BIOINFORMATICS ONTOLOGY

Vol. 16 no. 12 2000
Pages 1129–1144

Semiotes: a semantics for sharing

Toni Kazic [1]

[1]Institute for Biomedical Computing, Washington University, 700 South Euclid Avenue, St. Louis, MO 63110, USA

Received on April 5, 1999; revised on November 11, 1999 and June 1, 2000; accepted on June 18, 2000

Abstract

Motivation: *Reliable, automated communication of biological information requires methods to declare the information's semantics. In this paper I describe an approach to semantic declaration intended to permit independent, distributed databases, algorithms, and servers to exchange and process requests for information and computations without requiring coordination or agreement among them on universe of discourse, data model, schema, or implementation.*

Results: *This approach uses* Glossa*, a formal language defining the semantics of biological ideas, information, and algorithms, to* executably *define the semantics of complex ideas and computations by constructs of* semiotes*, terms which axiomatically define very simple notions. A database or algorithm wishing to exchange information or computations maintains a set of mappings between its particular notions and semiotes, and a parser to translate between its indigenous ideas and implementation and the semiotes. Requests from other databases or algorithms are issued as semiotic messages, locally interpreted and processed, and the results returned as semiotes to the requesting entity. Thus, semiotes serve as a shared, abstract layer of definitions which can be computably combined by each database or algorithm according to its own needs and ideas. By combining the explicit declaration of semantics with the computation of the semantics of complex ideas,* Glossa *and its semiotes permit independent computational entities to lightly federate their capabilities as desired while maintaining their unique perspectives on both scientific and technical questions.*

Availability: *http://www.ibc.wustl.edu/agora/semiotes/ and files therein.*

Contact: *toni@athe.wustl.edu*

Why do semantics matter?

Whenever one entity attempts to communicate with another, the message's utility is determined by how well the recipient understands it. This is obvious when the entities are two people conversing, but it is equally true when a program processes data it has received from a database; or one interfaces two or more independent pieces of code; or one enters information into a database. In each instance, a symbol of one or more characters is used to denote an abstraction—a scientific, mathematical, or computational idea, datum, or result. But the symbol *qua* symbol is mute: it doesn't say what abstraction it denotes. To know which abstraction a symbol denotes—formally, to map the symbol to its denoted semantics—requires an entity that knows the language accurately enough to define the symbol's semantics and is vigilant enough to detect and correct any errors of usage. When the choice of abstraction is uncertain or the mapping is not one-to-one, then the semantics of a symbol are ambiguous. This ambiguity is independent of the symbol's syntax: knowing a symbol is an array, a regular expression, or a CORBA method will not help one divine its meaning.

As long as computational entities—programs, databases, knowledge bases, and servers—do not communicate with each other or with human beings, semantic ambiguities are isolated and relatively inconsequential. But communicate, and each entity must determine which abstractions are denoted by the other's symbols and map the denoted abstraction to its notions, data structures, algorithms, and schema. If the mapping is inaccurate or incomplete, the accuracy of any computation involving more than one entity cannot be assured. (Obviously a computation's accuracy is independent of its completion or efficiency.) The ambiguities intensify when one moves from simple to more complex universes of discourse[†], such as from electronic commerce to biology. Semantic issues arise whether the goal is to build an integrated database that subsumes others or a looser federation intended only to provide interentity communications. The most interesting abstractions are those describing the science, not the mechanics of database arrangement or query execution. All too often, however, the most clearly defined semantics are for notions of the latter sort.

The importance of modern biological questions have

[†] By 'universe of discourse' I mean all the ideas and terms relevant to an area as modeled by a database or algorithm. One sometimes sees 'domain' used as a synonym for 'universe of discourse'. In this paper I will restrict the use of 'domain' to its mathematical usage (loosely, the set on which a function operates).

stimulated the production of many sophisticated databases and algorithms. One naturally expects these resources can jointly address those questions. Thus the challenge is plain: *people, databases, and algorithms must reliably and automatically define the semantics of their biological universes of discourse in a mutually intelligible way.* The penalty for unreliable definitions is the proliferation of scientifically meaningless (or worse, misleading) results, which without additional inspection are indistinguishable from meaningful ones. The penalty for nonautomatic methods is an insatiable demand for these human inspectors. The penalty for mutual unintelligibility is a very real tower of automated pseudoscientific Babel.

But the technical means to meet the challenge are much less clear. Any attempt at defining the semantics of a scientific universe of discourse faces four fundamental problems. One may summarize these as determining correctness, managing controversy, expressing complexity, and choosing the computational means.

First, humans are the ultimate arbiters of correctness for language for now, so human oversight is essential to any definitional effort. Second, deciding what something is really is hard: humans and their computational creations often disagree about a term's semantics for valid scientific, experimental, or technical reasons. To reify a controversial term one must either decide the controversy arbitrarily or suspend the term's implementation in code until the controversy is someday resolved. The first course means some usages of the term will not agree with the proposed definition, introducing errors and diminishing the expressivity, portability, and utility of the semantics. The second course means an indefinite delay in implementation. Third, the relationships among biological entities—e.g. molecules, physiological processes, ecosystems—are structurally complex. They vary with the particular biological entities and over time, often as noncontinuous functions; they are mechanistically intricate; and they incorporate many types of relationship, from geometric adjacency to systems of nonlinear differential equations. Many of these relationships cannot be reduced to set membership and subsumption (e.g. *part—whole* or *isa*). Indeed, the most relevant relationship between two entities in a particular instance will often not be related to sets at all. This structural complexity places a premium on the expressivity of the semantics, but in turn increases the number of terms. Finally, the short history of biological databases does not stimulate much optimism that there will be universal agreement on any but the simplest technical and epistemological standards. While *implementing* a computable semantics necessarily entails local technical choices, the *specification* of a computable semantics should depend on neither a particular technical apparatus nor data models.

Not surprisingly, defining the semantics of a universe of discourse is a very durable problem. One may crudely divide these efforts into three sometimes overlapping categories: those that attempt to build data models, software systems or languages for 'managing' terms; those that attempt to define semantics for particular universes of discourse; and those that focus on generic tools for the federation of multiple databases.

Efforts to build software for ontologies are in the first category. Examples include Ontolingua and its companion Reusable Ontologies (Knowledge Systems Laboratory, 1999); OML (Ontology Markup Language, Kent, 1999); GKB (Generic Knowledge Base Editor, which is specific to frame-based systems, Chaudhri and Lowrance, 1999); and XOL (Ontology Exchange Language, Karp *et al.*, 1999). (See Clark, 1999 for an on-line bibliography of similar projects, many based on Sowa's original notion of conceptual graphs, (Sowa, 1984), and Abernethy, 1999 for references to some biologically-inspired ones.) Schulze-Kremer's Ontology Editor combines term management with a more scientifically relevant ontology (Schulze-Kremer, 1996). Since ontologies aim to classify terms, rather than define them (Sowa, 1998), in most instances definitions *per se* do not exist. Given a (natural language) definition and an instance of usage, human inspection seems to be required to determine the correctness of the usage. For example, OML defines a syntax that allows one to mark up data as belonging to particular classes, and uses the conceptual graph formalism to provide a theoretical framework for the classes and the λ calculus to describe the graph. But apart from identifying that an entity is an instance of a class, there is no definition of the class *per se*. The terms and relationships used as examples are quite general, thereby minimizing controversy, specificity, and scientific utility. When definitions are included, they and their relationships are often separated from terms; for example, Ontolingua separates vocabulary, the model of the universe of discourse, and schema. Thus many of the elements required to make a definition computationally executable are diffused among different programs. Finally, each system lays specific technical bets which restrict portability. Though XOL attempts to minimize this problem, it does rely on an XML apparatus (Karp *et al.*, 1999).

In the second category are attempts to define the semantics of particular universes of discourse. Examples include controlled vocabularies or metathesauri for macromolecular structure (mmCIF, Bourne *et al.*, 1997; IUCr Working Party on Crystallographic Information, 1999); medical information (UMLS, Nelson *et al.*, 1993; Galen, Goble *et al.*, 1994); *Drosophila melanogaster* (FlyBase, The FlyBase Consortium, 1999a,b); genomics (GDB, Genome Data Base Staff, 1996); ontologies for experiments on ribosome (RiboWeb, Altman and Abernethy, 1997) and macromolecular structure (BioML, Proteometrics Inc.,

1999); and a semantic grammar for sentences found in immunology papers (Harris *et al.*, 1989). Each relies on humans to interpret the semantics, ranging from completely human-oriented (e.g. Flybase) to some definitions one might guess from the actions of extrinsic programs (e.g. mmCIF, CML, RiboWeb). Usually the only semantic clue available is an hierarchical arrangement of terms denoting subset relationships, making it harder to decide if a usage is correct. Moreover, hierarchical structures have a subtle but important impact on a semantics' granularity: notions are often fused just by being nested together in the tree. This produces a poorly resolved and sometimes ambiguous semantics. When coarse granularity collides with complex notions, the outcome is frequently controversy. For example, the formal semantic grammar of immunology by Harris *et al.* is complex and precise enough to minimize the problem of determining correctness (Harris *et al.*, 1989). But the granularity of its terms is too coarse to prevent controversy. Similarly, coarse granularity can produce problems in deducing a label's semantics from the actions of a program. For example, Murray-Rust's CML (Chemical Markup Language) relied heavily on computational chemistry programs spawned upon the recognition of particular file extensions to define the semantics of the labels (Murray-Rust, 1995). Yet the semantics of many such programs are extremely complex, both because of their underlying science (quantum and molecular mechanics) and the possible variations in the parameters, conditions, and methods for a given calculation. While extrinsic programs can express very complex semantics, they cannot be relied upon to reveal that semantics without significant human intervention.

