ONTOLOGY-BASED APPROACH TO ENABLE FEATURE INTEROPERABILITY BETWEEN CAD SYSTEMS

A Thesis
Presented to
The Academic Faculty

by

Sean Tessier

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Mechanical Engineering

Georgia Institute of Technology
August 2011

ONTOLOGY-BASED APPROACH TO ENABLE FEATURE INTEROPERABILITY BETWEEN CAD SYSTEMS

Approved by:

Dr. Yan Wang, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Chris Paredis
School of Mechanical Engineering
Georgia Institute of Technology

Dr. David W. Rosen
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: May 19, 2011

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Yan Wang for his guidance and support. His encouragement, intellectual input, and feedback pushed me to examine the problem more critically. His support was an invaluable contribution to this work.

I would also like to thank our colleagues from the University of Massachusetts Amherst, John Altidor and Dr. Jack Wileden, for their assistance and contributions. The exportmodel program that John developed and shared with us was a major contribution, which made my humble implementation possible. Their input also greatly facilitated my understanding of ontologies and this area of computer science research.

I would also like to express my gratitude to Dr. Ali P. Gordon, for first encouraging me to consider graduate school. His emphasis on maintaining integrity and honor within the engineering field had a big impact on me and is something that I will always remember from his classes.

Finally, I would like to express my gratitude to my family for all their love and encouragement. I am truly proud to be the first college graduate from my family, and I know it would not have been possible without them pushing me succeed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Data interoperability between computer-aided design (CAD) systems remains a major obstacle in the information integration and exchange in a collaborative engineering environment. The standards for CAD data exchange have remained largely restricted to geometric representations, causing the design intent portrayed through construction history, features, parameters, and constraints to be discarded in the exchange process. In this thesis, an ontology-based framework is proposed to allow for the full exchange of semantic feature data. A hybrid ontology approach is proposed, where a shared base ontology is used to convey the concepts that are common amongst different CAD systems, while local ontologies are used to represent the feature libraries of individual CAD systems as combinations of these shared concepts. A three-branch CAD feature model is constructed to reduce ambiguity in the construction of local ontology feature data. Boundary representation (B-Rep) data corresponding to the output of the feature operation is incorporated into the feature data to enhance data exchange.

The Ontology Web Language (OWL) is used to construct a shared base ontology and a small feature library, which allows the use of existing ontology reasoning tools to infer new relationships and information between heterogeneous data. A combination of OWL and SWRL (Semantic Web Rule Language) rules are developed to allow a feature from an arbitrary source system expressed via the shared base ontology to be automatically classified and translated into the target system. These rules relate input parameters and reference types to expected B-Rep objects, allowing classification even when feature definitions vary or when little is known about the source system. In cases when the source system is well known, this approach also permits direct translation rules to be implemented. With such a flexible framework, a neutral feature exchange format could be developed.

CHAPTER 1

INTRODUCTION

Computer-Aided Design (CAD) systems have become invaluable tools for engineers in all fields of engineering. Today, it is standard practice to use commercial CAD systems for various design tasks. CAD systems not only fill the vital role of conveying the shapes of designed parts, but they are also used to convey other design information, such as dimensions, tolerances, materials, and manufacturing processes. These CAD systems are also used with various Finite Element Analysis (FEA) and Computer-Aided Manufacturing (CAM) tools to better facilitate the design process. As computers became more powerful, CAD systems evolved from simple 2-Dimensional drawings to fully parameterized, 3-Dimensional solid models. In order to facilitate the product development process and the creation of more complex models, tools to automate geometry creation were developed. These tools modeled geometry through a sequence of instructions, where specific types of geometry changes are defined by features, and the construction history of a CAD part is stored as a combination of these features. Within the CAD domain, a feature can be considered a region of a part with a particular geometric or topological pattern. When constructing solid geometry, form features which represent specific shape concepts and are defined using parametric information are used to modify the geometry of the part. Feature-based design simplified the design process by requiring far less input from the user and automating the calculation of the geometric data. As a result, solid modeling in current CAD systems is almost exclusively done through parametric feature-based design due to its added ease of use, ability to convey design intent, and the ease with which designs can be altered or edited. Further information on feature-based design will be presented in the following chapter. With such systems, it is possible to create models dependent on sets of user defined variables,

allowing a computer to automatically create entirely new models optimized to suit different design goals when integrated with optimization programs. The current trend for CAD systems is to further improve and expedite the design process and aid in collaborative design by integrating various design tools to work seamlessly together.

