Translation of Various Bioinformatics Source Formats for High-Level Querying

John McCloud and Subhasish Mazumdar

New Mexico Institute of Mining and Technology
Socorro, New Mexico, U.S.A
{amccloud, mazumdar}@cs.nmt.edu

Abstract. Representations of genomic data is currently captured for use in the field of bioinformatics. Each of these representations have their own format, caveats, and even specific "sub-languages" used to represent the information. This makes it necessary to use different specialized programs to make sense of — and then query — the data. Unfortunately, this means a great deal of lost time necessary for learning both the tool and the format itself in hopes of extracting details that can then later be used in answering queries. There is, therefore, a great need for a general purpose tool that can address all of these different formats at once: a tool that will abstract such low-level, format-specific representations of raw data to a higher, operating level with terms that are immediately recognizable to experts within the bioinformatics and biology domains. In addition, it will help biologists engaged in research pose questions that were not anticipated by the authors of the special purpose software built for the data files.

1 Introduction

Ontologies are a way to formalize conceptual representations of a domain: conceptual entities, their constituents, and/or how they relate to other entities. Such formalizations are necessary to bring a coherent, shared view of items and processes within domains, removing ambiguity in both definition and relationship. There are many issues, however, when it comes to automatically classifying such information, especially when one is attempting to merge ontologies of both related and unrelated domains. Such difficulties are compounded when the data within such heterogeneous sources are obfuscated in some manner — unable to simply be decoded by language processing or recognition through keywords. Instead, an expert in the domain is generally needed to make sense of such data.

In this paper, we explore the issue of both merging different low-level data sources and making them accessible to queries at a higher level of abstraction, addressing a current need in the field of bioinformatics and biology. We present a way to build robust and correct ontologies that allow querying across various, low-level data formats. We show that this system is very beneficial to many researchers in the bioinformatics domain, giving them the ability to combine data from all their various data formats and also pose queries that are hard to formulate within the dedicated software systems that come with the data files.

If we only examine DNA sequencing data, there are already many different formats with their own definitions. SAM, FASTA, BEM, FASTQ, PIR, MSF, CLUSTAL, and so on, along with different annotation formats BED, GTF, GFF3, GVF, and VCF. Right now, biologists have a lot of dedicated software that has been built to deal with one or a few data formats as these. The problem is that there is a real need for a general purpose tool that allows biologists to explore all the data that they have in any format, but there is no tool that allows them to look at anything they wish in the data. Instead, they are forced to use whatever definitions and terms those specialized pieces of software define.

We have developed a solution to this issue by manually mapping terms in the ontology for these various, low-level data sources. Alongside an effective, simple target at an operational level for querying, contextual explanation for the ontology-level classifications are presented, giving a more cohesive understanding for adding new information to an ever-growing map of relationships in data.

The approach does not have only one, completely unambiguous ontology from which all knowledge representation stems. Instead, it allows for multiple domain-level ontologies to coexist at once in a shared, operational level (from which queries are made).

This paper is structured as follows. In the next two sections, we present a review of related work and a short primer on biology. Subsequently, in section 4, we delve into design issues. A case study dealing with files using the SAM format is presented in section 5, followed by respective sections of short analysis and concluding remarks.

2 Previous Work

Much of the work that has been done in linking data sources usually starts out as a way to combine low-level ontologies. While such research is extremely valuable in the work presented here, we remind the reader that we are not merging well-defined ontologies, but instead linking the raw or semi-raw data that a researcher in the field may collect — and linking *data formats* — with larger ontologies. Our research attempts to implement ideas from the previous work below.

2.1 C-OWL

Contextual OWL took the original language of OWL and made a system that assumed a single, global ontology for all term interpretation would not provide suitable representation for particular *local* cases. By this extension, the research realized that context within a local ontology must be maintained. In order to represent ideas from knowledge at the local and higher levels, contextual mappings are created with relationships to the various levels of abstractions (from local to higher levels) linked by *bridges* [1].

These bridges are similar to what is represented here in this research, but they are much more abstract, often times explicitly mentioning the levels of abstraction (where a given term within the ontology resides), and the relationship to another term in the bridge. (This allows for such mappings as $Term_A$ is identical to $Term_B$ from low-level ontology A and high-level ontology B.) The bridges and their mappings allow for denoting *contextual levels of specificity* of terms, but do not describe how to actually derive terms from low-level sources.

This is because both source-level and target-level information they deal with are assumed to be OWL ontologies themselves. In our research, we assume all data from the low-level source is *not* within any such formalization to begin with, but must instead be built from RAW data and linked to a higher-level ontology. Much as in OWL, though, the source-level definitions maintain a useful context, and terms within that context must be dealt with differently than terms with other contexts (from other sources).

2.2 KRAFT

Perhaps the most inspiring work for this research comes from the KRAFT project, which recognized the necessity of a shared ontology amongst many low-level ontologies and manual linking from the low-level definition up to the single, shared one [2]. Their agent-model design also includes a distributed solution, whereby one user agent can receive knowledge from another user agent (and his definitions) across a network.

By creating a new language that acts as a link between the shared ontology and the low-level sources, this Constraint Interchange Format (CIF) made it possible for user agents to make queries against data not defined in the shared ontology. This turned the query into a Constraint Satisfaction Problem. A user could then ask a question, and these constraints would be developed for the various low-level sources, using their particular, individual languages. This kept a user from needing to know the format and definitions of anything other than the shared ontology.