The third category is the attempts to devise generic methods for federating heterogeneous and distributed database systems. The earliest work, summarized in Sheth and Larson (1990), required significant schema uniformity and identical database management systems and query languages among federation members. The result is a mandatory implicit semantics for an arbitrary universe of discourse. Though this approach has been used in some instances for biological databases (for example, see Ritter, 1994 and references therein), in practice agreement on the components—semantics, data model, and schema—has often proven difficult to obtain and maintain, even for the relatively simple semantics of the commercial and financial sectors. So considerable effort has been devoted to methods that are useful even when these conditions are not met. (For a review, see Elmagarmid *et al.*, 1999; the following citations are merely exemplars, not a comprehensive survey.) The main efforts have been in schema declaration, mapping, and reconciliation schemes (Chen *et al.*, 1997; Seo *et al.*, 1997; Bellahsene, 1997; Ram, 1991; Topaloglou *et al.*, 1999); multidatabase query languages and optimization (Mihaila *et al.*, 1998; Chen *et al.*, 1997; Wu, 1996; Missier *et al.*, 1999; Hameurlain and Morvan, 1996; Ounis and Chevallet, 1996; Lee and Yoo, 1994; Hammer *et al.*, 1997; Li *et al.*, 1997; Papakonstantinou *et al.*, 1995); and semantic mapping and conversion (Huemer *et al.*, 1997; Kashyap and Sheth, 1999; Hammer and McLeod, 1999; Goble *et al.*, 1994). In the first two categories, the focus of research is on the database machinery *assuming* the semantics of the participating databases have been manually (or simplistically; e.g. Lee and Yoo, 1994) resolved. Thus the semantics treated are those of the database models and machinery, not the universe of discourse (Goble *et al.*, 1994; Houstis *et al.*, 1994, being exceptions). Occasionally one sees methods to resolve manually mapped terms, usually relying on the semantics of a particular data model. Thus OPM's definition of the semantics of the databases that will be converted to its data model includes considerable information on that database's tables, classes, attributes, and queries (Chen *et al.*, 1997), and TSIMMIS uses essentially the same strategy (Hammer *et al.*, 1997; Li *et al.*, 1997; Papakonstantinou *et al.*, 1995). There are also fledgling efforts to automatically resolve a semantics declared in natural language (Kashyap and Sheth, 1999). Here too terms from each database are usually manually mapped to each other, forming all possible pairs of terms; then a probabilistic guess of the best mapping is computed. The sophistication of these systems rests, not in their declaration or use of semantics, but in their attempts to *infer* the semantics of a novel term or construct. The terms themselves are quite simple, especially as compared to biology, and it is an open question how well these systems could infer the semantics of biological terms.

However, there are two fields that routinely manipulate or define semantics: linguistics and the theory of formal languages, which includes programming languages. In linguistics, many of the semantic definitions are in human natural language (for example Eco, 1984; Lakoff, 1987); formal work has been directed at disambiguating extant natural language (Aronoff and Rees-Miller, 2000). There have been many computational attempts to extract, and to some extent, 'understand' phrasal semantics as part of projects to extract information from *corpora* of texts. This area has recently blossomed for biologically related texts (Baclawski *et al.*, 1993; Futrelle, 1997; Craven and Kumlien, 1999; Blaschke *et al.*, 1999; Rindflesch *et al.*, 2000; Salton *et al.*, 1993; Stapley and Benoit, 2000; Thomas *et al.*, 2000; Lehnert, 2000; Baclawski *et al.*, 2000; Message Understanding Conference, 1992; Humphreys *et al.*, 2000). The semantic constructs are too coarsely grained for computational exchange and require further resolution, presumably because they are simply taken as they are in the natural language *corpora*. Conversely, the semantics of terms and con-

structs is explicitly defined and composed in the case of formal languages (for example Meyer, 1990). However in this case the universe of discourse of the language is extremely limited: even a very expressive programming language is semantically trivial compared with modern biology. Nonetheless, formal languages offer a potentially powerful set of methods for defining and implementing the semantics of complex constructs, provided the basis of the language is well defined both formally and in terms of the semantics of the particular universe of discourse.

Here I outline a new approach to defining the semantics of scientific knowledge for use by computational entities. The goal of this work is to enable scientifically reliable computations distributed over many participating entities, without constraining the participants' semantics, schema, or computational machinery. Its fundamental premises are that the only useful semantics for machines is a computable one, and that controversy about ideas signals a healthy scientific discourse and should not be artificially constricted. This approach uses a formal language to declare the semantics of biological ideas and computations by providing *computable* definitions built from terms whose own semantics are so simple as to be axiomatic. These definitions are executable computations, not texts in a natural language. I call this language *Glossa* (the Greek for 'tongue' in multiple senses), and the symbols denoting the terms, *semiotes* (a neologism reflecting their debt to semiotics while avoiding the controversies native to the terms *seme*, *sign*, and others, Eco, 1984). The semiotes and their semantics are required to be unique, disjoint, and elementary. The set of semiotes, called the *semantic basis set*, serves as an abstract layer shared by the participants. Each resource translates only between itself and the basis set by using mappings and parsers between the basis set and its local implementation, not to each of the other resources. The effect is to lightly federate the participating resources for that subset of computations and information each shares.

Glossa aims to minimize the problem of correctness by making semantics computable; to avoid the problems of controversy and complexity altogether by providing simple, finely grained terms which can be flexibly combined to express complex ideas; and to avoid the problem of computational choices by stating definitions declaratively and avoiding extrinsic definitions. The choices of implementation are left to each computational entity participating in a very lightly federated consortium. Semiotes were sketched in Kazic (1995), and *Glossa* is being implemented as part of *The Agora*, an infrastructure for the transparent sharing of curatorial and computational functions among biological databases (Bugrim *et al.*, 2000).

The rest of the paper is organized as follows. The next section sketches the formal bases of *Glossa*. The main result is that *Glossa* is semantically and computationally well behaved if certain conditions are met. The following section illustrates several semiotes and bundles used in *The Agora* to describe reaction equations, and also gives some examples of what semiotes are not and why. The penultimate section briefly describes the use of *Glossa* in very lightly federating participating servers; and the last section discusses some final points.

An outline of *Glossa*

To begin the exposition I formally define the basics of *Glossa*, especially its notion of semantics. I then prove a theorem about the semantics of constructs in *Glossa*, and finally describe its most basic terms, the semiotes. Formal apparatus is important for three reasons. It forces us to be as clear as possible in our use of terms; it provides a standard against which one can judge whether the language has the necessary properties; and it enables automatic evaluation of the semantics of a specific computation. This section should be regarded only as a current statement of work in progress.

Let *Glossa* be a formal language describing a biological universe of discourse, including operations in it, implemented in a software system. For example, the subset of *Glossa* describing a taxonomic database would include symbols both for the phylogenetic ideas and the abstract operations on the notions in the database. Like all other formal languages, *Glossa* has a finite, discrete set of symbols it uses, Σ, and a set of rules Π for combining those symbols into 'sensible' constructs. The individual members of these sets are denoted σ and π, respectively, with subscripts as needed. Σ includes symbols for substantives, operations, and modifiers (crudely, nouns, verbs, and {adjectives, adverbs, prepositions}). It also includes the empty symbol, $\sigma_\emptyset$: when concatenated to the left or right of any other symbol it leaves the other symbol unaltered.

It is useful to distinguish several different subsets of Σ, which fall into three groups. The first group is the subsets encompassing the semiotes and nonsemiotic symbols (Σ_ς and its complement taken over Σ, Σ'_ς). Any symbol may be a member of either Σ_ς or Σ'_ς, but not both. The second group is the subsets encompassing symbols whose semantics are those of, or derived from, the predicate calculus (of any form); mathematical functions; and procedural statements intended to control execution of an expression or communicate with the shell or other processes. For brevity I'll call these the logical, functional, and procedural classes of symbols (Σ_l, Σ_f, and Σ_p respectively), and refer to these subsets as 'classes of symbols'. Any symbol can be a member of only one of Σ_l, Σ_f, and Σ_p, though it is not required to be a member of

any. This last condition forms the third group of symbols, the 'classless' ones (Σ_u). Classless symbols are provided so that symbols denoting the same variable can be used in any context—e.g. X can occur in logical expressions (`not(X)`), mathematical functions (`X = sin(omega * t)`), and procedural statements (`while X < $N do`). Thus the set relationships are

$$\emptyset = \Sigma_l \cap \Sigma_f \cap \Sigma_p \cap \Sigma_u \tag{1}$$

$$\emptyset = \Sigma_\varsigma \cap \Sigma'_\varsigma \tag{2}$$

and

$$\Sigma = \Sigma_l \cup \Sigma_f \cup \Sigma_p \cup \Sigma_u \tag{3}$$

$$\Sigma = \Sigma_\varsigma \cup \Sigma'_\varsigma. \tag{4}$$

Glossa has a set of rules, Π, for organizing symbols into constructs and interpreting those constructs. These rules correspond to the production rules of a grammar. *Glossa*'s Π is currently described by a left-recursive, context-free Backus–Naur Form grammar (Kazic *et al.*, 2000). Ω is the set of semantic mappings. The members of Ω and Π operate on the members of the symbol set Σ, and each mapping (ω, indexed as needed) and rule (π) is one-to-one and onto. But just as the function $\sin(x)$ is not defined if $x =$ true, so too not every mapping or rule is defined for every member of Σ. Therefore every member of Ω and Π is a partial finite function, its domain restricted to those members of Σ for which it is defined. $\longmapsto$ indicates the mapping operation and is used in preference to $\longrightarrow$ to avoid confusion with biochemical reactions. Applied to the sets, the notation $\Omega\colon \Sigma \longmapsto \Omega(\Sigma)$ should be interpreted as 'the set of mappings, each of which is defined for at least one member of Σ and produces one and only one member of $\Omega(\Sigma)$ when applied to that member of Σ' (and analogously for Π).

