicated locally and integrated via a common schema. Lee et al. (2006) list several reasons why decentralized federation approaches may be less than adequate, and argue that local replication can have definite benefits.

To facilitate the import of different datasets, BioWarehouse has the concept of Loaders, specialized programs that are responsible for importing data from some dataset (e.g. Swiss-Prot, the Gene Ontology) into the common BioWarehouse schema. The CORE576 system has similar facilities; an Open Biomedical Ontologies (OBO) importer has been developed for import of the Gene Ontology and other OBO-

formatted ontologies, as well as importers for output data from Mascot, GFS, and PROCLAME software.

While BioWarehouse does have schema-level support for experimental data, the system appears to be targeted towards more large-scale integration efforts. In contrast, CORE576 is currently very much focused on the integration of experimental data on a local level, though incorporation of large data sets on the scale of BioWarehouse is a development goal.

While BioWarehouse integrates data from various sources, it does not store any information regarding methods of processing these data. Here, CORE576 and BioWarehouse are clearly different. Further, BioWarehouse achieves its integration using traditional relational database modeling approaches, as opposed to the novel rule-based approach of CORE576.

4.6 Summary

Clearly the creation of information management systems is an popular and important area of research for proteomics and biological research in general. Most systems address the problem of data integration in some way, usually by importing data from a number of sources into some common repository. Some systems are essentially for storage only, while others include some form of data analysis capabilities; some are specialized while others are general. All are built on standard information technologies, whether it be relational databases or XML. The CORE576 system is the next research project in this line of work. It is distinct from other approaches in its use of the Ultra-Structure methodology; no other system to date has explored this technology for biological information management.

Chapter 5

Future Work

The work presented in this thesis is largely exploratory in nature. As such, there is much work to be done in the future in order to create a truly robust and user-friendly information system. To that end, several efforts are outlined below.

5.1 Interface

The interface to the CORE576 is currently very limited. Most interaction takes place via server command line. The Network Propagation Engine has a basic GUI implemented in Swing (the standard Java widget set). Interaction with the rules of the system is facilitated through the use of the PgAdmin III software (`http://www.pgadmin.org/`), and Open Source frontend to the PostgreSQL DBMS. For development purposes, these interfaces are acceptable, but will not be appropriate for end-users. Thus, a comprehensive user interface must be developed. Initial efforts will likely provide either command line interaction or a simple Swing GUI for a small number of tasks. Further efforts will likely be devoted to expanding the scope of the Swing GUI. If a web interface is desired, a natural choice will be J2EE servlet technology, using Tomcat, the J2EE servlet container reference implementation, and either the popular Jakarta Struts framework, or the newer Stripes framework, which

leverages new features of the Java 1.5 JDK.

5.2 Ad Hoc Querying

Existing Ultra-Structure systems are "closed" systems, in that all the queries and analyses the systems are capable of handling are encoded as rules in the system. Ad hoc querying is thus not supported. This works fine for business systems, where common analyses are repeatedly performed, but may not be suitable for biological research contexts. To be sure, there are regular analyses that will be performed, but it is certainly reasonable for a researcher to come up with some exploratory question, the solution for which may not be addressable by existing rules. However, an analysis of the kinds of questions commonly asked by researchers may reveal commonalities; these may be leveraged in the design of Ultra-Structure rules and animation procedures that can respond to broad classes of queries.

Should ad hoc querying prove necessary for the CORE576 system, a key requirement for its support will be an improved method of network propagation, dealt with in the next section.

5.3 Network Propagation

Possibly the most pressing work for the CORE576 lies in network ruleform propagation. Previously implemented Ultra-Structure systems have taken the approach of explicitly deducing all logical consequences of the given rules in a particular network ruleform at once, writing these new rules into the database. This approach requires several passes of logical deduction through the ruleform; each pass extends the "inference frontier" out by one step. Since newly deduced rules can combine with previously deduced rules to generate even more rules, deduction must continue

until a pass through the ruleform results in no additional rules being generated. This approach works well with small collections of rules, or in situations where immediate access to deduced facts is not of great importance: some implementations of the CORE650 business management system propagate their networks only on a weekly basis (Jeffrey Long, personal communication).

In a systems biology context, however, such approaches may not be appropriate. A single mass spectrometry experiment, for example, can generate copious amounts of data, resulting in the addition of thousands and thousands of new rules. Furthermore, any facts and connections implicit in these data will likely need to be immediately accessible. For example, upon importing data from a new experiment, a researcher might want to know how that data relates to existing data in the system. The rulebase of such a system would be expected to grow to millions of rules (as have previous Ultra-Structure systems). In such a scenario, the brute force deduction scheme outlined above would rapidly become impractical.