The issue with the KRAFT system is that it allows mappings to be arbitrary and possibly completely wrong. This, however, is an acceptable drawback of the system, since we must trust those knowledge experts who deal with their data regularly to define it and expose caveats, fringe cases, and easy-to-miss constraints within the defined resources.

Additionally, it is not clear if the mappings allow for any links larger than one step. That is, multiple transformations from low-level heterogeneous source and the shared ontology. Certain multiple steps may be necessary to produce an effective, overall mapping. Furthermore, combining these intermediate definitions may be an important task that should be available to provide richer explanation from low-level details.

3 Primer on Biology and Sequencing

For readers unfamiliar with the concepts and terms we will use shortly, this section provides a quick primer on the central dogma of biology. This begins with *DNA*, an extremely long sequence of nucleotide bases, which makes up almost every living thing (*genome* is another term used to refer to the total genetic data of an organism, and is more general); and all living organisms are a collection of cells containing a certain number of recognizable *chromosomes*, wound up in tight bundles around proteins; each chromosome contains one or two DNA molecule(s).

DNA is a code that contains all the genetic information necessary for every type of *cell* in our bodies; from cells that make up our eyes to cells that form our skin. Each cell is specially tuned to perform a certain way, depending on which portions of the DNA are manifested in it and how the organism should respond to the environment it lives in.

Portions of the DNA have regions called *genes*, and when these genes are *expressed*, it can result in the creation of tiny, biological actors within the cell, called *proteins* (and if expressing the region does result in proteins, we say that it is a *coding region*), which perform all types of functions. Each cell usually only contains a certain subset of proteins, and this is dependent on the type of cell in which the proteins are built in. When an organism exhibits a certain type of protein, we say it expresses the gene that codes for the protein; this is referred to as *gene-expression.*

Usually, cells produce a reasonably normal amount of proteins in each cell (as per the cell's type). However, when the pressure from the environment results in some shift in the need of the organism, a cell's DNA can adapt to create many of a particular type of protein. This is usually to protect the organism or help it cope with whatever environmental stresses it needs to respond to. Such shifts in gene expression are interesting to examine in biology, because it can assist in discovering new regions of the genome that were not previously known. Often, such discoveries are made by comparing an observed DNA coding region against a *reference genome*; an agreed upon reference for an organism's total genetic material.

For the purposes of this paper, it also helps to understand sequencing and sequence alignment. Sequencing is the process of discovering the actual order of the nucleotide bases that constitute the organism's DNA. Once sequenced, the various sections of the DNA are matched against a reference to create a standard alignment of the data.

As a result of sequencing, one may find a series of *reads* (portions/fragments of genetic data) that are very similar, appear in many locations of the genome, or sometimes just shifted slightly by location. When the locations and lengths are overlaid one on top of one another, the ones with the greatest overlap – these overlapping *pileups* — leads to stronger confidence in accurate representations of that fragment of the genetic code; they are referred to as *consensus sequences*. These consensus sequences are important, because one might conclude that a gene exists there, with gene expression strength depending on how many reads were overlapping along this same area [8].

4 Design

In building the ontology design, these core goals have been identified:

– An expert in the domain should require only the knowledge of her domain and not the additional definitions in the file formats for querying and query resolution. We must separate the knowledge of the low-level, specific data formats from the domain expert's "standard" knowledge of the domain.

– We must create intuitive methods for domain experts to implement additional mappings as new data formats present themselves. A tool that can deal with multiple data formats should be easily extensible.

– Queries at the topmost level must be independent of data formats. Seamless execution on different files (and therefore, different formats), will allow the data to be cross-referenced, shared, and pooled together.

4.1 The Ontology's Structure

In describing the architecture of this ontology design, it helps to think of a staircase. At the top of the staircase are the top-level definitions, while at the bottom are the source-file definitions. The source-file definitions are transformed at each ascending step, becoming a little closer to the shared-level definitions at the top. Definitions at the highest, top step of the staircase are what experts in their domain might use to describe a question.

The definitions at the lowest, source-level step are what each particular file format defines. These are terms that are designed by individuals crafting file formats. Such definitions can be arbitrary, and do not generally share the language of the high-level. Instead, they define terms that are terse, complex, or otherwise lacking in sufficient identification as to be used by a general expert of the domain. For instance, DNA Polymerase, in such a low-level, could be defined as DNA_P1 or DP1, but this is not how biologists refer to it; this is how someone who built the file format has chosen to represent it.

The goal is to allow queries using the concepts of the high-level to pull information from the lower levels, even though their definitions may not immediately align. Their definitions are mapped from the source-level to the shared-level (and vice-versa), transforming a definition in one format, to intermediate definitions, and then finally to the agreed upon definitions of terms used by an entire field or domain.

Take this staircase analogy and imagine that the steps are now layers. There are several layers, and at each layer L a given definition D exists. In order to go from one layer to another, some transformation defined by a function is performed on that definition. Sets of such functions are contained within bridges B.

Users, such as biologists, may ask queries such as: ***Which alignment is repeated the most in the HFD1.sam data, and does it represent a new gene?*** This query aims at finding possibly unknown genes that may have been exposed by environmental stress in the experimental, sequenced organism.

Next, let us break down the query into the high-level terms (and their attributes) that need to exist at the shared-level.