First I define *Glossa*'s notion of semantics.

DEFINITION 1. For all $\sigma_j \in \Sigma$, j a positive integer index, its *semantics*, $\omega(\sigma_j)$, is a computationally executable definition of σ_j's meaning. It is found or produced by applying a semantic mapping, ω to σ_j, denoted $\omega\colon \sigma_j \longmapsto \omega(\sigma_j)$, such that ω is one-to-one, onto, and defined for that σ_j. Under these conditions, we call both symbol and mapping *semantically well-formed*.

If for each $\sigma_j \in \Sigma$ there is at least one ω that is semantically well-formed, then we say the *term semantics* of $\Omega\colon \Sigma \longmapsto \Omega(\Sigma)$ are well-formed, where Ω is the set of all semantically well-formed mappings operating on Σ and $\Omega(\Sigma)$ is the set of defined semantics for the members of Σ.

For the computer to take a symbol and test or derive its meaning, *Glossa* must be unambiguous. So the first requirement is that *Glossa* have well-formed term semantics. The definition gives tests for new terms (symbols), mappings, and their semantics: the semantics must be computationally executable and produce a unique definition for each term for which a particular mapping is defined. This unambiguity does not imply that the definition of a term's semantics must exclude conditional or disjoint statements: it simply says there can be only one definition of a symbol, in contrast to human natural languages. (The distinction is the difference in the definitions of 'bank' (multiple nouns and verbs) and 'read' (a noun, and two parts of the same verb) *versus* 'several' (either three, or four, or five objects). The last would be a single definition with disjoint statements or a range, while the other two terms would not have a single definition.)

To determine the semantics of constructs of symbols requires a bit more apparatus. To simplify the notation I first define *expressions*.

DEFINITION 2. An *expression* ε is a delimited sequence of syntactically and semantically well-formed symbols which either belong to only one class of symbols or are classless. An expression may also include any mappings $\omega \in \Omega$ or rules $\pi \in \Pi$ defined for the symbols in the sequence, either individually or taken together as a tuple. An expression is delimited by the prefix and postfix logical operators $\lceil$ and $\rceil$ to indicate its boundaries. Let $\langle \sigma_1, \sigma_2, \ldots, \sigma_j \rangle$ denote the sequence of symbols inside the delimiters for any particular expression, where j is a positive integer indexing the symbols of that sequence. Then

$$\varepsilon = \lceil \langle \sigma_1, \sigma_2, \ldots, \sigma_j \rangle \rceil \tag{5}$$

where each σ_j in the sequence is a member of only one of Σ_l, Σ_f, or Σ_p; or is a member of Σ_u. The class of an expression is that of its classed symbols. The semantics of an expression, $\omega(\varepsilon)$, for an ω defined for each symbol in the sequence, is

$$\omega(\varepsilon) = \omega(\lceil \langle \sigma_1, \sigma_2, \ldots, \sigma_j \rangle \rceil). \tag{6}$$

Since the only requirement is that the symbols within an expression be from the same symbol class or be classless, the expressions can be quite long and complex. Each expression is wrapped in a logical tissue, and substitution of a variable for an expression allows for the use of meta-expressions (i.e. expressions using or about other expressions). These two rules (logical delimiters and substitution) are of course members of Π.

Now we are ready to build big constructs.

DEFINITION 3. A *construct* $\mathfrak{c}$ in *Glossa* is a combination of one or more expressions formed by at least two rules drawn from Π. The first rule is that every pair of expressions is separated by a logical infix operator. The other rule or rules specify the syntax of the construct, and

can include rules which prefix and postfix the entire construct with logical operators or delimiters. Let $\bullet_i$ denote the separating logical operators, and let i index the operators, k the expressions, and j the symbols, each index independent of the others and all ranging over the positive integers. Then (exhibiting the most general form)

$$\mathfrak{c} \equiv \bullet_0\, \varepsilon_1\, \bullet_1\, \varepsilon_2\, \bullet_2\, \cdots\, \bullet_{i-1}\, \varepsilon_k\, \bullet_i\, . \tag{7}$$

The set of all such constructs $\mathfrak{c}$ which can be formed from Σ by applying all members of Π defined for the symbols in $\mathfrak{c}$ (taken individually or jointly according to syntax) is denoted $\mathfrak{C}$ ($\mathfrak{C} = \Pi(\Sigma)$); symbolically,

$$\Pi\colon \Sigma \longmapsto \mathfrak{C}. \tag{8}$$

For example,

$$\mathfrak{c} = (\varepsilon_1 \wedge (\varepsilon_2 \vee \varepsilon_3)) \tag{9}$$

where

$$\varepsilon_1 \equiv \lceil y = mx + b \rceil \tag{10}$$

$$\varepsilon_2 \equiv \lceil m = 5 \rceil \tag{11}$$

$$\varepsilon_3 \equiv \lceil b = 7 \rceil, \tag{12}$$

giving a functional expression to evaluate under two different boundary conditions.

The grammatical rules Π ensure that every construct has a framework of logical expressions, each of which in turn may conceal multiple functional or procedural expressions. This permits one to operate on the whole with the various predicate calculi so long as functional and procedural expressions retain their native operations and semantics. The scope of expressions is specified by their logical delimiters, and can range from the entire construct to a single symbol. The definition does not distinguish among different types of expressions, nor does it restrict the form the construct can take (in particular, it is not constrained to be a string). For notational convenience I have lumped together delimiters, such as (,), [, and], conditional operators ('if x then y'), set operators, etc., so that a construct which looks like Equation (7) could easily conceal extensive nesting, conditionals, and other Baroque syntactic structures. The formally inclined will recognize that $\mathfrak{c}$ and its components are simply special cases of S-expressions (Meyer, 1990).

What are the semantics of constructs in *Glossa*? First a useful lemma, then the main theorem.

LEMMA 1. Given any two mappings, ω_l and $\omega_{l'}$, defined for an expression ε (l a positive integer index, $l \neq l'$), such that

$$\omega_l\colon \varepsilon \longmapsto \omega_l(\varepsilon) \tag{13}$$

$$\omega_{l'}\colon \omega_l(\varepsilon) \longmapsto \omega_{l'}(\omega_l(\varepsilon)) \tag{14}$$

and the domain of $\omega_{l'}$ is the range of ω_l, the composition of $\omega_{l'}$ and ω_l, $\omega_{l'} \circ \omega_l$, maps

$$\omega_{l'} \circ \omega_l\colon \varepsilon \longmapsto \omega_{l'}(\omega_l(\varepsilon)). \tag{15}$$

The proof is by induction and will not be elaborated (Meyer, 1990; James and Beckenbach, 1968). The effect is to provide a path over the semantics of each successive substitution in an expression, for those semantic mappings that are defined for the expression and the results.

THEOREM 1. The semantics of $\mathfrak{c}$ is the composite of the semantics defined for its symbols, for all $\mathfrak{c} \in \mathfrak{C}$. Taking the most general case, if

$$\mathfrak{c} = \bullet_0\, \varepsilon_1\, \bullet_1\, \varepsilon_2\, \bullet_2\, \ldots\, \bullet_{i-1}\, \varepsilon_k\, \bullet_i \tag{16}$$

then denoting the semantics of $\mathfrak{c}$ as $\omega_n(\mathfrak{c})$,

$$\omega_n(\mathfrak{c}) = \omega_n(\bullet_0\, \varepsilon_1\, \bullet_1\, \varepsilon_2\, \bullet_2\, \ldots\, \bullet_{i-1}\, \varepsilon_k\, \bullet_i) \tag{17}$$

$$= \omega_n(\bullet_0(\omega_{n-1}(\varepsilon_1(\omega_{n-2}(\bullet_1(\omega_{n-3}(\varepsilon_2(\omega_{n-4}(\bullet_2(\ldots\omega_3(\bullet_{i-1}(\omega_2(\varepsilon_k(\omega_1(\bullet_i))))) \ldots)))))))))) \tag{18}$$

$$= \omega_1 \circ \omega_2 \circ \omega_3 \circ \ldots \circ \omega_{n-4} \circ \omega_{n-3} \circ \omega_{n-2} \circ \omega_{n-1} \circ \omega_n(\bullet_0\, \varepsilon_1\, \bullet_1\, \varepsilon_2\, \bullet_2\, \ldots\, \bullet_{i-1}\, \varepsilon_k\, \bullet_i), \tag{19}$$

where each ω is defined for the symbols (individually or jointly, depending on syntax) on which it operates, and the domain of ω_{l+1} is equal to the range of ω_l, for each such pair, l a positive integer indexing ω, $1 \leqslant l \leqslant n$.