An alternative to pre-computation of deducible rules is restricted dynamic query-time deduction. This approach would incur a runtime performance penalty, but would eliminate the repeated and lengthy full-ruleform deductions that would necessarily have to be performed. In Section 2.4.3, the concept of relcode networks was discussed, a concept that will be necessary to implement query-time deduction. Since network ruleforms define directed acyclic graphs, a given entity may have multiple parents. This offers several choices of direction for deduction; pre-computing deducible rules entails deduction proceeding along all these paths, but query-time deduction requires some way of narrowing down the choices. For instance, the deduction of which other proteins a given protein interacts with may not need to proceed down a path dealing with which gene encodes that protein. A relcode network would allow relcodes to be grouped together into broader classes, enabling animation procedures to choose only deduction paths based on relcodes of a desired class. Such an approach

stands to considerably reduce the amount of deduction that must be done for a given query.

5.4 Advanced Functionality

In addition to working out implementation-level details, additional functionality will be added. One straightforward yet helpful feature will be the creation of reports detailing all the currently known information on a given entity, particularly proteins. Another more exciting function would be to use the CORE576 system to assess the validity of PROCLAME predictions, given a set of mass spectrometry data. When a peptide mass fingerprinting program like Mascot or GFS identifies proteins based on peptide masses, there are invariably peptides that remain unmatched. This may be for a number of reasons, but post-translational modifications are among the most interesting. With all mass spectrometry data and identifications stored in the same system, it will be straightforward to automatically generate a list of unmatched peptides. Predicted posttranslational modifications from PROCLAME can be evaluated relative to these peptides; for each peptide, does the mass change predicted map it to a Mascot- or GFS-identified protein? PROCLAME predicts several scenarios, each of which can appear equally legitimate; the only way to determine which one is actually true is to correlate each scenario to the experimental data on hand. This task is currently one that is tediously performed by hand; automated analysis would be a boon to researchers. This work is currently underway.

Chapter 6

Conclusion

Ultra-Structure presents an information system design methodology that may prove beneficial to the biological research community by enabling diverse data and protocol information to be stored, queried, and manipulated in a single repository. This can facilitate analyses that incorporate all available knowledge in ways that current approaches cannot. This thesis presents the results of an exploration of Ultra-Structure in a biological research setting by building on and extending previous work done on a prototype system. A more complete CORE576 will certainly contain more ruleforms than are listed here, and the ruleforms described herein are likely to change in the course of development, as the "true" deep structure of biological research is approached. Indeed, these ruleforms described here are rather different than the ones in the original prototype, and have undergone revision since the beginning of this work. The processes encoded into these ruleforms, while simple, do convey the general ideas of an Ultra-Structure system, and this overall project represents the first application of these ideas to biological research. It is hoped that this work can be a foundation on which further development can be based. The ultimate goal will be the creation of an information management system that serves as an effective tool to aid biological researchers in their work.

Appendix A

Example Ruleforms

Presented here example ruleforms, shown in tabular format, which are referred to many times in the text. As a convention, all factors are shown to the left of the double vertical line, and all considerations are on the right. The header of table defines the ruleform itself, while rows of the table are instances of individual rules in that ruleform.

The ruleforms that follow are condensed versions of the actual ruleforms used in the CORE576 system. In practice, ruleforms generally have a number of additional considerations which annotate rules with metadata, such as declaring what agency is responsible for asserting this rule, when the rule was last modified, and other "bookkeeping" information. These condensed ruleforms are given in order to convey the essence of the ruleform, as well as for space considerations.

Table A.1: The `BioEntities` Ruleform

Name	*Description*
Serine	The amino acid
Serine Residue	The residue form of serine
Experiment 5 LC Fraction 6	The sixth liquid chromatography fraction obtained from Experiment 5
Q4FBC3_ECOLI	A protein in the MSDB; see BioEntity Aliases
Carbon	The chemical element
Carbon-12	An isotope of carbon

Table A.2: The `BioEntity Network` Ruleform

Resource	*Parent*	*Relcode*	*Seq_Nbr*	*Child*
ANY	Amino Acid (Neutral)	Includes	1	Glycine
ANY	Amino Acid (Neutral)	Includes	2	Cysteine
ANY	Amino Acid (Neutral)	Includes	3	Tyrosine
ANY	Amino Acid	Includes	1	Glycine
ANY	Amino Acid	Includes	2	Amino Acid (Polar)
ANY	Amino Acid	Includes	3	Amino Acid (Non-Polar)
ANY	Amino Acid	Includes	4	Amino Acid (Neutral)
ANY	Water	Polarity	1	Polar Molecules
Universal Genetic Code	CTG	Amino Acid Encoded	1	Leucine
Ciliate Nuclear Genetic Code	TAA	Amino Acid Encoded	1	Glutamine
Universal Genetic Code	TAA	Signal Encoded	1	Translation Stop Signal

