Development of an Ontology for the OMEGA EP Laser System

Richard C. Marron

Allendale Columbia School

Rochester, New York

Advisor: Richard Kidder

Laboratory for Laser Energetics

University of Rochester

Rochester, New York

February 2006

Abstract

An ontology is a working model of objects and relationships in a particular domain of knowledge. It can allow a computer to infer relationships between separate objects. There are several different types of relationships: transitive, symmetrical, inverse and functional. The use of these relationships together allows a complex ontology to be created, an ontology that a computer can infer knowledge from. Using these relationships an ontology could understand that if all Chardonnay comes from France, and a particular wine is a Chardonnay, then this wine must be from France.

The base for a laser system ontology was created to allow a computer to understand that if the beam passes through one object in the beam path it will then pass though the next object. These relationships are the building blocks for the ultimate goal of a semantic web. This semantic web will allow a user to ask a question about the laser systems, and the service will be able to answer this question. This will eliminate the tedious search process through documents to find small hidden facts.

I. Introduction

The ultimate goal of an ontology is to create a system for storing and searching data, a single uniform repository for all data used at the Laboratory for Laser Energetics (LLE). LLE currently has an extensive database system with vast amounts of information. This data is stored in many complex repositories in many different formats ranging from Word documents to schematics to spreadsheets. These repositories are difficult to navigate without previous experience in the particular database. The current databases are the Project Data Management (PDM) database and the Oracle Database. The PDM system can be navigated in three ways: a hierarchy of links, specifying the range of the documents viewed, a Google search bar, or a serial code. The document can then be viewed in PDF format. The Oracle Database is simply a series of tables with information relevant to shot operations [1]. The ontology would unify these databases into one.

This system would not only contain the information represented by the text documents, the spreadsheets, and the images, but also the information coded for in the ontology itself, and this information coded into the ontology would not only be information but would be knowledge. The key difference between information and knowledge is that knowledge contains relationships defining how the elements combine and work together. Knowledge can allow new information to be inferred creating a dynamic wealth of data, while information alone can only be understood as it is.

Once one has created an ontology, and therefore a knowledge base, the possibilities are endless. Some possible applications relevant to LLE are question-answering systems that produce "knowledgeable" answers, applications to determine the current path of a beam line and provide information relevant to the beam line, and applications to display shot data in a more efficient manner.

2. Process

2.1. Know the Application

The future functions of the ontology determine the necessary structure. In order to determine the information and the relationships that must be put into the ontology the final applications must be known and carefully analyzed [3]. There is too much information relevant to LLE for all of it to be coded into the ontology, and therefore, it must be reduced to information relevant to the application at hand. If one were creating an ontology solely for the purpose of determining the beam path, information regarding the requirements of the UV calorimeter would be irrelevant. However if one were creating an application for question answering regarding the diagnostic tables this information would be relevant and should be included in the ontology. While it is the relationships between the information that makes an ontology special, the base of information defines the applications that it can be used for.

Knowing the future applications before starting the ontology is also necessary in determining the proper relationships to include. Relationships between components of the laser system such as beamGoesTo, which defines where the beam goes immediately after leaving a component, can be used in the application determining beam path. However, a relationship between components such as hasDiagnostic, which determines which diagnostics are branched off a component, would not be relevant to simply determining the beam path [3].

2.2. Individuals

Once the applications are determined, the creator of the ontology must determine which objects to include. The objects will be known as individuals. They are anything that can be seen or touched; in essence, anything that exists[4]. The Short Pulse Highly Reflective Mirror 2 is considered an individual; however, a short pulse mirror in general is not considered an individual but rather a class. Once all of the individuals that will be included in the ontology have been determined, choosing the proper class structure for the ontology is the next step.

2.3. Classes

Classes can be anything conceptual, anything that cannot be felt [4]. A good way of determining whether an object would be a class or an individual is if one would refer to the object using the article “a” or the article “the.” When the article “a” is used to describe the object it is normally a general term, known as a class, under which an individual would fall. When the article “the” is used it is normally a specific object that exists and therefore an individual. The class system is important as it is used to determine the basic hierarchy of the ontology. An ontology is based off a tree of “is a” relationships. These relationships are crucial to the function of the ontology as they provide a map for the ontology, a static structure that allows specific objects to be located.   Figure 1 displays a portion of the class hierarchy for the ontology that was created for the EP laser system.