Under these conditions the semantics of $\mathfrak{c}$ are well-formed. If there is a mapping $\omega \in \Omega$ that is semantically well-formed for each $\mathfrak{c} \in \mathfrak{C}$, then we say the *constructional semantics* of *Glossa* are well-formed and write $\Omega\colon \mathfrak{C} \longmapsto \Omega(\mathfrak{C})$.

The proof of the theorem is by induction, relying on the identity of ε and S-expressions and on the lemma (Meyer, 1990). Briefly, there are two cases to consider: the first is if the construct consists entirely of logical symbols, and the second is if it includes nonlogical symbols. In the first case, one simply has the predicate calculus, its standard operations, and constructs built on the predicate calculus. The semantics of the predicate calculus and its operations are well-defined, and the semantics of logical constructs are defined by the executable (predicate calculus) code specifying the construct, *per* Definition 3. So the semantics of the first case are defined *per* the theorem. For the second case, one has nonlogical statements embedded in the framework of the predicate calculus. The semantics of the framework are defined, so the question is what are the semantics of the nonlogical expressions? These are simply the functional or procedural semantics of those expressions, which are

also well-defined. Thus the semantics of the second case are also defined.

There's no mystery about either the theorem or its proof—real programming languages mix logical, mathematical, and procedural statements while retaining a well defined semantics. This is simply what *Glossa* has been carefully defined to do.

Now that we understand what semantics are for both symbols and constructs, we can formally define the set of symbols having the most basic semantics, the semiotes.

DEFINITION 4. A *semiote* is a symbol denoting the semantics of a useful elementary part of an idea, datum, or computation.

More formally, let a symbol in *Glossa* be denoted σ_0 and the set of all symbols in *Glossa* other than σ_0 be denoted Σ', or $\Sigma' = \Sigma - \{\sigma_0\}$. A member of the set Σ' is denoted σ_j, j a positive integer indexing Σ'. Then σ_0 is considered to be an *elementary symbol*, or *semiote*, if three conditions are fulfilled.

well-formed There is one and only one well-formed mapping ω such that $\omega\colon \sigma_0 \longmapsto \omega(\sigma_0)$.

unique The symbol σ_0 and its semantics, $\omega(\sigma_0)$, are unique, or

$$\sigma_0 \neq \sigma_j, \tag{20}$$

$$\omega(\sigma_0) \neq \omega(\sigma_j), \tag{21}$$

$\forall\, \sigma_j \in \Sigma'$.

elementary Denote the set of all possible constructs of symbols in Σ' by $\mathfrak{C}'$, and a particular construct, $\mathfrak{c}_k$, k a positive integer index. Then σ_0 is elementary if, for a well-formed mapping ω, the semantics of every construct, $\omega(\mathfrak{c}_k)$, is not equal to the semantics of σ_0, $\omega(\sigma_0)$, or

$$\omega(\sigma_0) \neq \omega(\mathfrak{c}_k), \tag{22}$$

$\forall\, \mathfrak{c}_k \in \mathfrak{C}'$.

The semiote σ_0 is denoted ς. Every semiote has four properties:

- its formally defined, computable semantics;
- its formally defined, computable syntax;
- its informally defined, natural language semantics; and
- its informally defined, natural language syntax.

The set of all semiotes in *Glossa* is denoted Σ_ς, and is also called the *semantic basis set* of *Glossa*. A construct consisting of semiotes is called a *bundle of semiotes* or a *semiotic bundle*.

We now have the most fundamental symbols in our formal language. Like the rest of the symbols in Σ, the semantics of semiotes, and the semantics of constructs formed from them, are defined. The semiotes include members of the classes and classless symbols. The non-semiotic symbols (the members of Σ'_ς) can vary among computational entities as long as they are semantically well-formed and explicitly defined semiotically (though it is certainly possible to share these symbols and their definitions as well). *Glossa*'s syntax and semantics are independent of any particular computer language: all that is needed is to syntactically transform constructs in the formal language into the computational language(s) of one's choice.

Glossa and *The Agora*

To illustrate some uses of *Glossa*, I now describe some examples drawn from our work on *The Agora* (Bugrim *et al.*, 2000). In preparation for computations shared by the three participating databases, we are developing semiotes and bundles for the transmission of legacy data to a central query server and for the *de novo* deposit of information on biochemical reactions for subsequent automated and human review. Much of these data concern aspects of reaction equations. Legacy data tend to focus on overall biochemical reaction equations as catalyzed by an enzyme or class of very similar enzymes, with varying charge/mass balance and many synonymous names for molecules. Directly deposited information supports a much richer model of biochemistry for all types of reactions, including information on reaction and enzyme mechanisms, kinetics, thermodynamics, biological localization, and phylogenetic distribution. But the reaction equation is fundamental for both.

To begin, consider the semiote denoting a name of a molecular species, cpd_name/1. Figure 1 shows the computable, formal definitions of its semantics and syntax and their narrative translations.

Its defined semantics has elements that clearly fall into the category of something that must be checked by a human—the only clues indicating this is a compound's name are the names of the semiote and the variable. (The computational definitions of semiote syntax and semantics illustrate the huge gap between syntactic and semantic parsers.) This illustrates both the axiomatic nature of the semiotes—their meaning should be obvious—and the importance of human supervision. Similarly, the congruence of formal and informal definitions can only be detected by humans. The syntactic specification gives a regular expression against which to test the semiote's syntax and the language for executing that test, in this case specifying a regular expression using JavaScript syntax. If the definition was used to check that a particular

cpd_name/1

```
cpd_name(_NameOfAMoleculeOrMolecularComplex).
```

Any term used to reference a molecule, molecular complex, ion, or molecular assembly, for all molecules or pseudo-molecules (such as genes) of interest, independent of their structure. The name itself may be a trivial, biochemical, or IUPAC name.

```
semiote_syntax(cpd_name(CpdName)) :-
        nonvar(CpdName),
        apply_regexp(cpd_name(CpdName)).


regexp(cpd_name,javascript,
       '\\''*[\\w\\d\\-\\_\\,\\{\\}\\â\\s\\.\\'']+\\''*::g').
```

A Prolog atom. Either it is enclosed in single quotes ('term') or it obeys the following rules: the first character is a lower-case letter; only alphanumeric characters and underscores are present.

Fig. 1. A semiote and its definitions. The upper left hand phrase is the name of the semiote; to its right, any pertinent mathematical symbol would appear; in the top of the top box, its formal declarative semantics; below, its informal natural language semantics; in the top of the lower box, its formal declarative syntax; and below, its informal natural language syntax. The pseudocode follows the standard Prolog syntax: variables are capitalized, predicates are not; conjunction is indicated by commas, disjunction by a semi-colon; :− is 'is defined as'; and the end of a definition is marked by a period. I have written out the name of the anonymous variable (beginning with _) for declarative clarity; also the Prolog convention for naming predicates and semiotes (name/arity) is used. The predicate *apply_regexp/1*, omitted here for brevity, calls a regular expression definition of the semiote's syntax stored in *regexp/3*, which is shown. The regular expression's single right quotes are escaped for Prolog by doubling them and for JavaScript by a back slash.

datum met the definition of cpd_name/1 from within a Prolog process, a call to the shell to spawn a JavaScript process and execute the regular expression would be issued. In practice, this type of syntactic check is done by a JavaScript function called by an HTML form used to enter data *de novo*, so that only syntactically well-formed, semantically labelled data are transmitted to the server (code omitted for brevity).

The resemblance of the pseudocode to Prolog is not coincidental (Sterling and Shapiro, 1986; O'Keefe, 1990): each formal definition is in fact executable Prolog. One can of course write the definitions in an even more abstract manner, but this seems to confer no real advantage since it would still have to be translated into the computational language of one's choice.