- **Chromosome** (*Number*)
- **Gene** (*Name*, *Start Location*, *End Location*, *Expression*)
- **Sequence** (*Start Location*, *End Location*, *Length*, *Bases*)
- **Alignment** (*Unique*, *Total Matches*, *Mapped*, *Mismatching Positions*)

Since these terms (and attributes) are used in queries, a domain expert must build bridges for them from the source-level up to the shared-level (Fig. 1).

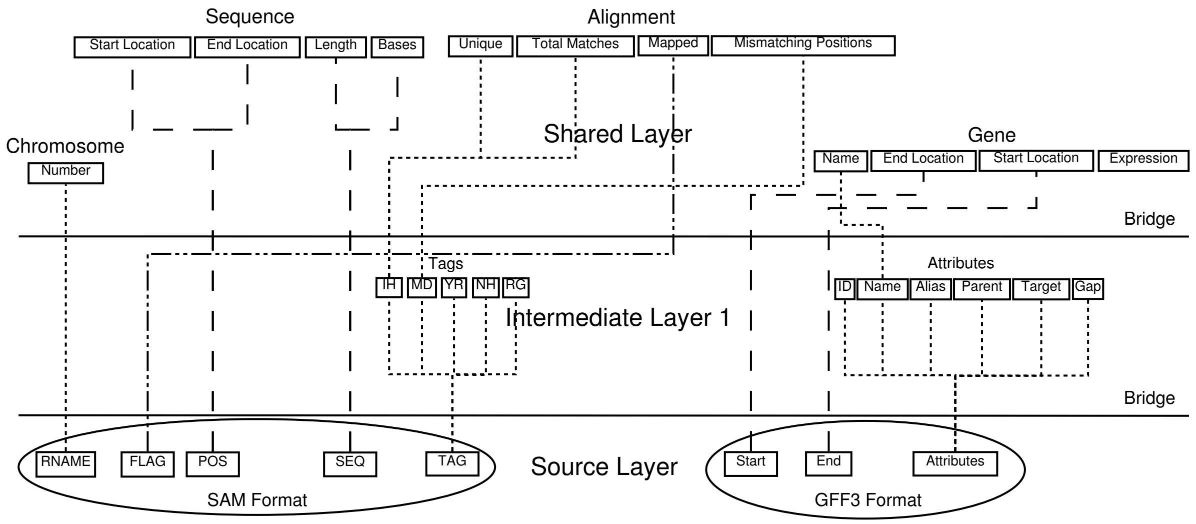

Fig. 1: *Depiction of layers and bridges within the architecture. Each layer contains its own particular set of definitions, while bridges contain the sets of functions for transforming definitions at a given level to adjacent ones. The example here has two source formats: SAM and GFF3.*

Our ontology design requires at least two layers and one bridge: the *shared* L_S and *low-level* ("ground") L_G layers, and the single bridge $B_{L_S L_G}$ containing two sets of functions: one set of functions for the transformation of data defined in the low-level layer *to* the shared layer f_{ij} and another for the inverse functions (mapping shared-layer definitions back down to the low-level) f_{ij}^{-1}. An arbitrary number of additional, intermediate layers and their associated bridges, however, is possible (and necessary in some cases).

Generalized representations for these concepts are given here:

– A function f that lives within a bridge, transforming some definition i at a lower layer into definition j at a higher layer.

$$f_{ij} : D_i \to D_j \ . \tag{1}$$

– The inverse of the transformation in Equation (1), showing the transformation of some definition j into definition i (where j exists at a higher layer and i exists at a lower layer).

$$f_{ji} : D_j \to D_i \ . \tag{2}$$

– Each pair of layers only needs one bridge, and both functions in Equations (1) and (2) are included in the same bridge, but in two different sets (one set for transformations from the higher-to-lower layer with the other set dealing with lower-to-higher layer transformations). These sets are denoted in (3) and (4) (with generic layers A and B).

$$B_{L_A L_B} \equiv B_{L_B L_A} \ . \tag{3}$$

$$B_{L_A L_B} = \{f_{ij}\}, \{f_{ji}\} \ . \tag{4}$$

These definitions, however, do not elucidate the full definition of bridges and the functions within; *bridges can be non-surjective and non-injective.* From the source-level, a mapping to the domain level *may* exist, and *if* it exists, the inverse function(s) defined by the bridge can be performed to return to the source-level representation.

That is, given f_{nm}, a function to transform definition n to m it's inverse would be f_{nm}^{-1} if it exists.

Such transformative functions can also be combinations of other, already defined functions. This means that a transformation from one layer to another can be dependent on terms defined at the same level as well as those below it. This allows one to create intermediate definitions and functions at various levels that can be called on at any time to be used in adding further transformation.

4.2 Foundational Definitions

Upper-level ontologies define abstract definitions (which we refer to as *primitives*) for the elements within domain-specific ontologies, establish relationships amongst these abstract definitions, and so on. Perhaps it is easiest to think of the upper ontology — indeed, the entire ontology — from an object-oriented point-of-view, where the top level is as generic as possible, and with the definitions becoming more concrete as one descends the hierarchy.

These primitives have been identified in our design:

– **Component**: The pieces an entity is composed of; what makes up an entity. Components "belong to" an entity.