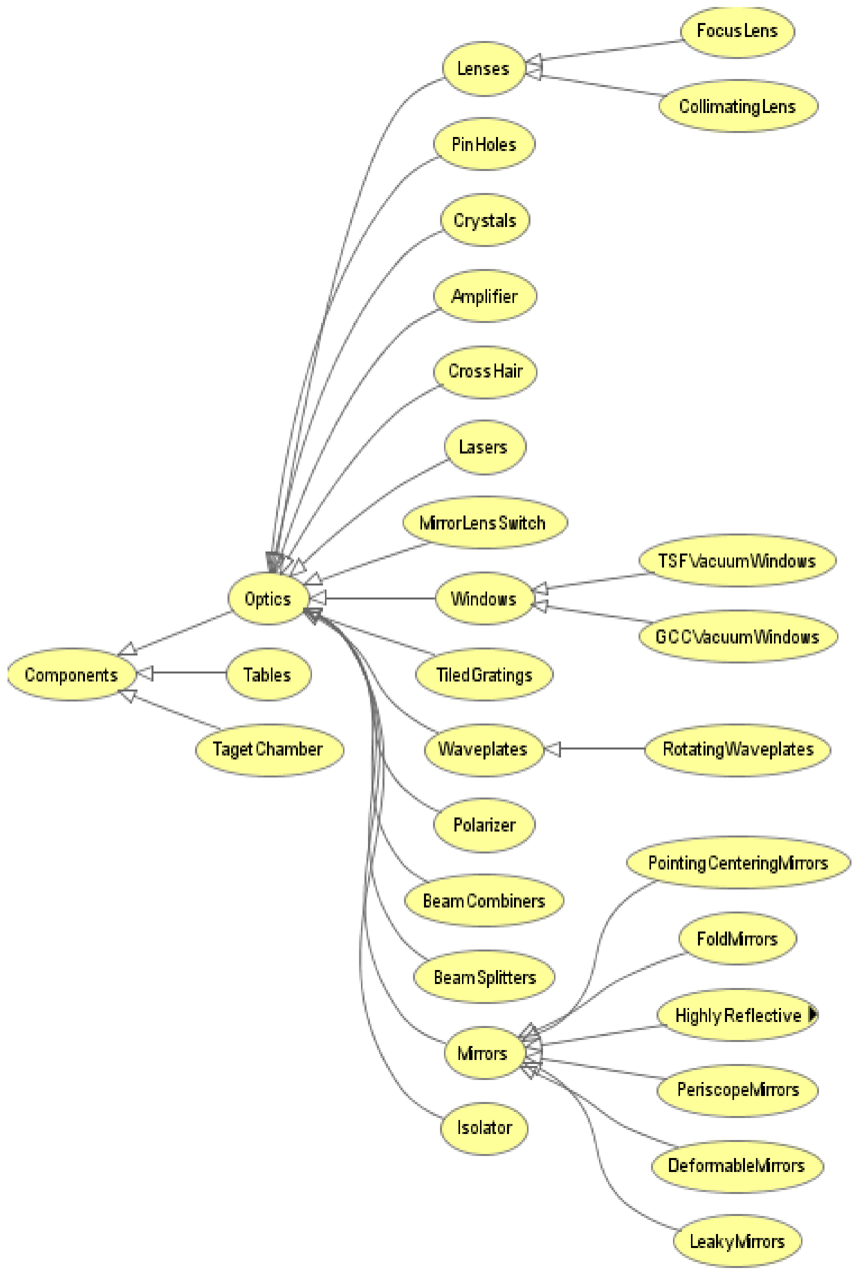

Figure 1. A portion of the class structure for LLE's Ontology, each line representing an "is a" relationship.

Inside each class are a series of individuals that are a part of the class. For example, the sixth short pulse highly reflective mirror would be located in the class "short pulse highly reflective mirrors."

2.4. Properties

After both the individuals and the classes have been determined and essentially the basic structure of the ontology is complete, the properties of the objects must be determined. These properties are what change the ontology from information to knowledge. They represent the relationships that each object has with each other. There are four separate types of relationships between objects; they are inverse, transitive, symmetrical, and functional [4]. Depending on the type of relationship, it will behave differently.