There are two equally valid ways to read the definitions: as a declarative definition in a formal language, and as computational operations in a particular language. The definitional reading of the JavaScript clause is thus 'the value of the variable matches this JavaScript regular expression'; the computational reading is 'call JavaScript and instruct it to perform a pattern match with this pattern on the value of this variable.' Systems written in other languages would implement their parsers between the semiotes and the local system in their own choice of languages, following the definitional reading of the

```
sinistra_cpd(CompoundName) :-
      ( cpd_name(CompoundName)
      ;
        synonym(_PrimaryCompoundName,CompoundName)
      ).

dextra_cpd(CompoundName) :-
      ( cpd_name(CompoundName)
      ;
        synonym(_PrimaryCompoundName,CompoundName)
      ).
```

Fig. 2. Two terms which are not semiotes: their semantics are neither well-formed, unique, nor elementary.

```
data_source_abbrv(umbbd,'UMBBD').
accession(_EntryAccessionNameOrCodeFromContributingDatabase).
cpd_name(_NameOfAMoleculeOrMolecularComplex).
stoich(_StoichiometryOfSpeciesInReaction).
mol_state(_StateOfAMolecularSpeciesInAParticularReaction).
compartment(_CompartmentOfAMolecularSpeciesInAParticularReaction).
```

Fig. 3. Semiotes and their formal semantic definitions used in the transmission of data about reactions. The pertinent instance of data_source_abbrv/2 is shown. I have omitted the names of particular mol_states and compartments for brevity; these are included in the syntactic definitions of these semiotes.

semiotes and their syntactic conventions. We define the syntactic equivalents for the semiotes in Perl, JavaScript, and HTML as needed. Our forms use JavaScript functions to syntactically check input data, prepare the semiotes and bundles, and transmit the message to the server to circumvent the limited functionality of HTML.

Why is cpd_name/1 a semiote? The answer depends on inspection of the full sets of semiotes and bundles (Kazic, 2000), which space prevents showing here, to see if the semiote fulfills the conditions of Definition 4. However, it is useful to state how the semiote meets the definition. First, the semiote's semantics are well-formed: there is only one definition for this semiote. Second, the semiote and its semantics are unique with all other semiotes: there is only one semiote of this name and definition in the set of semiotes, and there are no semiotes of different names having the same definition. Finally, the semiote is elementary: there are no constructs in the set of bundles whose semantics is equal to the semiote.

But it is probably more telling to consider some alternatives to see if they fulfill the definition—say some that denote the name of a compound appearing on a particular side of an arbitrary reaction equation. Consider two such candidates, shown in Figure 2. Why are these not semiotes? The obvious answer is that the result of applying the definitions would be identical whether a particular compound was a sinistra_cpd or a dextra_cpd, so that the condition of uniqueness fails. But the more subtle answers are equally important. First, nowhere do the definitions specify on which side of a reaction equation the compound is found. Instead, that information is dependent on the term's context, given either by a human being or from some other information specified elsewhere, say on an HTML form. If that contextual information were incorporated in the definitions, then obviously they would become unique: but to do so requires knowing the definition of the rxn_eqn/1, which is shown below to involve bundles involving other bundles and terms related to cpd_name/1. So one ends up with, if not a circular, then a very spiral set of definitions. Second, the proposed terms combine two ideas: that of the name of a compound and that of the side of a reaction equation on which it appears. In fact the notion of a reactant is quite complex (e.g. it includes stoichiometry), and the alternatives arbitrarily truncate that set of ideas. Thus the proposed terms fail both the formal definition and several aspects of an intuitive notion of axiomatic.

Given the semiote cpd_name/1, how is it used? In Figures 3 and 4 I list the semiotes, bundles, and their formal semantic definitions used in transmitting UM-BBD legacy data on reaction equations (Ellis and Wackett, 1995a,b).

Reaction equation information comes from UM-BBD in a bundle of bundles, shown in Figure 5. The bundle and its components are computed from a set of mappings between UM-BBDs CORE database and the semiotes; the bundling code is written in Java and is run at UM-BBD, and the results sent as files of bundles. Reading declaratively, a reaction equation must have two sets of coreacting species (sinistras and dextras), but is not required to have a catalyst. This permits both spontaneous reactions and alternative approaches to bundling reaction equation information. A recursive definition for rxn_eqn is used to conveniently exploit the computational model of Prolog; if I were writing the bundle definition in Perl, I would instead iterate over a hash whose keys were the bundle names (sinistras, dextras, catalysts) and whose values were the complete bundles. Figure 5 shows a rxn_eqn bundle instantiated for a particular UM-BBD reaction. Given this bundle, *The Agora* can immediately test to see if it fulfills the conditions specified for a rxn_eqn/1 bundle. If the bundle does not fulfill the semantics of rxn_eqn/1 (or any other available bundle), then the test fails and the bundle is rejected by *The Agora*.

These UM-BBD bundles relied on their names to convey information, for example about a molecule's role in the reaction (reactant, catalyst). In the transfer of legacy data this was sufficiently clear. However in other applications, such as the deposit of information to *The Agora*, it is preferable to label each value in a bundle with its semiote's name and to indicate the rxn_role explicitly to ensure the sets of coreacting species are correctly formed (in

```
data_source(Source) :-
        ( data_source_abbrv(Source,_)
        ;
          data_source_abbrv(_,Source)
        ).

rxn(_AgoraAcc,_TypeOrEC,_SubRxns).

reactant(CompoundName,Stoich,State,Compartment) :-
        cpd_name(CompoundName),
        stoich(Stoich),
        mol_state(State),
        compartment(Compartment).

catalyst(CatalystName,State,Compartment) :-
        cpd_name(CatalystName),
        mol_state(State),
        compartment(Compartment).

sinistras(ListOfSinistraLateralReactants) :-
        forall(member(X,ListOfSinistraLateralReactants),
                X = reactant(_CompoundName,_Stoich,_State,_Compartment)).

dextras(ListOfDextraLateralReactants) :-
        forall(member(X,ListOfDextraLateralReactants),
                X = reactant(_CompoundName,_Stoich,_State,_Compartment)).

catalysts(ListOfCatalysts) :-
        forall(member(X,ListOfCatalysts),
                X = catalyst(_CompoundName,_State,_Compartment)).

rxn_eqn([]).
rxn_eqn([H|T]) :-
        compound(H),
        functor(F,H,1),
        arg(1,F,ListParticipants),
        ( F == sinistras →
                sinistras(ListParticipants)
        ;
                ( F == dextras →
                        dextras(ListParticipants)
                ;
                        F == catalysts,
                        catalysts(ListParticipants)
                )
        ),
        is_unique_set(ListParticipants,T),
        rxn_eqn(T).
```

Fig. 4. Definitions of bundles used in the transmission of legacy data about reactions. The predicate *forall/2* has the same meaning as ∀ and is in this instance most conveniently read, '*X* is a reactant(_,_,_,_) for all *X* which are members of the list.' The definition of *is_unique_set/2* is omitted for brevity. *compound/1* is a Prolog predicate that checks a term's atomicity and has nothing to do with chemistry.

```
[dcb(data_source(umbbd),accession(r0234)),
        sinistras([reactant(ethylbenzene,1,_,_),reactant('H2O',1,_,_)]),
        dextras([reactant('1-phenylethanol',1,_,_),reactant('H+',2,_,_)]),
        catalysts([catalyst('ethylbenzene dehydrogenase',_,_)])]
```

Fig. 5. A bundle transmitted from UM-BBD for its reaction r0234. The bundle is a list which in this case includes the bundles dcb/2 (for datum_constant_bundle), sinistras/1, dextras/1, and catalysts/1. Each of the last three bundles contains a list; the catalysts/1 bundle is not used if the reaction is spontaneous.

```
[sinistras([reactant(cpd_name('ethylbenzene'),stoich(1),mol_state(uns),
                    compartment(prok_unk),rxn_role(noncatalyst)),
            reactant(cpd_name('H2O'),stoich(1),mol_state(uns),
                    compartment(prok_unk),rxn_role(noncatalyst))]),
 dextras([reactant(cpd_name('1-phenylethanol'),stoich(1),mol_state(uns),
                    compartment(prok_unk),rxn_role(noncatalyst)),
          reactant(cpd_name('H+'),stoich(2),mol_state(uns),
                    compartment(prok_unk),rxn_role(noncatalyst))])]
```

Fig. 6. The same reaction as currently bundled by *The Agora*'s reaction equation entry form. Catalyst information is separately bundled. Semiotes used for tracking and relating information are not shown.

```
catalyst('ethylbenzene dehydrogenase',80092,[unsaid],[unsaid]).
datum_index(r0234,80092).
datum_serial_num(80092,reaction(80092,putative_ec_num,[])).
datum_serial_num(80093,putative_ec_num(80092,ec_num(1,17,_,_))).
datum_serial_num(80094,sinistra(ethylbenzene,80092,1,[unsaid],[unsaid])).
datum_serial_num(80095,sinistra('H2O',80092,1,[unsaid],[unsaid])).
datum_serial_num(80096,dextra('1-phenylethanol',80092,1,[unsaid],[unsaid])).
datum_serial_num(80097,dextra('H+',80092,2,[unsaid],[unsaid])).
datum_serial_num(80098,catalyst('ethylbenzene dehydrogenase',80092,[unsaid],[unsaid])).
datum_serial_num(80099,evidence(80092,citation(ball96),unknown)).
datum_serial_num(80102,datum_index(r0234,80092)).
datum_serial_num(80103,xref(80092,umbbd,r0234)).
datum_serial_num(80104,info_entry(80092,[curator_role(person([
        given_names('Ryan'),surname('McLeish')]),typist)],931243011,'hydra.labmed.umn.edu')).
datum_serial_num(80105,search(80092,medline,'http://www.ncbi.nlm.nih.gov/',
        'ethylbenzene[ALL]+AND+anaerobic[ALL]+AND+biodegradation[ALL]')).
datum_serial_num(86548,reaction(86548,'ethylbenzene (anaerobic) pathway',
        [77403,77417,80092,80214,80230,missing,missing])).
datum_source(80092,umbbd).
dextra('1-phenylethanol',80092,1,[unsaid],[unsaid]).
dextra('H+',80092,2,[unsaid],[unsaid]).
evidence(80092,citation(ball96),unknown).
info_entry(80092,[curator_role(person(
        [given_names('Ryan'),surname('McLeish')]),typist)],
        931243011,'hydra.labmed.umn.edu').
putative_ec_num(80092,ec_num(1,17,_,_)).
reaction(80092,putative_ec_num,[]).
reaction(86548,'ethylbenzene (anaerobic) pathway',
        [77403,77417,80092,80214,80230,missing,missing]).
search(80092,medline,'http://www.ncbi.nlm.nih.gov/',
        'ethylbenzene[ALL]+AND+anaerobic[ALL]+AND+biodegradation[ALL]').
sinistra('H2O',80092,1,[unsaid],[unsaid]).
sinistra(ethylbenzene,80092,1,[unsaid],[unsaid]).
xref(80092,umbbd,r0234).
```