Table A.3: The `BioEvents` Ruleform

Name
Definition
Protein Translation
DNA Replication
RNA Transcription
Reverse Transcription
MALDI Mass Spectrometry Experiment 534
NIH Grant 734883 Experiments
Christopher Maier's Experiments
Mass Spectrometry Experiments

Table A.4: The `BioEvents Network` Ruleform

Parent	*Relcode*	*Seq_Nbr*	*Child*	*Is_Original*
MALDI Mass Spectrometry Experiment 534	IsA	1	Mass Spectrometry Experiments	t
NIH Grant 734883 Experiments	Includes	1	MALDI Mass Spectrometry Experiment 534	f
Christopher Maier's Experiments	Includes	1	MALDI Mass Spectrometry Experiment 534	f
Mass Spectrometry Experiments	Includes	1	MALDI Mass Spectrometry Experiemnt 534	f

Table A.5: The `Resources` Ruleform

Name	*Credibility*
Ciliate Nuclear Genetic Code	10
Contra Software	0
GFS	0
IUPAC	10
Mascot	0
Propagation Software	0
Property Inheritance Software	0
Top Node	0
Universal Genetic Code	10

Table A.6: The `Resources Network` Ruleform

Parent	*Relcode*	*Seq_Nbr*	*Child*	*Is_Original*
Universal Genetic Code	HasException	1	Ciliate Nuclear Genetic Code	t
Standards Body	Includes	1	IUPAC	t
Genetic Code	Includes	1	Universal Genetic Code	t
Genetic Code	Includes	2	Ciliate Nuclear Genetic Code	t
Ciliate Nuclear Genetic Code	IsExceptionTo	1	Universal Genetic Code	f
IUPAC	IsA	1	Standards Body	f
National Institute of Standards and Technology	IsA	1	Standards Body	f
Universal Genetic Code	IsA	1	Genetic Code	f
Ciliate Nuclear Genetic Code	IsA	1	Genetic Code	f
Genetic Code	IsA	1	Top Node	f

Table A.7: The `Relcodes` Ruleform

Name	*Contra*	*Is_Preferred*	*Is_Transitive*
SAME	SAME	t	f
Attribute Class	Includes Attribute	t	t
BaseComponent	IsBaseComponentOf	t	f
HasException	IsExceptionTo	t	f
Includes	IsA	f	t
Includes Attribute	Attribute Class	f	t
IsA	Includes	t	t
IsAddedTo	Addition	f	f
IsBaseComponentOf	BaseComponent	f	f
IsPolarityOf	Polarity	f	f
MoleculeType	MoleculeTypeOf	t	t
MoleculeTypeOf	MoleculeType	f	t
PartOf	HasPart	t	t
Polarity	IsPolarityOf	t	f
RNA-Equivalent	DNA-Equivalent	f	f
Subtraction	IsSubtractedFrom	t	f

Table A.8: The `Attributes` Ruleform

Name
ORIGINAL
Abundance
Amino Acid Sequence
Atom Count
Atomic Number
Atomic Weight
Average Mass
Delta Mass
Mass
Mass Charge State
Mass Tolerance
Monoisotopic Mass
Nominal Mass
Nucleotide Sequence
Top Node

Table A.9: The `Attribute Network` Ruleform

Parent	*Relcode*	*Seq_Nbr*	*Child*	*Is_Original*
Average Mass	IsA	1	Mass	TRUE
Mass	Includes	1	Average Mass	FALSE
Mass	Includes	1	Monoisotopic Mass	FALSE
Mass	Includes	1	Nominal Mass	FALSE
Mass	IsA	1	Top Node	TRUE
Monoisotopic Mass	IsA	1	Mass	TRUE
Nominal Mass	Attribute Class	1	Mass	FALSE
Nominal Mass	IsA	1	Mass	TRUE
Nominal Mass	IsA	1	Top Node	FALSE
Top Node	Includes	1	Average Mass	FALSE
Top Node	Includes	1	Mass	TRUE