2.4.1. Inverse Relationships

If a property is an inverse property, it will have a corresponding property that is its inverse [4]. A common example of an inverse relationship is a father-son relationship. If John is Billy's father, and the computer knows that father and son are both corresponding inverse relationships, then it can infer that Billy is John's son. This type of a relationship would apply to LLE in the situation of the beamGoesTo and beamComesFrom relationships. If the computer knows that after the Third Infrared Highly Reflective Mirror the beam goes to the First Tile Grating Assembly through the use of the beamGoesTo relationship, then the computer can infer that the beam came from the Third Infrared Highly Reflective Mirror before hitting the First Tile Grating Assembly (Fig. 2). This inference can be made based on the inverse

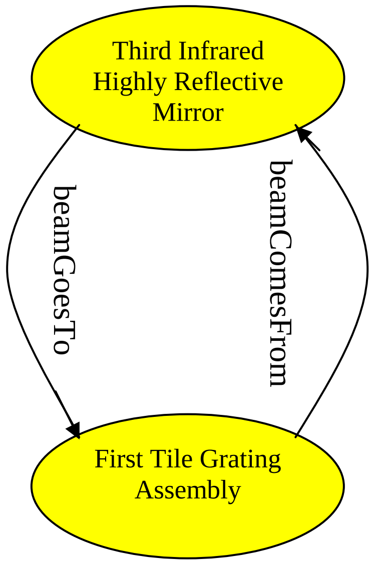
Third Infrared Highly Reflective Mirror
beamGoesTo
beamComesFrom
First Tile Grating Assembly

Figure 2. Diagram of an inverse relationship

description of the beamGoesTo and beamComesFrom relationships.

2.4.2. Transitive Relationships

If property P is transitive, and property P is applied between object a and object b and property P is applied between object b and object c, then it can be inferred that property P also relates object a to object c [4]. A common example of a transitive property is the property relatedTo. If Joan is related to Paul and Paul is related to Joe, then it can be inferred that Joan is related to Joe. This can be used in the ontology for LLE with the relationship beamGoesTo. If the beam comes from the OPCPA output spatial filter and then goes to the short pulse apodizer and then goes to the apodizer output spatial filter, the computer can infer that after the beam goes through the OPCPA output spatial filter it will later go to the apodizer output spatial filter (Fig. 3).

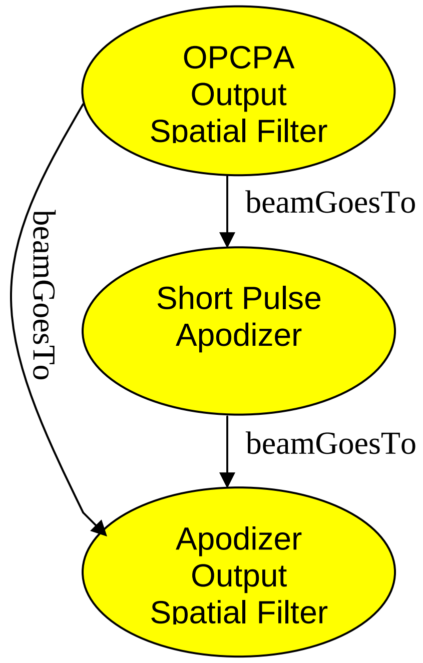

Figure 3. A diagram representing a transitive relationship. The computer inferred the relationship on the far left.

2.4.3. Symmetrical Relationship

If property P is symmetrical, and property P relates object a to object b, then property P also relates object b to object a [4]. A common example of a symmetrical property is the sibling relationship. If Andy is Dale’s brother, then Dale must be Andy’s brother. This can be used in the ontology for LLE with the relationship isNextTo. If the Discrete Zoom Spatial Filter is next to the Periscope to the Laser Bay, and the isNextTo relationship is symmetrical, then it can be inferred that the Periscope to the Laser Bay is next to the Discrete Zoom Spatial Filter (Fig. 4).

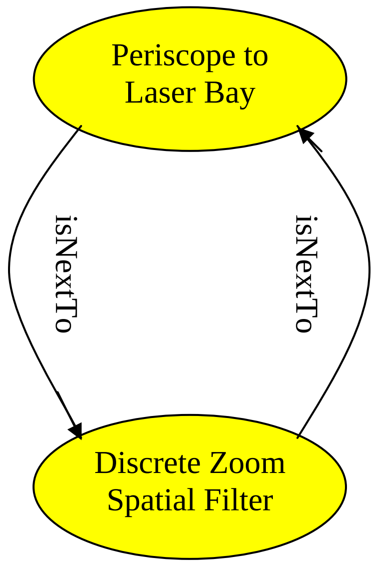

Figure 4. Diagram of a symmetrical relationship

2.4.4. Functional Relationship

If property P is functional, and property P relates object a to object b, then object a is related to no other object besides object b by property P [4]. A common example of a functional relationship is the birth mother relationship. If Jane is Dan's birth mother, it can be inferred that all other individuals are not Dan's birth mother. However, if Janine is also described as Dan's birth mother then it can be inferred that Jane and Janine are the