However, despite the advancements in CAD systems, there are several problems that still stand in the way of such collaborative design processes. One of the more prevalent problems impeding collaborative design environments is the difficulties involved in the exchange of data between heterogeneous systems. This is especially true concerning the exchange of data between different CAD systems. The resources spent translating data between different CAD formats, reprocessing the data in different applications, and redesigning due to information loss can be very costly. Additional resources may also be spent on error checking and correction that may be necessary depending on the data exchange process. Data exchange standards such as the Standard for the Exchange of Product Model Data (STEP) have been developed to facilitate data exchange through geometric representations, but this comes at the expense of the various construction elements that convey the design intent. Exchange of parametric data is still rather difficult and time consuming. Ad hoc commercial translators offer feature-based conversion between CAD systems, but they are prohibitively expensive due to high development and maintenance costs and problems that arise between incongruous feature sets often require resolution via the user. A general automated solution is needed to reduce time and resource expenditures in a collaborative engineering environment. Interoperable CAD model generation, intent and knowledge capturing, and semantic-level information exchange are therefore needed to enable such automation.

Importance of Exchanging Parametric Feature Data

Features are the basic means of geometric construction in modern CAD systems. Unfortunately, these features have no standard definition, and as a result, they can vary

amongst different CAD systems. Further exacerbating this problem is the inherent differences in the way each CAD system represents and stores data. Most CAD file formats are proprietary, so direct exchange is limited to systems which have access to information about the other format. This occurs most often in different CAD systems owned by the same company, but may also be possible by reverse engineering the file format of a competitor. Some companies offer data exchange through translation programs or services using their own proprietary intermediate file format. However, this approach has its limitations due to the added cost, computational resources, and the need for maintenance as CAD systems change. Additionally, because the various CAD systems store models as an amalgam of features, exchange of model data between differing CAD systems can become even more problematic if some of the features used in one system do not exist in another or are defined by a different set of parameters.

In an attempt to better facilitate data exchange between various CAD systems, a number of open intermediate file formats for storing the geometric data contained in the CAD model have been developed. For sake of clarity, the CAD system in which the file was created will be referred to as the source system, and the system that it will be sent to as the target system. In this process, the source system exports the data by converting it to an intermediate format, which can then be read by the target system. Examples of these include the Initial Graphics Exchange Specification (IGES), the Standard for the Exchange of Product model data (STEP), Data Exchange Format (DXF), and Parasolid format. Unfortunately, these intermediate formats only focus on the exchange of the final geometry of the part, so valuable information such as construction history, features used, parameters, and constraints are all lost in the exchange.

This is a significant problem in a collaborative engineering environment because the data lost is what conveys the design intent of the person who created that part. The choice of features is often related to the purpose of the part, which parameters are important to the design, and even what types of manufacturing process may be used.

Additionally, the loss of the parametric relationships between each feature of the part makes modifying the parts in the target system very difficult, if not impossible. This means that any revisions, which are common in the design process, can only be done efficiently on the system in which the part was originally created. This inability to efficiently modify parts amongst different CAD systems means that most design groups that need to collaborate on a design are forced to use the same software. Due to this hindrance, a company's choice of CAD system in industry is often dictated by the systems used by their business partners. As collaborative design environments become more distributed amongst groups of specialized engineers, this issue will become far more problematic. Any company wishing to work with diverse client groups may need to invest in multiple CAD system solutions in order to collaborate effectively, increasing both the cost and resources expenditures to maintain a catalogue of parts amongst multiple CAD systems.

Research Overview

In order to allow exchange of CAD data with little to no data loss, several research groups are working on interoperable feature modeling. The goal of these approaches would be a neutral format, similar to the intermediate format used by commercial translation companies, but different in several key ways. The commercial translation companies usually take a static identify-and-map approach, meaning they use their knowledge of the source and target CAD systems to create a one-to-one match for each feature. The problem with this low-level approach is that it is very narrowly targeted, so any differences arising between features in each system either require user input or are resolved through the use of surface patches. The work done in this research involves a more generalized approach, which would not require the constant upkeep and human involvement of the more ad hoc approach of the commercial translation companies.