Fig. 7. UM-BBD data as represented in *The Agora*. *The Agora* accession numbers are arbitrary for now.

our case by the JavaScript syntax checking/bundling code). Figure 6 shows the same reaction as currently bundled by *The Agora*'s reaction deposit form. Since data on catalysts are separately bundled, the catalyst for the reaction equation is not shown here. Each form element is named by the semiote of the information to be entered in it. Once entry for a given form is complete and the user has pushed the 'submit' button, JavaScript functions check the syntax of the entered data, wrap each datum in its corresponding semiote, and bundle the semiotes together. Data rejected on syntactic grounds are shown to the user, and the user must approve the final data before their transmission to the server. *The Agora*'s model of reaction equations is shown in Figure 7. It can be seen immediately that *The Agora*'s model is quite different from UM-BBDs, and condenses information from many semiotes and several bundles with internally generated information, such as *The Agora* accession numbers and time stamps.

Lightly federating databases and servers

How can *Glossa* and semiotes be used to share information and computations among disparate, independent systems? Three components are needed:

(1) a public set of semiotes and their definitions forming a shared abstract semantic layer, publicly defined and maintained;

(2) publicly visible mappings between the notions reified by a software system and the semiotic layer, for just those computations and data the software system wishes to share, manually constructed and maintained by the local participating system;

(3) a local parser translating between semiotic requests and the local query or implementation language, manually constructed and maintained by the local participating system.

In practice, a user would access a request interface at a participating site and compose a request using the

tools the site provided. Behind the scenes, however, that site would maintain a mapping between its local ideas and the semiotes, and a parser between its own interface language and the semiotes. It would translate the request into semiotes and send it to one or more answering sites (including itself), depending on the requirements of the request and the routing mechanism chosen. Answering sites would maintain mappings and parsers. (Alternatively a central router could distribute requests and answers, but it still would require similar public mapping information.) Upon receipt of the semiotic request local resources would translate it into their local language using their parser; fulfill it; translate the result into semiotes; and return that semiotic result to the requesting site. That site would translate the result as it sees fit and return it to the user. The process is quite similar to http protocols, except that in this case every request and result has a defined semantics. It is quite likely that a request posed by one site would require data or computations from several others: these could now be mixed and matched automatically using their declared semantics with complete confidence in the integrity of the result.

This scheme, which we are implementing in *The Agora*, offers several advantages. First, semiotes can be defined as we need them with only minimal oversight. Thus the task of trying to formalize all biological knowledge at once is very sharply reduced; only as new areas of biology, or new ideas in existing areas are discovered or needed, need the set of semiotes increase. We also have a test for the completeness of any set of semiotes, relative to the nonsemiotic terms, in any reification of *Glossa*: the semantics of every nonsemiotic term should be generable from those of the members of Σ_ς. Because the semiotes themselves only refer to very simple ideas, those most likely to engender controversy are left where they belong—as the private opinions of people, databases, or algorithms. Second, every contributing system is perfectly free to change its internal details and profferred services (database management system, schema, query language, parameter values, computational engine, etc.) without forcing any change in the semiotic layer (assuming that such changes don't suggest new semiotes). Third, the use of a shared layer sharply reduces the number of parsers which must be written. In the worst case of each of n systems writing a unidirectional parser to all other systems, $(n-1)^{n-1}$ parsers would need to be built; with a shared layer, this drops to n bidirectional parsers. Fourth, the semiotes themselves depend on very little technology. Written in ASCII pseudocode and transmitted by http protocols, they can be translated into the local system's language of choice by its resident parser. Once the local parser and mappings have been written, they change only if the local system changes. Finally, the scheme allows for technological evolution. Because it is relatively technology-independent, changes in technologies can evolve independently and in turn use the semiotic layer as they see fit, and *vice versa*.

Discussion

Glossa and its components are tools for declaring the semantics of ideas, data, and computations so that these can be shared among different computational entities. It is best thought of as a *lingua franca* for requesting computations. *Glossa* is neither a multidatabase query language—which would require more agreement among the participating entities than the model advocated here—nor is it a shared schema. Semiotes differ from ontologies in that they computably define and label much smaller ideas and ignore classifications. Semantics are distinct from database schema, which describe the relationships among objects internal to the database but depend on the observer to recognize their meanings (Naqvi and Tsur, 1989; Date, 1990). Thus the semiotes are not shared schema: they are shared fragments from which unshared schema can be built, and indeed are indifferent to schema. Indeed, the intent is to completely bypass such considerations, which necessarily dominate database integrations, in favor of a much more lightly federated approach.

This paper concentrates on machinery for defining the semantics of a biological universe of discourse, and completely ignores issues such as query distribution and efficiency of query execution. The last may become important if network speeds improve significantly.

It remains to be seen if *Glossa* will need to be a context-dependent language. For the moment I am striving to keep it context-free so that a variety of automatic tools that generate look-ahead(1) left-right parsers from a Backus–Naur Form grammar can be employed.

Semantics of database information is sometimes called *metadata*, and there is considerable interest in declaring metadata information so that databases from different sources and even disciplines can be jointly used (e.g. Dombrowski *et al.*, 1994). As usual, the difficulty is deciding if X in one database is X, Y, or some relative of Z in another. Comparing the topologies of the ontologies is obviously flawed, but even matching by identical symbols is insufficient to determine the symbols' usage in this case. Only if the semantics of the symbols and each of their instantiations are provably identical can one then say the two databases are identical in this regard. For data which emanate from one or very few sources, are semantically simple, and circulate among a tightly-knit, very similarly educated community, reliance on an implicit, human-based semantics may be adequate. Thus there are several fairly successful efforts using metadata among the remote

sensing and astronomical communities (though problems are encountered when one tries to use another's data, e.g. radio versus optical astronomers, because the implicit semantics are no longer sufficent; Kurt Weller and Harlan Onsrud, personal communications). But none of these conditions are true in biology, and there are ample examples of inconsistent usage of symbols within a single database. Thus on logical, pragmatic, and social grounds, the need for an explicit, computable semantics is clear.

The strategy detailed here shares with those described in the introduction the first problem—that the ultimate arbiter of correctness is a well-educated and discerning human. *Glossa* seeks to minimize this problem by populating the basis set with terms which express very simple, relatively noncontroversial ideas, by clearly defining the semantics of the semiotes, by defining the semantics of constructs of the semiotes as computations, and by explicitly identifying semiotes, constructs computed with them, and the mappings between the semiotes and a particular computational resource. Provided the definitions of the semiotes are accurate and that they are accurately used—something that for now only humans can judge—then the semantics of their constructs, and the results computed by the constructs, are *verifiable automatically and without reference to the models of the universe of discourse of the databases and algorithms which have contributed to the results*. Hence the emphasis on a computable semantics and the effort to develop a formal structure to use in proving the correctness of constructs.

The semantics of the basis set is constrained to be very elementary and disjoint both to make it easier for humans to agree (an operative approximation of the notion of *axiomatic*) and to permit expression of a very wide range of ideas. These include ideas about which either no consensus exists, epistemologically or in the inner workings of databases and algorithms, or are structurally complex notions. In this framework there is no need to argue over the definitions of reaction or standards for sharing reaction information or how to calculate a molecularity: one simply identifies all the components and fundamental relationships that are recognized by some scientific entity, human or computational, in describing reactions; defines appropriate semiotes for them; and lets each resource, if it wishes to share information on reactions, express its definition of a reaction as an executable bundle of the appropriate semiotes. The definition of semiotes does not depend on the resolution of underlying scientific questions, and in fact may stimulate such discussions by explicitly defining the alternatives. As a rule of thumb, the existence of scientific or technical differences is a signal that more than one semiote is needed for the basis set.

The semiote definitions are written in declarative pseudocode which happens to be executable Prolog. In writing the definitions I have tended to opt for declarative clarity rather than procedural speed (the choice of *memberchk/2*, which does not backtrack if it succeeds, over *member/2* illustrates an exception). I also opted to use a computational language as our basic definitional tool, rather than a purely formal system, to emphasize the computability of the definitions. The reader should note the semantics of the pseudocode folds in some of the semantics of Prolog, for example in *forall/2*, *compound/1*, and *memberchk/2* (Swedish Institute of Computer Science, 1999). While the definitions do not rely on a particular computational model for their semantics, they do exploit notions of recursion, iteration, and term rewriting and equivalence testing common in computational languages.