Table A.10: The `BioEntity Attributes` Ruleform

Resource	*BioEntity*	*BioEvent*	*Attribute*	*Seq_Nbr*	*String_Value*	*Numeric_Value*
IUPAC	Hydrogen	Definition	Atomic Number	1		1
IUPAC	Hydrogen	Definition	Average Mass	1		1.00794
IUPAC	Hydrogen	Definition	Monoisotopic Mass	1		1.0078250321
IUPAC	Hydrogen	Definition	Nominal Mass	1		1
IUPAC	Carbon	Definition	Atomic Number	1		6
IUPAC	Carbon	Definition	Average Mass	1		12.0107
IUPAC	Carbon	Definition	Monoisotopic Mass	1		12
IUPAC	Carbon	Definition	Nominal Mass	1		12

Table A.11: The `BioEntity Attribute Authorization` Ruleform

BioEntity	*Seq_Nbr*	*Attribute*
Molecule	1	Mass
Element	1	Average Mass
Element	2	Monoisotopic Mass
Element	3	Nominal Mass

Table A.12: The `BioEntity Network Attribtues` Ruleform

Parent	*Parent*	*Relcode*	*Child*	*Net_Seq_Nbr*	*Seq_Nbr*	*Attribute*	*Numeric_Value*	*String_Value*
ANY	Serine	MustHavePart	Hydrogen	1	1	Atom Count	7	
ANY	Serine	MustHavePart	Carbon	2	1	Atom Count	3	
ANY	Serine	MustHavePart	Nitrogen	3	1	Atom Count	1	
ANY	Serine	MustHavePart	Oxygen	4	1	Atom Count	3	
ANY	Serine Residue	BaseComponent	Serine	1	1	Count	1	
ANY	Serine Residue	Subtraction	Water	1	1	Count	1	

Table A.13: The `BioEntity Aliases` Ruleform

Alias	*Sense*	*Seq_Nbr*	*BioEntity*
C	Amino Acid	1	Cysteine
T	Amino Acid	1	Threonine
G	Amino Acid	1	Glycine
C	Nucleotide	1	Cytosine
G	Nucleotide	1	Guanine
T	Nucleotide	1	Thymine
C	Chemical Element	1	Carbon
N	Chemical Element	1	Nitrogen
P	Chemical Element	1	Phosphorus
Val	Amino Acid	1	Valine
Hypothetical Protein. - Eschericia coli.	MSDB Protein	1	O05283_ECOLI
Hypothetical Protein. - Eschericia coli.	MSDB Protein	2	Q4FBC3_ECOLI

Table A.14: The `Attribute Metarules` Ruleform

BioEvent	*Seq_Nbr*	*BioEntity1*	*Attribute1*
Calculate Attribute	1	MoleculeType	Attribute Class
Calculate Attribute	2	MoleculeType	(SAME)

Table A.15: The `Attribute Protocol` Ruleform

BioEvent	*BioEntity1*	*Attribute1*	*Seq_Nbr*	*Method*	*BioEntity2*	*Attribute2*	*Class*
Calculate Attribute	Base Chemical, Molecule, Or Group	Mass	1	chemicalMass	ORIGINAL	ORIGINAL	core576.MassCalculator
Calculate Attribute	Composite Chemical, Molecule, Or Group	Mass	1	compositeChemicalMass	ORIGINAL	ORIGINAL	core576.MassCalculator
Calculate Attribute	Element Mass Type	Nominal Mass	1	elementNominalMass	ORIGINAL	ORIGINAL	core576.MassCalculator
Calculate Attribute	Element Mass Type	Average Mass	1	elementAverageMass	ORIGINAL	ORIGINAL	core576.MassCalculator
Calculate Attribute	Element Mass Type	Monoisotopic Mass	1	elementMonoisotopicMass	ORIGINAL	ORIGINAL	core576.MassCalculator

Table A.16: The `Transformation Metarules` Ruleform

BioEvent	*Seq_Nbr*	*Resource*	*BioEntity*
Protein Translation	1	(SAME)	(SAME)
Protein Translation	2	IsExceptionTo	(SAME)
DNA Replication	1	(SAME)	(SAME)
RNA Transcription	1	(SAME)	(SAME)
Reverse Transcription	1	(SAME)	(SAME)

Table A.17: The `Transformation Protocol` Ruleform

BioEvent	*Seq_Nbr*	*Resource*	*BioEntity*	*Relcode*
Protein Translation	1	Ciliate Nuclear Genetic Code	ORIGINAL	Signal Encoded
Protein Translation	2	Ciliate Nuclear Genetic Code	ORIGINAL	Amino Acid Encoded
Protein Translation	1	Universal Genetic Code	ORIGINAL	Signal Encoded
Protein Translation	2	Universal Genetic Code	ORIGINAL	Amino Acid Encoded
DNA Replication	1	Nucleic Acid Pairing	ORIGINAL	Complement
RNA Transcription	1	Nucleic Acid Pairing	ORIGINAL	Equivalent
Reverse Transcription	1	Nucleic Acid Pairing	ORIGINAL	Equivalent