The approach proposed in this thesis uses a shared base ontology to represent features across the CAD domain. It differs from other ontology based approaches in that it addresses the conceptual definition of features by correlating the types of input parameters used to define a specific feature to the expected output. This is achieved by representing each feature as an instance of an ontology class using a three-branch CAD feature model. The feature tree of a given CAD model is saved as a sequence of instance-level representations of a feature class. Each instance of the feature class stores all of the input parameters used by the CAD system in the definition of the feature as instances of parameter classes defined in the shared base ontology. Additionally, each instance of a feature also includes information on the changes made by the feature to the overall geometry of the model, thereby full encapsulating all of the data that defines a feature and the results of its application. This approach exports the entire feature tree of a model from the source system as instances of the shared CAD ontology, which will act as a neutral exchange format. The data is imported into the target system by translating each feature and recreating the feature tree. The translation process uses ontology reasoning tools to first determine which feature classes in the target system are capable of reproducing each source feature, and then uses a dynamic mapping process to translate the source feature onto the correct target feature class. Once a suitable match is found, the target system uses the source feature data to create an instance of the corresponding feature in the target system. The resulting geometry from each feature operation is then compared to that stored in the file, to verify the translation.

Research Contributions

This research intends to create a robust method for the interoperability of CAD systems which will allow lossless storing of feature data. Most work in the field of CAD interoperability has been in the area of geometry recreation, with some emphasis in constraint and parameter exchange. The approach presented implements ontologies in a

new way, by attempting to fully store feature data in terms of the geometric creation concept it represents. It differs from other research using ontologies by attempting to characterize features as operations that correlate specific types of input to specific types of output, as opposed to simply defining them as classes that require a certain set of input types. Other research in using ontologies to represent CAD data focus more on semantic comparisons and domain specific class definitions, which would require all CAD data to conform to a predefined standard. By using sets of rules relating input to expected output, this approach aims to allow a reasoner to make inferences on which features are suitable matches before relying on semantic data similarity. This additional inferencing ability should allow for improved feature mapping and reduce the amount of human input needed. This approach would require a less restricting standard data format, as it only prescribes how feature data should be stored instead of mandating a set of standardized features. This research aims to represent data exactly as intended in the source system using a set of very basic ontology classes that represent concepts shared among the entire CAD domain, so data loss would only occur when the target system cannot support specific feature data.

In this thesis, Chapter 2 will present approaches taken by others and the background necessary to understand this approach. Chapter 3 will discuss the motivation to this approach and present a general overview. Chapter 4 will describe the three-branch CAD feature model in further detail. Chapter 5 will discuss the use of Ontology Web Language (OWL) to construct the shared base ontology. Chapter 6 will describe how features are classified in a local ontology using rules based on feature conceptualization. Chapter 7 will show implementation and proof of concept. Chapter 8 will conclude with a summary and discussions of future work.

CHAPTER 2

BACKGROUND

Feature-Based Design

In order to understand the importance of exchanging parametric feature data between CAD systems, it is first necessary to define a feature within the context of CAD systems and understand why feature-based design has become the dominant method of constructing solid geometry in the CAD domain. Shah and Mäntylä [1] describe features as "modeling entities that allow commonly used shapes to be characterized and associated with a set of attributes relevant to an application". Within the context of this thesis, the discussion of features will be limited to form features, which are used to describe portions of a part's geometry, as they are the tools used to construct the solid geometry in feature-based design. As the user of a CAD system creates a part, the features used are stored in a feature history tree, which acts as an instruction manual or recipe for how the part is constructed. The features used and the order in which they are added reflect the design intent of the user, and in a well designed part, changes to a step in the feature history tree can be made such that the changes carry through the subsequent steps of the model and a new part can be regenerated.