- **Constraint**: Conditions that allow/do not allow a certain function or process to occur.
- **Entity**: Any physical building block or standalone actor that performs actions or functions. Entities "have one-to-many" functions. Entities "have one-to-many" components.
- **Function**: Actions which are performed by entities. Acts "with an" entity. May be "part of" a process. Can "depend on one-to-many" constraints.
- **Process**: Possibly non-linear sequence of interacting functions. Processes "has one-to-many" functions. Processes may "depend on one-to-many" constraints.
- **Property**: Descriptive parts of a primitive type (any of component, entity, etc). It is possible for everything to "have one-to-many" properties.
- **Rule**: Axioms for determination of validity in interaction or outcome thereof. It is possible for everything to "have one-to-many" rules that govern them.

Using such terms, we can take concepts from a given domain, set its interactions, rules, and so on in order to relate definitions within the ontology to one another.

Upper ontologies contain basic, atomic definitions for concepts in the world. The shared ontology (here, the collections of domain-specific ontologies) contains various implementations of these atomic, abstract concepts. For example, an abstract definition may be an entity. A particular implementation in the financial domain may be a bond, stock, or future. This concrete definition may differ depending on the domain-specific ontology in which it is used, since a "future" in physics is not the same concept as in finance.

If "future" in both physics and finance is defined as an *entity*, it means that both definitions have the same primitive type that possesses rules and context for interaction with one another. The terms have both been given meaning from that base definition, but can also extend their abilities by defining something more specific to their domain-related purpose.

Such relationships similar to "has a", "is a", and others are defined for understanding connections amongst definitions at this level and other levels. For example, consider that an entity is an object that is distinctly different from a function; that functions are performed by entities. Therefore, an entity would "have a" function.

For example, *RNA Polymerase* is a term that biologists know; it is an entity, with rules and functions associated with it. A process might be DNA Replication, with a series of functions in a particular order, and rules for how they must interact. All would be defined at the shared-layer, using the actual biological terms.

To clarify, take the example of RNA Polymerase, while again considering the structure in Fig. 1. At the shared-level, we develop a definition for this entity and define its various relationships as given in Fig. 2. Even from such a small example, one is able to see how these various primitive definitions for the model are linked together.

4.3 The Shared-Level Ontologies

The shared-level ontologies are a series of domain-specific views. At this level, multiple domain ontologies coexist as a union of all the domains that are relevant (and would be loaded into the system). Within each specific domain, basic terms are defined, and the shared-level deals with overlaps, such as identical terms and possible definitions, that may arise as a result of this union.

In our design, the shared-level ontology is also the operational level, meaning that queries are formed from here, potentially involving every domain in the union. This is to allow for the same terms across domains to be used in a single query. However, this also means that a system of

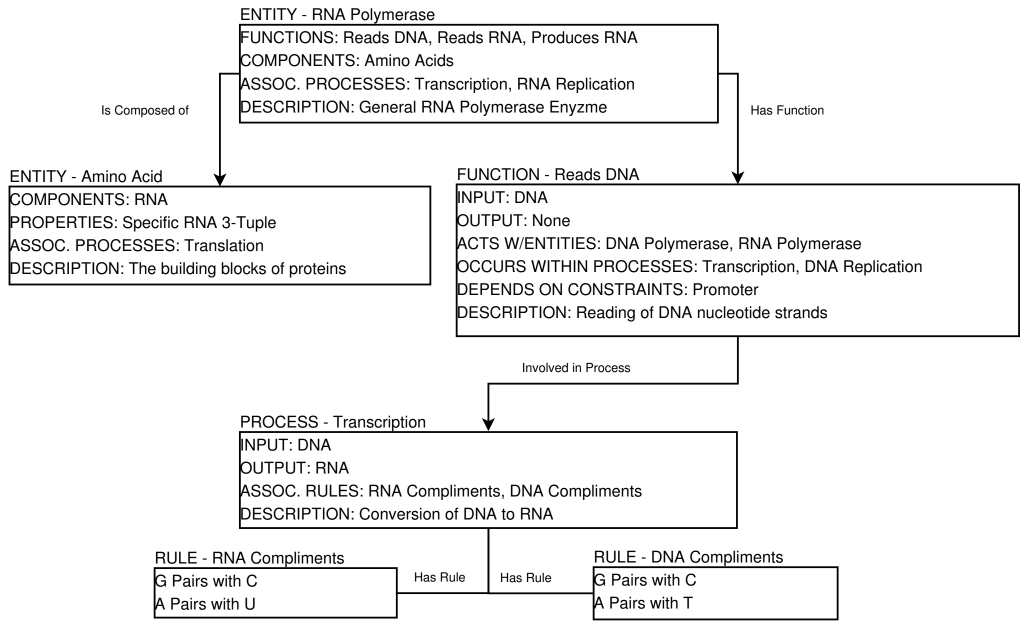
ENTITY - RNA Polymerase
FUNCTIONS: Reads DNA, Reads RNA, Produces RNA
COMPONENTS: Amino Acids
ASSOC. PROCESSES: Transcription, RNA Replication
DESCRIPTION: General RNA Polymerase Enyzme
Is Composed of
Has Function
ENTITY - Amino Acid
COMPONENTS: RNA
PROPERTIES: Specific RNA 3-Tuple
ASSOC. PROCESSES: Translation
DESCRIPTION: The building blocks of proteins
FUNCTION - Reads DNA
INPUT: DNA
OUTPUT: None
ACTS W/ENTITIES: DNA Polymerase, RNA Polymerase
OCCURS WITHIN PROCESSES: Transcription, DNA Replication
DEPENDS ON CONSTRAINTS: Promoter
DESCRIPTION: Reading of DNA nucleotide strands
Involved in Process
PROCESS - Transcription
INPUT: DNA
OUTPUT: RNA
ASSOC. RULES: RNA Compliments, DNA Compliments
DESCRIPTION: Conversion of DNA to RNA
RULE - RNA Compliments
G Pairs with C
A Pairs with U
Has Rule
Has Rule
RULE - DNA Compliments
G Pairs with C
A Pairs with T