Appendix B

Mass Spectrometry and Proteomics Background

B.1 Mass Spectrometry

Mass spectrometry is a technique for accurately determining the mass of a molecule. Originally conceived by J.J. Rutherford in 1899, mass spectrometry has grown into an extremely powerful analytical tool for biologists, as it allows detailed mass measurements to be taken on large protein molecules. These measurements can then be used in a variety of ingenious ways to identify and characterize proteins, a key goal of modern bioinformatics research.

Briefly, chemicals and molecules introduced into a mass spectrometer are given an electric charge through any of a number of ionization techniques. Often, the process of ionization will cause a molecule to fragment into a number of ions. A detector in the instrument then measures not the mass of the ions, but their mass-to-charge ratio m/z. Data is presented in the form of a graph, called a mass spectrum, with m/z on the x-axis and normalized ion detection intensity on the y-axis. Algorithmic techniques can be used to deconvolve the spectrum so that mass may be read directly. The spectrum produced by a molecule is generally unique to the molecule, and acts

as a "fingerprint" of sorts, enabling molecular identification on the basis of mass alone. Improved mass spectrometry instrumentation and data analysis techniques have resulted in the ability to obtain extremely high resolution mass measurements, which results in higher confidence identifications, as well as the ability to discriminate between different isotopes of an analyte.

B.2 Proteomics

Proteomics is the study of the entire protein complement of an organism, similar to to how the term genomics refers to the study of the entire genome of an organism. As proteins are the embodiment and instantiation of the instructions contained in a genome, their function and interrelationships determine virtually every aspect of an organism's development and function. Two key tasks of proteomics research are identification and characterization[1].

Given an unknown protein, the identification task seeks to answer the obvious question, what protein is this? Besides the pairing of a name to a protein, the identification task can provide additional important information about a protein. Tools such as GFS (Giddings et al. 2003) can help discover which gene encodes the protein in question. Additionally, one may want to know what proteins are similar to a given protein. This can be helpful both in identifying proteins as well as discovering what the function of a protein may be.

While knowing the identity of a protein is certainly important, other questions remain. The function of a protein can be altered dramatically depending on any modifications that may have been made to it. Post-translational modifications such as methylation and phosphorylation can act as signals and functional modulators, as can truncations. Additionally, a protein under study may be mutated in some way that affects its function. Alternatively spliced genes can give rise to a number

of related but distinct protein forms, each of which can have potentially different activities. Thus, elucidation of the precise state of a protein in the cell is extremely important and can reveal a wealth of information. This is the characterization task.

B.3 Mass Spectrometry-Based Proteomics

With the advent of soft-ionization techniques in the 1980s, mass spectrometry evolved from a technique used mainly by chemists into one amenable for use by biologists. Further refinements and advances over the years have led to mass spectrometry techniques becoming key tools for bioinformatics researchers. Mass spectrometry is a versatile technique for proteomics, and experimental approaches generally fall into one of two classes: bottom-up and top-down.

B.3.1 Bottom-Up Proteomics

The most common kind of mass spectrometry proteomics application is known as "bottom-up." In this approach, a protein sample is fragmented, generally through the use of proteases such as trypsin, and the resultant peptides are analyzed using tandem mass spectrometry. In this approach, once the mass of an individual peptide ion has been determined, that ion is subjected to an additional step of fragmentation, this time by e.g., bombardment with inert gas atoms. The masses of the ion fragments that are produced are determined by a second spectrum analysis (thus, "tandem" mass spectrometry). Since peptides generally break apart along the peptide backbone in this situation, a "ladder" is produced, yielding a series of successively longer ions, which differ by the mass of a single amino acid residue. By analyzing the spectra that are produced, the amino acid sequence of the peptides may be deduced. This method is also known as "shotgun" proteomics.

A peptide mass fingerprint (PMF) can be used to identify a protein, or at least specify a number of candidate proteins. Once spectrally-derived amino acid sequence has been obtained for peptides, it can be used to search sequence databases to help identify the protein. Software such as GFS eliminates the need to consult a sequence database altogether, as it can search an unannotated genome sequence to discover the genetic region that encoded the protein.

A strength of bottom-up proteomics is rapid identification of proteins that are present in a mixture. However, because measurements of the intact protein are not taken, any information reflecting the state of the protein in the biological milieu is lost.