Each computational resource is free to choose and change its own machinery to implement its use of semiotes. Moreover, any future technological changes which prove advantageous can readily incorporate semiotes, without altering their definitions. In *The Agora*, we are multilingual and multisystem in our technical choices: implementation of the definitions and everything else involves many languages, software systems, and platforms, and constantly evolves.

We are testing the utility of this approach by first demonstrating these ideas, implementing them as part of *The Agora* (Bugrim *et al.*, 2000). A fuller description of the implementation is the subject of another paper; the examples shown are selected from many semiotes, bundles, operators, and mappings (Kazic, 2000). Our experience so far indicates one can indeed define semiotes incrementally and communally, and that vigilant checking against the definition of semiotes is important. It is premature to speculate on how well this approach will prove to scale. I do not yet have a good estimate for the eventual size of the semantic basis set for *The Agora*: it is certainly more than a few terms but so far does not seem as if it will be intractably large. As each area of biology is added the basis set will continue to grow. The hope is that by choosing axiomatic notions the number of semiotes will be minimized. While incremental definition reduces the effort it obviously doesn't eliminate it altogether. We have experimented to find the right 'granularity' of ideas for definition of semiotes, but some future fluidity is possible. Preliminary definitions of semiotes can be found at (Kazic, 2000).

Implementing these ideas for many computational resources will certainly require community participation and effort. Some sort of oversight body to resolve conflicts, check definitions, and monitor simplicity and uniqueness will be needed; and for those who wish to participate, some effort to write and maintain the mappings and parsers. Apart from official scientific nomenclature bodies, efforts at communal definition have often been still-born in the past, though they have tended

to focus more on big ideas and so are not necessarily directly applicable to the present proposal. Yet we have several remarkable successes to cheer us on: the growth of standards for http and HTML, and the rapid and transparent exchange of financial information worldwide come immediately to mind. Moreover, the need for ready, semantically reliable exchange continues to grow as each of us wants the information in somebody else's database or needs that other person's algorithm for our own task. It's more efficient to share, and *Glossa* and semiotes offer one way of doing so.

Acknowledgements

I thank Russ Altman, Helen Berman, Sinéad Boyce, Andrej Bugrim, Lynda Ellis, Francis Fabrizio, Mark Feldman, Douglas Hershberger, Peter Karp, Maureen Kelly, Harlan Onsrud, Jun Ouyung, Andrew McDonald, Jakub Slomczynski, Keith Tipton, Kurt Weller, and Guang Yun for useful discussions; Jakub Slomczynski for writing HTML forms and JavaScript functions; and Jun Ouyang and Guang Yao for writing the UM-BBD mappings and parser. Parts of this work were conceived while the author was attending workshops on gene recognition at the Aspen Center for Physics. This work is supported by the National Institutes of Health (GM-56529-01).

References

Abernethy,N. (1999–present). The Molecular Biology Ontology Working Group. http://smi-web.stanford.edu/projects/bio-ontology/: Stanford University.

Altman,R.B. and Abernethy,N.F. (1997) Standard representation of the literature: combining diverse sources of ribosomal data. In Gaasterland,T., Karp,P., Karplus,K., Ouzonis,C., Sander,C. and Valencia,A. (eds), *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology* vol. 5, American Association for Artificial Intelligence, Menlo Park, CA, pp. 15–24.

Altman,R.B., Dunker,A.K., Hunter,L. and Klein,T., (eds) (2000) *Pacific Symposium on Biocomputing*. vol. 5, World Scientific, Singapore.

Aronoff,M. and Rees-Miller,J., (eds) (2000) *The Handbook of Linguistics*. Blackwell, Oxford.

Baclawski,K., Cigna,J., Kokar,M., Mager,P. and Indurkhya,B. (2000) Knowledge representation and indexing using the Unified Medical Language System. In Altman *et al.*, 2000, (ed.), pp. 490–501.

Baclawski,K., Futrelle,R., Hafner,C., Pescitelli,M.J., Fridman,N., Li,B. and Zou,C. (1993) Data/knowledge bases for biological papers and techniques. In Chu,W.W., Cardena,A.F. and Taira,R.K. (eds), *Proceedings of the NSF Scientific Database Projects* National Science Foundation, Washington, DC, pp. 23–28.

Bellahsene,Z. (1997) Extending a view mechanism to support schema evolution in federated database systems. In Hameurlain and Tjoa, 1997, (ed.), pp. 573–582.

Blaschke,C., Andrade,M.A., Ouzounis,C. and Valencia,A. (1999) Automatic extraction of biological information from scientific text: protein-protein interactions. In Lengauer *et al.*, 1999, (ed.), pp. 60–67.

Bourne,P., Berman,H.M., Watenpaugh,K., Westbrook,J. and Fitzgerald,P.M.D. (1997) The macromolecular Crystallographic Information File (mmCIF). *Meth. Enzymol.*, **277**, 571–590.

Bugrim,A., Boyce,S., Yao,G., Slomczynski,J., McDonald,A., Feng,B., Wise,W.B., Ellis,L., Tipton,K. and Kazic,T. (2000–present). The Agora. http://www.ibc.wustl.edu/biognosis/agora_interface/html/agora_entrance.html/: Institute for Biomedical Computing, Washington Univesity.

Chaudhri,V.K. and Lowrance,J.D. (1999–present). Generic Knowledge Base Editor. http://www.ai.sri.com/~gkb: SRI.

Chen,I.-M.A., Kosky,A.S., Markowitz,V.M. and Szeto,E. (1997) Constructing and maintaining scientific database views. *Proceedings of the 9th Conference on Scientific and Statistical Database Management* IEEE Computer Society Press, Rockville MD, pp. 237–248.

Clark,P.E. (1999–present). Some Ongoing KBS/Ontology Projects and Groups. http://www.cs.utexas.edu/users/mfkb/related.html: University of Texas at Austin.

Craven,M. and Kumlien,J. (1999) Constructing biological knowledge bases by extracting information from text sources. In Lengauer *et al.*, 1999, (ed.), pp. 77–86.

Date,C.J. (1990) *An Introduction to Database Systems*. vol. I. 5th edn, Addison-Wesley, Reading MA.

Dombrowski,E.G., Snyder,W.A. and Heckathorn,H.M. (1994) Metadata management and the VISTA system. In Nunamaker, Jr, J.F. and Sprague, Jr, R.H. (eds), *Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences* vol. 3, IEEE Computer Society Press, Los Alamitos CA, pp. 418–427.

Eco,U. (1984) *Semiotics and the Philosophy of Language*. Indiana University Press, Bloomington IN.

Ellis,L.B.M. and Wackett,L.P. (1995) A microbial biocatalysis database. *Soc. Indus. Microbiol. News*, **45**, 167–173.

Ellis,L.B.M. and Wackett,L.P. (1995–present). University of Minnesota Biocatalysis/Biodegradation Database. http://www.labmed.umn.edu/umbbd/index.html: University of Minnesota.

Elmagarmid,A., Rusinkiewicz,M. and Sheth,A., (eds) (1999) *Management of Heterogeneous and Autonomous Database Systems*. Morgan Kaufmann, San Francisco.

Futrelle,R.P. (1997) Distributed intelligent systems for a knowledge-based scientific literature, Unpublished manuscript, privately circulated.

Genome Data Base Staff, (1996) *Genome Data Base*. http://gdbwww.gdb.org/.

Goble,C., Crowther,P. and Solomon,D. (1994) A medical terminology server. In Karagiannis, 1994, (ed.), pp. 661–670.

Hameurlain,A. and Morvan,F. (1996) Parallel relational database systems: why, how, and beyond. In Wagner and Thoma, 1996, (ed.), pp. 302–312.

Hameurlain,A. and Tjoa,A.M., (eds) (1997) *Database and Expert Systems Applications. 7th International Conference, DEXA '97. Lecture Notes in Computer Science*, 1308, Springer, Berlin.

Hammer,J., García-Molina,H., Nestorov,S., Yerneni,R., Breunig,M. and Vassalos,V. (1997) Template-based wrappers in the TSIMMIS system. In Association for Computing Machinery, (ed.),

SIGMOD '97, Proceedings of the ACM SIGMOD International Conference on Management of Data pp. 532–535.http://info.acm.org/pubs/contents/proceedings/mod/253260/: Association for Computing Machinery.

Hammer,J. and McLeod,D. (1999) Resolution of representational diversity in multidatabase systems. In Elmagarmid *et al.*, 1999, (ed.), pp. 91–118.

Harris,Z., Gottfried,M., Ryckman,T., Mattick, Jr, P., Daladier,A., Harris,T. and Harris,S. (1989) *The Form of Information in Science: Analysis of an Immunology Sublanguage*. Reidel, Dordrecht.

Houstis,C., Paptheodorou,T.S., Verykios,V., Floratos,A. and Elmagarmid,A. (1994) BIND: a Biomedical INteroperabile Database system. In Karagiannis, 1994, (ed.), pp. 671–679.