Fig. 2: *Example representation of RNA Polymerase Entity and its relationship to other types of example primitives.*

metadata tagging may be necessary for terms to be differentiated when there are repeated terms across domains [3]. This allows a user to make queries using terms that would overlap on the same domain.

4.4 The Low-Level Source Materials

The low-level sources are the raw sources of data that one may capture from the field.

Here, low-level sources represent the final word in implementation specifics for given definitions. One format may define a term or even *hide* a necessary definition within a larger, format-specific definition, while another format defines it completely differently. However, an individual performing queries should not — indeed, does not — need to concern oneself with such implementation specifics. This is the familiar concept of encapsulation.

Consider multiple source-level mappings to the shared-level have been made. An individual with great familiarity of a given format has taken the raw data, with its specific representations of knowledge, and created a hierarchy of definitions and mappings for us. Once this mapping is built, the most precise definition exists at the source-level, but we can still call on the specific implementation from the top of the hierarchy, hiding the source-specifics transformation details.

Consequently, a user interfaces only with the shared-level queries using shared-level terms such as *Packets* and *Timestamp* instead of source-level terms like *RPACK77* and timestamps as *FIELD3*. Even when the correspondences between terms are simple, they are hidden from the user and it is the system that translates shared-level terms into their format-specific implementations at the low level data source.

4.5 Bridging and Linking

Data transformation from the shared (or *operational*) level of the ontology is taken care of through a series of manually mapped *bridges*. These bridges exist as points between levels or *layers* of the overall ontology; each contains two sets of functions: transforming definitions from a lower layer to a higher layer, and vice-versa.

5 Case Study: Mappings with SAM Files

To demonstrate the usefulness of this ontology design, we consider the example of building some mappings with a low-level source format. Many different tools have been developed in the field of bioinformatics for organizing and representing the data, all with their own specific formats; there may even be versions of formats *in response* to the evolving needs of scientists. One such format is the *Sequence Alignment Mapping* (SAM) [4], [5].

The SAM format was originally built to be generic enough to respond to various different alignment formats that were developed. In so doing, it became very flexible, capable of holding a great deal of information in such a small size. However, in order to accommodate translation from various other formats, additional information had to be developed (by way of tagging and other pieces of the format).

The following outlines a short case of mapping between the SAM file low-level format, and the shared-level ontology for querying.

5.1 Data

Research is currently being conducted at the New Mexico Institute of Mining and Technology that involves the examination of fat transport within *rattus norvegicus* (common, brown rat) on two different and distinct diets. The raw data has been sequenced, and now exists in a SAM format [6]. (For the rest of this example, we will be using data from the rats fed the high-fat diet.)

5.2 The SAM Format

The SAM format is similar to that of a CSV file. That is, there are a series of rows (called *alignment lines*) with different, delimited fields. The two exceptions to this rule are found in optional header section (which we do not fully detail in our study here) and a variable number of elements for the "TAG" field defined for each alignment line. (Note that columns in SAM files are not given a named mapping as is done in CSV. The field/column names are implicitly derived by the SAM format.)

The header section (which is optional) defines information that corresponds to the entirety of reads in the file. Such general information as the SAM format version number, the specifics of the alignment lines at various locations, the order in which data will appear, and so on are defined within this section.

Immediately following the header section (should it exist at all) are the rows of alignment lines (Fig. 3). Each of these lines are broken up into eleven distinct fields — all with their own caveats — and an additional, variable-length twelfth field (which contains various tags).

```
SNPSTER3_0511:3:21:542:508#0/1 0 gi|62750345|ref|NC_005100.2|NC_005100 289632 0 36M *
0 0 CTCCTCCTCCCCTTCTTTATGGGTATTCCCCTCCCC HHHHHHHHHHHGGHHHHHHHHHHHHHHHHHHHHHGGG MD:Z:36
RG:Z:s_3_sequence.txt.gz IH:i:2 NH:i:2 YR:i:1

SNPSTER3_0511:3:37:1137:468#0/1 0 gi|62750345|ref|NC_005100.2|NC_005100 305783 0 36M *
0 0 CTCTCATGGCTTAAAGTCCTGCCCGGGACCACCCTG HHHHHHHHHHHHHGHHGHHHHHHHEHGHHHHHHIHH@ MD:Z:36
RG:Z:s_3_sequence.txt.gz IH:i:4 NH:i:4 YR:i:1

SNPSTER3_0511:3:23:1266:1799#0/1 0 gi|62750345|ref|NC_005100.2|NC_005100 329635 0 36M *
0 0 ATTNAATATAGTTCTCGAACTCCTAGCCAGAGCAAT GGG&GHHHFHFGGHHHHHHIHHIHHFIIHHGIGHIG MD:Z:3X32
RG:Z:s_3_sequence.txt.gz IH:i:2 NH:i:2 YR:i:1

SNPSTER3_0511:3:108:1205:762#0/1 0 gi|62750345|ref|NC_005100.2|NC_005100 368226 0 20M2D16M
* 0 0 GTTTTGGGGATTTAGCTCAGGTAGAGCGCTTGCCTA HGHHHHHHHHHHHHHHHHHHHGHIHHHHHIFHHHHH MD:Z:38
RG:Z:s_3_sequence.txt.gz IH:i:5 NH:i:5 YR:i:1

SNPSTER3_0511:3:31:1390:225#0/1 16 gi|62750345|ref|NC_005100.2|NC_005100 389280 0 36M *
0 0 GGGGTCAACCTTCAGTCAGCCCCTCGTAGCGAGGGA EGEGEGGEEGGGGGGDEEAFGGCEAGGGGGGGAGGG MD:Z:36
RG:Z:s_3_sequence.txt.gz IH:i:1 NH:i:1

SNPSTER3_0511:3:103:43:1576#0/1 0 gi|62750345|ref|NC_005100.2|NC_005100 394139 0 36M *
0 0 TACGAATGTGATCAATGTGGTAAAGCCTTTGCATCT HHHHHHHHHHHHHHHHHHHHHGHHHHGHHHHHHHHHH MD:Z:36
RG:Z:s_3_sequence.txt.gz IH:i:4 NH:i:4 YR:i:1
```