B.3.2 Top-Down Proteomics

So-called "top-down" proteomics entails the use of a mass spectrometer to ascertain the mass of the complete, unfragmented protein, rather than of individual peptide fragments of the protein. This enables a researcher to obtain information on any changes in the observed protein, compared with theoretical predictions, including mutations, alternate splicings, post-translational modifications (PTM), and truncations. Such insight is of key importance in understanding a protein's function and regulation. Such information is lost with bottom-up approaches, because the association between a peptide and the original parent molecule is lost.

Tentative identification of a protein may be made using only top-down information. If the sequence of a protein is known, as is the case for proteins which have entries in a protein sequence database such as the Protein Data Bank, a theoretical mass may be computed by summing the masses of the constituent amino acids. An experimentally-derived mass can thus be compared to the theoretical masses of all of an organism's known proteins. A ranked list of possible identifications can be thus

obtained according to how close the experimental mass matches a theoretical mass. This can be helpful for characterizing any modifications to a protein. This task can be performed by the PROCLAME software (Holmes and Giddings 2004), which uses intact mass measurements as an aid for PTM prediction.

B.3.3 Integration of Bottom-Up and Top-Down Approaches

By utilizing both bottom-up and top-down approaches, a researcher can leverage the strengths of both, and reveal more comprehensive information on complex protein mixtures than is possible with either approach in isolation (Strader et al. 2004; Connelly et al. 2006). The basic idea proceeds thusly. A bottom-up analysis of a mixture of proteins is performed, yielding a list of protein identifications; this list is usually a subset of the proteins in the sample, due to variety of reasons, depending on the identification method used, the ionization efficiencies of the peptides, sequence coverage of the peptides, etc. Then, top-down analysis is carried out on another aliquot of the same sample, yielding intact masses for the proteins present. These masses reflect the state of the protein in the cell, including PTMs, mutations, different isoforms, and any other covalent modifications. By comparing the theoretical masses of the proteins identified via bottom-up methods to the intact masses obtained through the top-down approach, a characterization of the state of the protein may be able to be determined.

A protein with some PTM may be recognized via top-down but not by bottom-up if the site of the PTM is not contained in any of the fragment ions measured in the bottom-up experiment. For example, if a protein contains a phosphorylated serine at the fifth residue, yet the tandem MS produces coverage beginning at the, say, tenth residue, that fact will remain unobserved. A measurement of the intact protein mass will necessarily include this information. By combining approaches, identifications

of PTMs, as well as a pinpointing of their locations, may be made with greater confidence. Data from one approach can be used to iteratively refine information obtained by the other approach.

Currently, there are no information systems in use by the mass spectrometry community that facilitate this complex data integration task.

Notes

[1] A third main task involves quantification, but as the aim of the current project does not concern quantification, this point will not be further elaborated.

Bibliography

Baker, Patricia G., Carole A. Goble, Sean Bechhofer, Norman W. Paton, Robert Stevens, and Andy Brass. "An ontology for bioinformatics applications." *Bioinformatics* 15 (June 1999): 510–520. Available from: `http://bioinformatics.oxfordjournals.org/cgi/reprint/15/6/510`, DOI:10.1093/bioinformatics/15.6.510.

Blinov, Michael L., James R. Faeder, Byron Goldstein, and William S. Hlavacek. "BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains." *Bioinformatics* 20 (November 2004): 3289–3291. Available from: `http://bioinformatics.oxfordjournals.org/cgi/reprint/20/17/3289`, DOI:10.1093/bioinformatics/bth378.

Brooks Jr., Frederick P. *The Mythical Man-Month: Essays on Software Engineering.* 2nd edition. Addison-Wesley, 1995.

Connelly, Heather M., Eric Hamlett, Kevin Ramkissoon, Robert L. Hettich, and Morgan C. Giddings. Characterization of ribosomal proteins in two streptomycin resistant E. coli strains. Manuscript in preparation, 2006.

Faeder, James R., Michael L. Blinov, Byron Goldstein, and William S. Hlavacek. "Rule-Based Modeling of Biochemical Networks." *Complexity* 10 (March/April 2005): 22–41. Available from: `http://cellsignaling.lanl.gov/downloads/Complexity_2005.pdf`, DOI:10.1002/cplx.20074.

Gardner, Martin. "Mathematical Games — The fantastic combinations of John Conway's new solitaire game "Life"." *Scientific American* 223 (October 1970): 120–123. Available from: `http://ddi.cs.uni-potsdam.de/HyFISCH/Produzieren/lis_projekt/proj_gamelife/ConwayScientificAmerican.htm`.