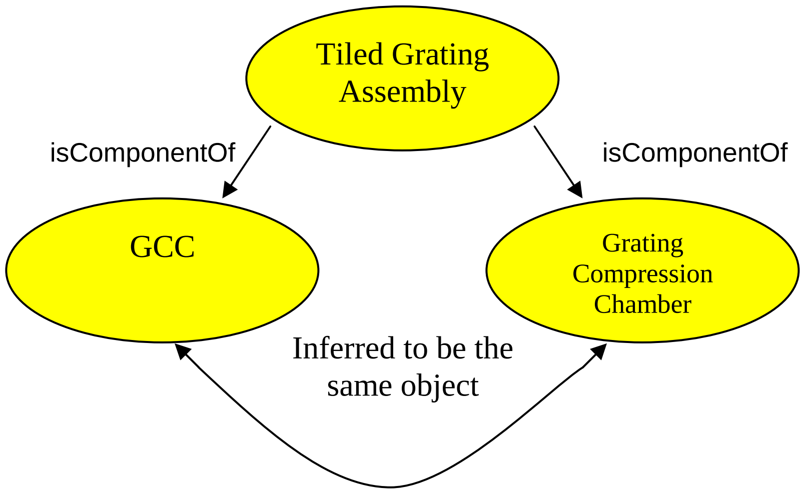

Figure 5. Diagram of a functional property and how it can help the cmputer infer relationships.

same person. A Functional relationship can be implemented in LLE's ontology in the relationship isComponentOf. If the tiled grating assembly is a component of the grating compression chamber and the GCC, it can be inferred that the grating compression chamber and the GCC are the same (Fig. 5).

2.5. Domain and Range of Relationships

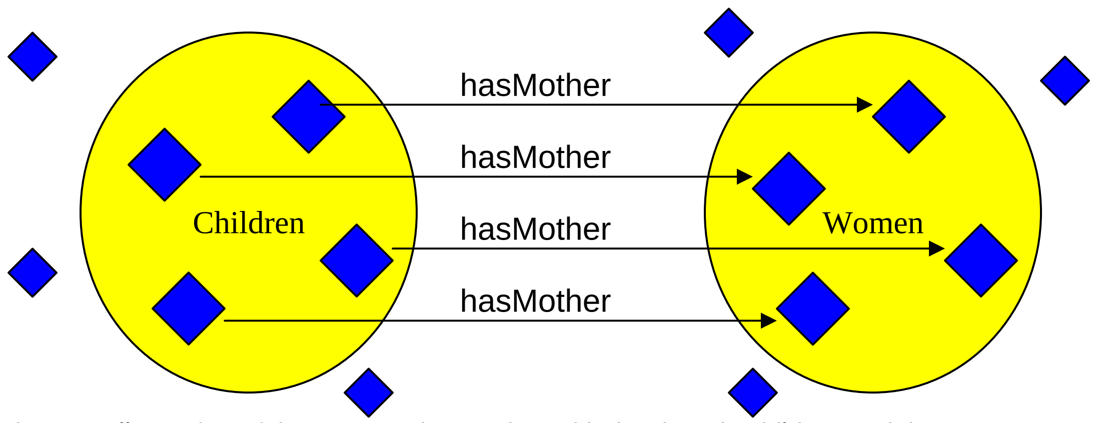

Figure 6. Illustration of the property hasMother with the domain children and the range women. The property can only be applied between objects in the domain and range.

The domain and ranges of a relationship must also be determined. This will stop the computer from making incorrect inferences. A property links an object from the domain to an object in the range [4]. A common example of this practice can be seen in the relationship hasMother (Fig. 6). If the domain of the relationship hasMother is all children and the range of the relationship hasMother is all women, then the computer will know not to infer that Dan has Bill as a mother.

While these properties are quite basic, they allow the computer to make these important inferences that are necessary to change the data represented from information to knowledge that can be understood by a computer. A computer is thus allowed to infer its own information.

2.6. Restrictions

After the applications, individuals, classes, and relationships for the ontology have been determined, the next step in the creation of the ontology is determining the proper restrictions on the classes. Restrictions restrict which individuals can be forced or inferred into the class. They have three basic components: the restriction type, the property and the filler. There are three basic categories of restriction types: quantifier restrictions, cardinality restrictions, and hasValue restrictions.

2.6.1. Quantifier Restrictions

Quantifier restrictions regulate what must be inside a class. There are two types of quantifier restrictions: an existential quantifier and a universal quantifier [4]. The existential quantifier can be thought of as an “at least one” restriction [4]. For example, if the existential quantifier is applied to a class with the property beamGoesTo and the filler short pulse highly reflective mirror, then for an object to be part of the particular class it must send the beam to at least one short pulse highly reflective mirror.