Fig. 3: *Several alignment lines from the HFD1.sam file. Fields are delimitted by whitespace. The final TAG field itself has several subfields of tags.*

5.3 The Alignment Section and Lines

The bulk of the SAM file is taken up by the alignment section, filled with various alignment lines. Each alignment line represents a particular portion of sequence that was read when sequencing was done on the observational data. These particular sequences are referred to as *reads*. In an alignment line, these *reads* correspond to some type of match that has been aligned to a reference genome.

There are at least eleven distinct fields, delimited by whitespace, in each alignment line. The twelfth field, TAG, is optional and slightly different, since there can be multiple tags within that field or even none at all (whose multiple tags are also delimited by whitespace). An example alignment line's fields (Fig. 4) are detailed below.

– **QNAME:** *SNPSTER3_0511:3:85:824:196#0/1*
– **FLAG:** *0*
– **RNAME:** *gi|62750345|ref|NC_005100.2|NC_005100*
– **POS:** *1350933*
– **MAPQ:** *0*
– **CIGAR:** *36M*
– **RNEXT:** ***
– **PNEXT:** *0*
– **TLEN** *0*
– **SEQ:** *CACACATACGAGGTGTACTTTGTGTGCAGAATGTGT*
– **QUAL:** *E?9BBCC8?C=?;C844555FFF?F:>==6>CCFCE*
– **TAG:** *MD:Z:36 RG:Z:s_3_sequence.txt.gz IH:i:3 NH:i:3 YR:i:1*

These identifiers correspond to fields (1-indexed, i.e., starting with index 1) 1 through 12 respectively. These are the low-level source representations of data in the SAM format.

5.4 Mapping from the Low-Level Source

This is where one defines the mappings for those shared, operational level terms such as **chromosome** and others. We first examine the alignment lines in the SAM file (many lines are given in Fig. 3, while a single sample line to be used in examples is given in Fig. 4) to better understand their structure.

While building the bridges, it is important to keep in mind that we are trying to match a common, shared-level definition. Let us try this with *Chromosome*.

At first glance, it may be difficult to identify chromosome information within the alignment lines. Fortunately, a domain expert is aware of such caveats and can identify that the chromosomal numbering is given within the RNAME field (3rd element in Fig. 4).

```
SNPSTER3_0511:3:85:824:196#0/1   0   gi|62750345|ref|NC_005100.2|NC_005100   1350933   0
36M   *   0   0 CACACATACGAGGTGTACTTTGTGTGCAGAATGTGT    E?9BBCC8?C=?;C844555FFF?F:>==6>CCFCE
MD:Z:36  RG:Z:s_3_sequence.txt.gz IH:i:3  NH:i:3  YR:i:1
```

Fig. 4: *A random alignment line from the HFD1.sam file; broken up by field.*

The last 2 digits within RNAME (**00** in Fig. 4) denote the chromosome number (0-indexed; meaning 00 is chromosome 1) for this alignment line. Slicing off the last two digits of the RNAME field is then defined as the function for this level's bridge for Chromosome.*Number*.

This transformation provides a mapping from the low-level source to the shared-level.

$$f_{RNAME-Chromosome.Number} \in B_{L_G L_S} : RNAME \rightarrow Chromosome.Number\,. \tag{5}$$

If one were to code the function by-hand, one might define it as (in pseudo-code):

```
FUNC RNAME-TO-CHROMNUM(RNAME): return RNAME[-2:]
```

This is one of the functions that the bridge would contain. Recall that bridges hold the functions necessary for transformations. As a high-level definition is transformed as part of a query (or a low-level definition is transformed), the bridges know which function to call for which definition. As a query descends the layers, each definition is transformed accordingly via these functions.

This satisfies a complete mapping for the chromosome number in the SAM file format, version 1.0. This mapping goes into a collection for the SAM format with this specific version, which is kept within a file and pointed to by the system's framework. This helps to maintain a proper organization for the bridges and the formats they are linked to.