Gene Ontology Consortium. "Gene Ontology: tool for the unification of biology." *Nature Genetics* 25 (May 2000): 25–29. Available from: `http://www.nature.com/ng/journal/v25/n1/pdf/ng0500_25.pdf`, DOI:10.1038/75556.

Giddings, Michael C., Atul A. Shah, Ray Gesteland, and Barry Moore. "Genome-based peptide fingerprint scanning." *Proceedings of the National Academy of Sciences* 100 (January 2003): 20–25. Available from: `http://www.pnas.org/cgi/reprint/100/1/20`, DOI:10.1073/pnas.0136893100.

Holmes, Mark R. and Michael C. Giddings. "Prediction of Posttranslational Modifications Using Intact-Protein Mass Spectrometric Data." *Analytical Chemistry* 76 (January 2004): 276–282. Available from: http://pubs.acs.org/cgi-bin/article.cgi/ancham/2004/76/i02/pdf/ac034739d.pdf, DOI:10.1021/ac034739d.

Hood, Leroy. "A Personal View of Molecular Technology and How It Has Changed Biology." *Journal of Proteome Research* 1 (October 2002): 399–409. Available from: http://pubs.acs.org/cgi-bin/sample.cgi/jprobs/2002/1/i05/pdf/pr020299f.pdf, DOI:10.1021/pr020299f.

HUPO PSI. *MIAPE: Mass Spectrometry*. 2005. Available from: http://psidev.sourceforge.net/gps/miape/MIAPE_MS_2.0.pdf.

HUPO PSI. The mzData standard [online]. 2005. Available from: http://psidev.sourceforge.net/ms/#mzdata.

Jacques, Pierre-Étienne, Alain L. Gervais, Mathieu Cantin, Jean-François Lucier, Guillaume Dallaire, Geneviève Drouin, Luc Gaudreau, Jean Goulet, and Ryszard Brzezinski. "MtbRegList, a database dedicated to the analysis of transcriptional regulation in Mycobacterium tuberculosis." *Bioinformatics* 21 (2005): 2563–2565. Available from: http://bioinformatics.oxfordjournals.org/cgi/reprint/21/10/2563, DOI:10.1093/bioinformatics/bti321.

Kiebel, Gary R., Ken J. Auberry, Navdeep Jaitly, David A. Clark, Matthew E. Monroe, Elena S. Peterson, Nikola Tolić, Gordon A. Anderson, and Richard D. Smith. "PRISM: A data management system for high-throughput proteomics." *Proteomics* 6 (March 2006): 1783–1790. Available from: http://www3.interscience.wiley.com/cgi-bin/fulltext/112402178/PDFSTART, DOI:10.1002/pmic.200500500.

Laszlo, Ervin. *The Systems View of the World: A Holistic Vision for Our Time*. Cresskill, New Jersey: Hampton Press, Inc., 1996.

Lee, Thomas J., Yannick Pouliot, Valerie Wagner, Priyanka Gupta, David W. J. Stringer-Calvert, Jessica D. Tenenbaum, and Peter D. Karp. "BioWarehouse: a bioinformatics database warehouse toolkit." *BMC Bioinformatics* 7 (2006). Available from: http://www.biomedcentral.com/1471-2105/7/170, DOI:10.1186/1471-2105-7-170.

Lo, Siaw Ling, Tao You, Qingsong Lin, Shashikant B. Joshi, Maxey C. M. Chung, and Choy Leong Hew. "SPLASH: Systematic proteomics laboratory analysis and storage hub." *Proteomics* 6 (2006). Available from: http://www3.interscience.wiley.com/cgi-bin/abstract/112395657/ABSTRACT, DOI:10.1002/pmic.200500378.

Long, Jeffrey G. "How could the notation be the limitation?" *Semiotica* 125 (1999): 21–31.

Long, Jeffrey G. "A new notation for representing business and other rules." *Semiotica* 125 (1999): 215–227.

Long, Jeffrey G. The Ultra-Structure of Rules. Unpublished manuscript, 2005.

Long, Jeffrey G. and Dorothy E. Denning. "Ultra-Structure: A Design Theory for Complex Systems and Processes." *Communications of the ACM* 38 (January 1995): 103–120. Available from: `http://portal.acm.org/citation.cfm?doid=204865.204892`, DOI:10.1145/204865.204892.

Mao, Chunhong, Jing Qiu, Chunxia Wang, Trevor C. Charles, and Bruno W. S. Sobral. "NotMutDB: a database for genes and mutants involved in symbiosis." *Bioinformatics* 21 (June 2005): 2927–2929. Available from: `http://bioinformatics.oxfordjournals.org/cgi/reprint/21/12/2927`, DOI:10.1093/bioinformatics/bti427.