The universal quantifier is very similar. However, it is an “all” restriction [4]. If the universal quantifier is applied to a class with the property beamGoesTo and the filler short pulse highly reflective mirror, then for an object to be a part of the particular class it must send the beam to only short pulse highly reflective mirrors.

2.6.2. Cardinality Restrictions

Cardinality Restrictions regulate how many objects can satisfy a certain property. A minimum cardinality restriction sets a minimum number of times that an object must satisfy a property to be considered a part of the class [4]. An example of this would be in the class beam splitter. There is a minimum cardinality restriction stating that an object must go to two or more other objects to be considered a beam splitter.

A maximum cardinality restriction sets a maximum number of times that an object can satisfy a property and still be considered a member of the specific class [4]. An example of this would be in a class such as a nonBeamSplitter in which a maximum cardinality restriction of one would be placed on the beamGoesTo property with the filler as the class component. This makes sure that the beam does not split after leaving the object.

An exact cardinality restriction sets an exact number of times that an object can satisfy a property and still be considered a member of the specific class [4]. An example of this would be in the class beamTerminator in which an exact cardinality restriction of zero would be placed on the beamGoesTo property with the filler as the class component. This restriction would state that in order for a component to be in the class beamTerminator it must not reflect the beam to another component.

2.6.3. HasValue Restrictions

HasValue restrictions are very similar to quantifier restrictions. However, instead of having the filler be a class the filler is an individual [4]. An example of this occurs in the class OMEGA backlighter OAP. The hasValue restriction allows us to state that if the object is a member of the class OMEGA backlighter OAP then it must come from the sixth short pulse highly reflective mirror.

These restrictions all display a necessary condition that an object must comply with in order to be a member of the class. There is another type of restriction that can be placed on these classes and that is a necessary and sufficient condition. These conditions state that if an individual satisfies the condition then it will be inferred into the class and treated as if it were a forced member of the class [4].

2.7. Programs

After the applications, individuals, classes, relationships, and restrictions have all been determined, the next step is to code for the ontology. An open source program exists that provides a convenient interface, eliminating the need for actually coding the ontology. This program is Protégé. It is very well documented and has a useful support forum for any questions. This program encodes the ontology in a Web Ontology Language (OWL). OWL is an extension of the language RDF [1].

With this program one can apply the planned ontology and connect the individuals through the relationships. This is a very tedious task and the computer will infer some of the relationships. However there are still many relationships that the computer cannot infer such as the beamGoesTo relationship, which must be manually applied.

2.8. Applications

The next step is creating the applications previously determined. This can be accomplished using Java and the Jena Library. [2] The Jena Library provides many very useful classes and methods that allow one to navigate the ontology, extract information, and manipulate the ontology. All of these actions are necessary to create an application that fully uses the powers of an ontology.

3. Discussion

Currently the applications of the ontology are limited by technology. The logic proof and trust layers of the semantic web are all under research. Once these layers are completed the applications of an ontology are endless. Applications such as natural language question answering will then be in reach. However, all of the future implementations are dependent on having an extensive ontology ready when the technology catches up.

One problem that is posed is if we do not understand the future applications of the ontology how can we create a good ontology at the moment? The answer to this question is that we can not be sure that we are creating a suitable ontology for future applications; however, if we create an ontology for current applications, there is a very good chance that there will be overlap and portions of the already created ontology will be useful in future applications.

Currently the use of the ontology could be supplemented by other technologies such as an Oracle database. However, the benefit to using an ontology over an Oracle database is not what it can do now, but rather it is what the ontology will be able to do in the future.

4. Acknowledgements

I would like to thank Richard Kidder, Daniel Gresh, Adam Kalb, Collin Kingsley, and Thang Nyugen for helping me with my questions regarding the laser system and ontolgies. I would also like to thank Stephen Craxton for giving me this wonderful opportunity to work at the Laser Lab this past summer.

5. References

1. D. Gresh. “Implementing a Knowledge Database for Scientific Control Systems.” 2006. <http://www.lle.rochester.edu/pub/HS_reports/2006/Gresh_Dan.pdf>
2. D. Gresh, private communication.
3. H. Chen. “My Experience in Building Ontology-driven Applications.” eBiquity Group Meeting. PowerPoint. February 9, 2004 <ebiquity.umbc.edu/get/a/resource/15.ppt>.
4. M. Horridge et al. “A Practical Guide To Building OWL Ontologies Using The Prot´eg´e-OWL Plugin and CO-ODE Tools Edition 1.0.” The University Of Manchester, August 27, 2004. <http://www.coode.org/resources/tutorials/ ProtegeOWLTutorial.pdf>