We examine alignment uniqueness next, since this is a particularly interesting "free-form" case of the SAM fields. In order to find uniqueness information, we have to consult the twelfth field, TAG. The information from the TAGs given in our example alignment line are:

```
MD:Z:36   RG:Z:s_3_sequence.txt.gz   IH:i:3   NH:i:3   YR:i:1
```

Each tag is given a name, a type, and a value. Anyone familiar with the SAM format can look at these tags and tell right away whether or not an alignment line is unique. Such information is located within tag "IH."

The “IH” flag denotes the number of alignments in the entire SAM file that have the same template name. The type for our “IH” flag is “i” meaning that the value will be an (i)nteger. In this case, it has value 3. If this template (QNAME) were unique, the value for IH would be 1.

To map this value, the low-level definition D_G is set as whatever the file defines. However, it cannot simply suffice to set the definition as TAG, since the TAG field defines many tags. Instead, one must set a more appropriate definition on a per-tag basis.

To do this, we create an intermediate level up from the low-level source. (This is to illustrate that an arbitrary amount of intermediate levels can be built to suit the needs of the low-level source, combining definitions from various levels if necessary.) The intermediate level splits up the TAG field into its component parts (the different tags).

– **Tags:** *IH*, *MD*, *RG*, *NH*, *YR*

From the low-level source to this intermediate layer, the functions defined in the bridge are to separate the Tag field up by cutting it into pieces, and setting those pieces (the individual tags), into a new set of definitions:

This transformation provides a mapping from the low-level source definition to the intermediate definition of “Tags.”

$$f_{TAG-Tags} \in B_{L_G L_1} : \quad TAG \rightarrow Tags.IH,\ Tags.MD,\ Tags.RG,\ Tags.NH,\ Tags.YR\ . \tag{6}$$

The internals of this function might look like the following (in pseudo-code) :

```
FUNC TAG-EXPANSION(TAG):
        local dictionary tag-map
        for Tag in TAG:
                tag-map[Tag[:2]] = Tag[Tag.findLast(':'):]
        return tag-map
```

To go the other direction, the inverse function suffices:

$$f^{-1}_{TAG-Tags} \in B_{L_1 L_G} : Tags.IH,\ Tags.MD,\ Tags.RG,\ Tags.NH,\ Tags.YR \rightarrow TAG\ . \tag{7}$$

Each tag in the example line (IH, MD, RG, NH, and YR) is mapped to the “Tags” definition in this intermediate layer. This provides a quick way for one to reference and use it in other intermediate layers as needs be, having singled-out the information from the larger sample line itself. Now the tags are split up, and from each of them, one can answer questions such as “uniqueness”.

Since the IH tag denotes uniqueness amongst all alignments in a SAM file, we can instead set the IH tag from our intermediate-level “Tags” definition to be “unique”, mapping the value from the intermediate layer to the shared layer. Only a boolean value of True or False is necessary. Also note that this particular tag can be used to set the Total Matches for the alignment, but in that case, the precise integer value becomes important (as opposed to being 1 or otherwise).

$$f_{Tags.IH-Alignment.Unique} \in B_{L_1 L_2} : Tags.IH \rightarrow Alignment.Unique\ . \tag{8}$$

The function for this transformation might be defined (in pseudo-code) as:

```
FUNC IH-IS-UNIQUE(Tags.IH):
        if Tags.IH == 1: return True
        else: return False
```

Note, however, that it may not be possible to reconstruct the IH tag merely from the “Unique” definition. Instead, the only way to get back the full, original value of this IH tag is from the “Total Alignment Matches” portion of the Alignment Definition. That is, there may not exist a suitable inverse function for uniqueness in the case where uniqueness is false, and instead, the function to go back to the full IH tag must be:

$$f^{-1}_{Tags.IH-Alignment.Unique} \in B_{L_2L_1} : Alignment.TotalAlignmentMatches \rightarrow Tags.IH \tag{9}$$

It should be clear that other mappings for the format — and indeed all different formats — can be constructed similarly. Some may be more simple, while others are far more complex.

5.5 Some Example Queries

Now that the low-level source information has bridges set up to allow for definition transformation, we can make queries on the data. The overall query is broken up into sub-queries, answering questions in parallel on the source files used with the aid of the bridges, transforming the SAM format into something more recognizable by all domain experts familiar with bioinformatics, but not necessarily the SAM format itself. Some examples are given below.[1]

– Which sequence is repeated the most in the HFD1.sam data, and does it represent a new gene?

```
USES FORMAT SAM, GFF3

(SELECT MOST REPEATED Sequence FROM FILE HFD1.sam) AS RepeatedSeq
(SELECT Gene FROM FILE RATGENOMEv3_4.sam) AS KnownGenes

SELECT Gene.Expression
FROM FILE HFD1.sam
WHERE NOT KnownGenes
AND WHERE RepeatedSeq
```

This will return all the possible pileups from the most repeated sequence in the HFD1.sam data that sufficiently overlap to denote a “new” gene. By forming sub-queries to count the pileup start locations, examining their overlaps, and finding runs of sufficiently piled up sequences, the system can have certain definitions for what constitutes — and then matches to — a possible gene region such as this.