Martens, Lennart, Henning Hermjakob, Philip Jones, Marcin Adamski, Chris Taylor, David States, Kris Gevaert, Joël Vandekerckhove, and Rolf Apweiler. "PRIDE: The proteomics identifications database." *Proteomics* 5 (August 2005): 3537–3545. Available from: `http://www3.interscience.wiley.com/cgi-bin/fulltext/110573390/PDFSTART`, DOI:10.1002/pmic.200401303.

Open Biomedical Ontologies [online]. 2006. Available from: `http://obo.sourceforge.net/`.

Overgard, Gary. "An Object-Oriented variation on Ultra-Structure." *Semiotica* 125 (1999): 187–195.

Pappin, Darryl J. C. and David N. Perkins. *Mass Spectrometry protein sequence DataBase*. 2005. Available from: `http://csc-fserve.hh.med.ic.ac.uk/msdb.html`.

Perkins, David N., Darryl J. C. Pappin, David M. Creasy, and John S. Cottrell. "Probability-based protein identification by searching sequence databases using mass spectrometry data." *Electrophoresis* 20 (1999): 3551–3567. Available from: `http://www3.interscience.wiley.com/cgi-bin/abstract/68500773/ABSTRACT`, DOI:10.1002/(SICI)1522-2683(19991201)20:18<3551::AID-ELPS3551>3.0.CO;2-2.

Prickett, Dennis, Matt Page, Angela E. Douglas, and Gavin H. Thomas. "BuchneraBASE: a post-genomic resource for Buchnera sp. APS." *Bioinformatics* 22 (March 2006): 641–642. Available from: `http://bioinformatics.oxfordjournals.org/cgi/reprint/22/5/641`, DOI:10.1093/bioinformatics/btk024.

Shostko, Alexander. "Design of an automatic course-scheduling system using Ultra-Structure." *Semiotica* 125 (1999): 197–213.

Strader, Michael Brad, Nathan C. VerBerkmoes, David L. Tabb, Heather M. Connelly, John W. Barton, Barry D. Bruce, Dale A. Pelletier, Brian H. Davison,

Robert L. Hettich, Frank W. Larimer, and Gregory B. Hurst. "Characterization of the 70S Ribosome from Rhodopseudomonas palustris Using an Integrated "Top-Down" and "Bottom-Up" Mass Spectrometric Approach." *Journal of Proteome Research* 3 (October 2004): 965–978. Available from: `http://www.ornl.gov/sci/GenomestoLife/pubs/pr049940z.pdf`, DOI:10.1021/pr049940z.

Taylor, Chris F., Norman W. Paton, Kevin L. Garwood, Paul D. Kirby, David A. Stead, Zhikang Yin, Eric W. Deutsch, Laura Selway, Janet Walker, Isabel Riba-Garcia, Shabaz Mohammed, Michael J. Deery, Julie A. Howard, Tom Dunkley, Ruedi Aebersold, Douglas B. Kell, Kathryn S. Lilley, Peter Roepstorff, John R. Yates III, Andy Brass, Alistair J. P. Brown, Phil Cash, Simon J. Gaskell, Simon J. Hubbard, and Stephen G. Oliver. "A systematic approach to modeling, capturing, and disseminating proteomics experimental data." *Nature Biotechnology* 21 (March 2003): 247–254. Available from: `http://www.nature.com/nbt/journal/v21/n3/pdf/nbt0303-247.pdf`, DOI:10.1038/nbt0303-247.

VerBerkmoes, Nathan C., Jonathan L. Bundy, Loren Hauser, Keiji G. Asano, Jane Razumovskaya, Frank Larimer, Robert L. Hettich, and James L. Stephenson Jr. "Integrating "Top-Down" and "Bottom-Up" Mass Spectrometric Approaches for Proteomic Analysis of Shewanella oneidensis." *Journal of Proteome Research* 1 (June 2002): 239–252. Available from: `http://pubs.acs.org/cgi-bin/article.cgi/jprobs/2002/1/i03/pdf/pr025508a.pdf`, DOI:10.1021/pr025508a.

Weinberg, Gerald M. *An Introduction to General Systems Thinking*. Silver anniversary edition. New York: Dorset House Publishing Company, Inc., 2001.

Wilkinson, Mark D. and Matthew Links. "BioMOBY: An open source biological web services proposal." *Briefings in Bioinformatics* 3 (December 2002): 331–341. Available from: `http://bib.oxfordjournals.org/cgi/reprint/3/4/331`, DOI:10.1093/bib/3.4.331.