Huemer,C., Quirchmayr,G. and Tjoa,A.M. (1997) A meta message approach for electronic data interchange (EDI). In Hameurlain and Tjoa, 1997, (ed.), pp. 377–386.

Humphreys,K., Demetriou,G. and Gaizauskas,R. (2000) Two applications of information extraction to biological science journal articles: enzyme interactions and protein structures. In Altman *et al.*, 2000, (ed.), pp. 502–513.

IUCr Working Party on Crystallographic Information (1999–present). Macromolecular Crystallographic Information File. http://ndbserver.rutgers.edu/mmcif/: Rutgers University.

James,R.C. and Beckenbach,E.F. (1968) *James and James Mathematics Dictionary*. van Nostrand-Reinhold, Princeton, NJ.

Karagiannis,D. (ed) (1994) *Database and Expert Systems Applications, 5th International Conference, DEXA '94. Proceedings Lecture Notes in Computer Science*, 856, Springer, Berlin.

Karp,P.D., Chaudhri,V.K. and Thomere,J. (1999–present). XOL: an XML-based Ontology Exchange Language. Approximately http://www-smi.stanford.edu/projects/bio-ontology/xol.doc: Stanford University.

Kashyap,V. and Sheth,A. (1999) Semantic similarities between objects in multiple databases. In Elmagarmid *et al.*, 1999, (ed.), pp. 58–89.

Kazic,T. (1995) Chemical information: how do we get it and what do we do with it? In Collier,H. (ed.), *Proceedings of the 1995 International Chemical Informatics Conference, Nîmes, France* Infonortics, Ltd, Calne, UK, pp. 48–61.

Kazic,T. (2000–present). *Glossa*, Semiotes, and Related Files. http://www.ibc.wustl.edu/biognosis/technologies/semiotes/: Institute for Biomedical Computing, Washington Univesity.

Kazic,T., Yao,G., Bugrim,A. and Slomczynski,J. (2000–present). *Glossa* Semiotes. http://www.ibc.wustl.edu/biognosis/technologies/reprints/semiote_list.ps: Institute for Biomedical Computing, Washington Univesity.

Kent,R.E. (1999–present). Ontology Markup Language, version 0.2. http://wave.eecs.wsu.edu/CKRMI/OML.html: Wayne State University.

Knowledge Systems Laboratory, (1999–present). Current Projects. http://www-ksl.stanford.edu/currentproj.html: Stanford University.

Lakoff,G. (1987) *Women, Fire, and Dangerous Things : What Categories Reveal about the Mind*. University of Chicago Press, Chicago.

Lee,Y.-W. and Yoo,S.I. (1994) Semantic query optimization in OODB systems. In Karagiannis, 1994, (ed.), pp. 651–660.

Lehnert,W. (2000) Information Extraction. http://www-nlp.cs.umass.edu/nlpie.html: University of Massachusetts, Amherst.

Lengauer,T., Schneider,R., Bork,P., Brutlag,D., Glasgow,J., Mewes,H.-W. and Zimmer,R., (eds) (1999) *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology* American Association for Artificial Intelligence, Menlo Park, CA.

Li,C., Yerneni,R., Vassalos,V., Garcia-Molina,H., Papakonstantinou,Y., Ullman,J. and Valiveti,M. (1997) Capability based mediation in TSIMMIS. In Association for Computing Machinery, (ed.), *SIGMOD '98, Proceedings of the ACM SIGMOD International Conference on Management of Data* pp. 564–566 http://info.acm.org/pubs/contents/proceedings/mod/276304/: Association for Computing Machinery.

Message Understanding Conference,, (ed) (1992) *Fourth Message Understanding Conference, (MUC-4): Proceedings of a Conference held in McLean, Virginia, June 16–18, 1992* Morgan Kaufmann, San Mateo CA.

Meyer,B. (1990) *Introduction to the Theory of Programming Langauges*. Prentice-Hall, Hemel Hempstead, Hertfordshire, UK.

Mihaila,G.A., Raschid,L. and Tomasic,A. (1998) Equal time for data on the Internet with WebSemantics. In Schek,H.-J., Saltor,F., Ramos,I. and Alonso,G. (eds), *Advances in Database Technology—EDBT '98, 6th International Conference on Extending Database Technology Lecture Notes in Computer Science,* 1377, Springer, Berlin, pp. 87–101.

Missier,P., Rusinkiewicz,M. and Jin,W. (1999) Schema integration: past, present, and future. In Elmagarmid *et al.*, 1999, (ed.), pp. 119–155.

Murray-Rust,P. (1995) *SGML Experiment Collaborative HyperGlossary*. http://www.cryst.bbk.ac.uk/glossary/sgml/cml-0.5/doc/cml/cml.html, Birkbeck College, London.

Naqvi,S. and Tsur,S. (1989) *$\mathcal{LDL}$: A Logical Language for Data and Knowledge Bases*. Computer Science Press, Rockville MD.

Nelson,S.J., Fuller,L.F., Earlbaum,M.S., Tuttle,M.S., Sherertz,D.D. and Olson,N.E. (1993) The semantic structure of the UMLS metathesarus. In Frisse,M.E. (ed.), *Sixteenth Annual Symposium on Computer Applications in Medical Care: A Conference of the American Medical Informatics Association* McGraw-Hill, New York, pp. 649–653.

O'Keefe,R.A. (1990) *The Craft of Prolog*. MIT Press, Cambridge MA.

Ounis,I. and Chevallet,J.-P. (1996) Using conceptual graphs in a multifaceted logical model for information retrieval. In Wagner and Thoma, 1996, (ed.), pp. 812–823.

Papakonstantinou,Y., Gupta,A., Garcia-Molina,H. and Ullman,J. (1995) A query translation scheme for rapid implementation of wrappers. In Ling,T.W., Medelzon,A.O. and Vieille,L. (eds), *Fourth International Conference on Deductive and Object-Oriented Databases, DOOD '95 Lecture Notes in Computer Science*, 1014, Springer, Berlin.

Proteometrics, Inc., (1999–present). The BIOpolymer Markup Language Home Page. http://204.112.55.140/BIOML/index.html: Proteometrics, Inc.

Ram,S. (1991) Heterogeneous distributed database systems. *Computer*, **24**, 7–9.

Rindflesch,T.C., Tanabe,L., Weinstein,J.N. and Hunter,L. (2000) EDGAR: extraction of drugs, genes and relations from the biomedical literature. In Altman *et al.*, 2000, (ed.), pp. 514–525.

Ritter,O. (1994) The integrated genomic database (IGD). In Suhai,S. (ed.), *Computational Methods in Genome Research* Plenum Press, New York, pp. 57–73.

Salton,G., Allan,J. and Buckley,C. (1993) Approaches to passage retrieval in full text information systems. In Korfhage,R., Rasmussen,E. and Willet,P. (eds), *SIGIR 93: Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* Association for Computing Machinery, New York, pp. 49–58.

Schulze-Kremer,S. (1996–present). Ontology Editor by Steffen Schulze-Kremer. http://igd.rz-berlin.mpg.de/~www/prolog/oe.html: Max-Planck Institute for Molecular Genetics.

Seo,D.-Y., Lee,D.-H, Moon,K.-S., Chang,J., Lee,J.-Y. and Han,C.-Y. (1997) Schemaless representation of semistructured data and schema construction. In Hameurlain and Tjoa, 1997, (ed.), pp. 387–397.

Sheth,A.P. and Larson,J.A. (1990) Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, **22**, 183–236.

Sowa,J.F. (1984) *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading MA.

Sowa,J.F. (1998) *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. PWS Publishing, Boston, MA.

Stapley,B.J. and Benoit,G. (2000) Biobibliometrics: information retrieval and visualization from co-occurrences of gene names in medline abstracts. In Altman *et al.*, 2000, (ed.), pp. 526–537.

Sterling,L. and Shapiro,E. (1986) *The Art of Prolog*. MIT Press, Cambridge MA.

Swedish Institute of Computer Science, (1999–present). SICS Quintus Prolog Manual. http://www.sics.se/isl/quintus/qp/frame.html: Swedish Institute of Computer Science.

The FlyBase Consortium, (1999) The flybase database of the *Drosophila* genome projects and community literature. *Nucleic Acids Res.*, **27**, 85–88.

The FlyBase Consortium, (1999–present). FlyBase. http://fly.ebi.ac.uk:7081/docs/lk/controlled-vocabulary.txt: European Bioinformatics Institute.

Thomas,J., Milward,D., Ouzounis,C., Pulman,S. and Carroll,M. (2000) Automatic extraction of protein interactions from scientific abstracts. In Altman *et al.*, 2000, (ed.), pp. 538–549.

Topaloglou,T., Kosky,A. and Markowitz,V. (1999) Seamless integration of biological applications within a database framework. In Lengauer *et al.*, 1999, (ed.), pp. 272–281.

Wagner,R.R. and Thoma,H., (eds) (1996) *Database and Expert Systems Applications. Seventh International Conference, DEXA '96. Lecture Notes in Computer Science*, 1134, Springer, Berlin.

Wu,X. (1996) An architecture for interoperation of distributed heterogeneous database systems. In Wagner and Thoma, 1996, (ed.), pp. 688–697.