– Which non-coding regions of the genome, near the coding region for the leptin gene, are expressive?

```
USES FORMAT SAM, GFF3

(SELECT Chromosome.Number, Sequence.StartBase.Number, Sequence.EndBase.Number
FROM FILE RATGENOMEv3_4.gff
```

[1] Queries are presented as small snippets of SQL-like code, but we want to stress that biologists will not have to learn SQL to make use of the system. Their high-level queries (in a language that will involve biological concepts — still a work in progress) will be translated into similar code.

```
WHERE Gene.Name=lep) AS LeptinGene

SELECT Gene.Expression
FROM FILE HFD1.sam
WHERE Chromosome.Number=LeptinGene.Chromosome.Number
AND Sequence.Start NEAR LeptinGene.Base.Start
```

Here, one is examining non-coding regions of rats that have been given a high-fat diet, exploring any up-regulated portions of the genome around leptin, a protein known to be involved in fat transport. This is to aid a researcher in a hunch regarding non-protein-coding regions of the genome being relevant in dietary aberrations beyond the simple analysis of coding regions alone.

Combining a sample with a particular reference is common in bioinformatics and is necessary for asking certain types of questions (here, the reference genomic information comes from a file in the GFF3 format [7]). With this model, a biologist can — in a single query — take two heterogeneous source formats, combine them together, and make complex queries that result in meaningful answers for questions. Questions that may prove useful in understanding diseases. Practically, any question whose information exists within the source formats is answerable. And this is only an examination of that particular field. This model can be applied to any field with source formats of any type and number.

Such an ability to make queries as this across many different, possibly overlapping source formats at once is impossible with current tools available. There is no single language or system by which such queries can be made with the richness of object-oriented design and the flexibility of any source format the mind can dream up, alongside coexisting, multiple domain-level ontologies, each with their own definitions.

6 Analysis

The example of SAM and GFF files is a fairly complex one, and while this paper is not extensive in its exhibition of how all transformations of that format are to be performed, they can be built as prescribed by this proposed design methodology

This proposed design still "suffers" from the same concern as existed in the KRAFT system, mainly that arbitrary — even wrong — mappings can be defined. This is a fundamental risk of crowd-sourced solutions. Since we imagine bridge definitions are just files that can be sent to others, we believe that a type of self-checking will go on amongst the community of researchers that make use of this design. More accurate, perhaps even faster, bridging definitions for the different formats will be accepted as the best, used most often, and signed off on, while those mappings that map data incorrectly will wither and die.

Of course, the question of how to ensure that bridge definitions do not produce poor results before using them is another task that we are currently looking into. The ability to verify that a mapping produces good, accurate results is something that we would certainly like to have. This would both cut down on conscious attempts to distort results as well as notify users about honest mistakes in transformation of definitions.

Facilitating the creation of bridging will require a great deal of work, since not everyone knows how to make a function to transform data into various definitions for different link levels. It will, however, be critical in producing something that experts want to use. For those that are comfortable with building functions, the ability to just program them in should also be available.

7 Conclusion

Ontologies offer a way to organize and transfer knowledge, but they suffer when overlap across domains occur or formats differ. In such cases as arise in merging/linking ontologies, some form of automation would be helpful in resolving issues of clarity and yield a single, well-formed and unambiguous ontology. Unfortunately, there are many problems that arise in pursuing such an automated approach.

Instead, we have presented a solution that relies on using the work of experts who build an ontology of their field of expertise, e.g., biology, with that of experts who can build transformations on data representations on a per-format basis. The result benefits practitioners of the field, who have working knowledge of terms within their fields, but neither possess specialized knowledge of a given data format, nor are technically trained to exploit it.

By defining shared-level terms for a given domain and building bridges from low-level sources up to those domains, queries can be made on all variety of source formats. These bridges can then be modified and shared by the community of researchers. Further, it is possible to query across an ensemble of data in different formats allowing comparison, contrast, and union of the diverse information content that no format-specific software system would enable.

8 Acknowledgements

We would like to thank Dr. Rebecca Reiss for her assistance in helping us formulate queries and realizing how those queries could be understood in a piece-wise manner from SAM files. We also thank anonymous reviewers whose comments have helped improve this paper.

References

1. Bouquet, Paolo, et al. *C-OWL: Contextualizing Ontologies.* The Semantic Web-ISWC 2003. Springer Berlin Heidelberg, 2003. pp. 164-179.
2. Preece et al. *The kraft architecture for knowledge fusion and transformation.* In Proceedings of the 19th SGES International Conference on Knowledge-Based Systems and Applied Artificial Intelligence (ES99). Springer, 1999.
3. Xu, Zhichen, et al. *Towards the Semantic Web: Collaborative Tag Suggestions.* Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland. 2006.
4. Li, Heng, et al. *The Sequence Alignment/Map Format and SAMtools.* Bioinformatics 25.16 (2009): 2078-2079.
5. *Sequence Alignment/Map Format Specification.* SAMTools. The SAM/BAM Format Specification Working Group, 29 May 2013. Web. 17 July 2013. http://samtools.sourceforge.net/SAMv1.pdf
6. *Fat-Rat SAM Data.* NMT Biology Bioinformatics Portal. New Mexico Institute of Mining and Technology. Web. 03 Aug. 2013. http://bioinformatics.nmt.edu/
7. *The Sequence Ontology - Resources - GFF3.* The Sequence Ontology. Web. 18 Sept. 2013, http://www.sequenceontology.org/gff3.shtml
8. Mortazavi, Ali, et al. *Mapping and Quantifying Mammalian Transcriptomes by RNA-Seq.* Nature methods 5.7 (2008): 621-628.