The reason why exchanging parametric feature data is so important ties directly with why feature-based design was developed in the first place. The earliest CAD systems were designed for simple drafting, showing objects using two-dimensional graphic models consisting of graphical primitives such as lines, arcs, and conics. This eventually evolved into three-dimensional graphical models, where graphical primitives were defined in three-dimensional space to create wireframe models. These wireframe models were difficult to create, the lack of surface information often made some

geometry ambiguous, and collections of graphical primitives did not always correspond to well-defined, realizable solids. A method to store such geometry as a realizable solid model that defined an actual volume was deemed necessary. During the 1970s, two main schools of thought concerning solid modeling were developed. Ian Braid and his colleagues at the University of Cambridge developed boundary representation for CAD systems [2], which represents solid objects as a collection of surfaces that bound a volume. The topic of boundary representation will be discussed in more detail later in this chapter. Voelcker and Requicha at the University of Rochester introduced Constructive Solid Geometry (CSG) models [3,4], which represented solid objects by the 3D space they occupied, describing shapes using mathematical expressions to determine if any point is space is internal, external, or on the boundary of the solid model. Solid models were created using CSG by using Boolean operators (union, intersection, and subtraction) to combine shape primitives into complex solid objects. Both systems had the advantage of creating only realizable solids, but each had limitations. Boundary representation had an advantage in that parametric surfaces and curves could uniquely describe an object, but in practice creating a model by defining the bounding surfaces was difficult, prone to errors, and hard to modify. CSG was much easier to work with, because solids were represented as simple combinations of shape primitives, which were easy to define and guaranteed the solid model was realizable provided the primitives were as well. CSG models could be represented with simple binary tree, with leaf nodes for the shape primitives and branch nodes for Boolean operations. However, CSG was limited to only reproducing shapes based on available primitives, did not uniquely describe a shape, and created an unevaluated model, meaning it must be checked with a boundary evaluation routine to determine information about vertices, edges, and faces. Both boundary representation and CSG had the disadvantage of only being suitable for final design, as there was no method to convey design intent or make changes quickly.

Feature-based design innovated the CAD industry by expressing solid geometry in terms of combinations of abstract features, which represented general shaping operations instead of the defined shapes of CSG. A form features can be described as a portion of nominal geometry, or a recurring, stereotypical shape [1]. Each feature is associated with several properties, such as generic shape, dimension parameters, constraint parameters and relations, location method and parameters, orientation method and parameters, recognition algorithm, inheritance and validation rules, and various other properties. The concept of a feature conveys generic shape, behavior, and engineering significance, but is not fully defined until a specific set of properties is defined. Extensive work with features was done in the late 1980s by the Shah group [5,6] and Dixon group [7,8]. When using a feature-based modeler, a library of generic feature classes is provided, and the user constructs a part by creating specific instances of these classes and combining them through Boolean operations. This has advantages over geometric modeling because it adds a level of abstraction to the design. Features combinations describe a general shape, while the specific property values of the instance define the dimensions. Values of the feature parameters can be changed and propagated throughout the history tree, meaning feature-based design allows for easy modification of designs and construction of part families. Additionally, features contain validation rules to ensure they are generating valid shapes when first created or when the history tree is modified.

Most commonly, solid object features are created from 2D sketches through extrude, revolve, or sweep commands. This is extremely useful because the sketch-based interface allowed users to describe shapes using constraint and dimension annotations, which were then used to automatically generate a sketch by solving the geometric constraints. The libraries of feature-based modelers also contain common part features of engineering significance, such as holes, ribs, edge rounds and chamfers, face drafts, shells, etc. In feature-based design, like CSG, it is possible to create the same shape using different combinations of features, so the choice of feature is often heavily dependent on

design intent and how the part may be modified in the future. The added functionality of feature-based design is what makes exchange of such data a significant research area.

The important thing to note is that there is no definitive set of features. A feature can be any operation to create a shape of some engineering significance. This is where difficulties arise in data exchange, because feature libraries provided by one CAD system do not match those of another. In general, there are a set of common features that are shared amongst different CAD systems, but even when they convey the same concept, there is no guarantee that the properties that define the feature are the same. Systems with different applications may have different features tailored to specific types of design, so some CAD systems may have features not supported by others. This lack of a unified structure is what makes feature-based data exchange so difficult.

CAD Feature-Based Data Exchange

In order to better understand the approach to represent CAD feature data proposed in this dissertation, it is important to examine the work done by others in the past. The two areas of research that tie directly into exchanging feature data are feature recognition and feature mapping. Feature recognition generally describes the determination of form features from geometric structures, while feature mapping refers to converting feature data from one application to another. The idea of feature recognition by discovering topological and geometric patterns was introduced by Kyprianou in 1980 [9]. Various methods of automatic feature recognition have been developed since the advent of feature-based design. Most often, these approaches are meant to take data from design programs to process planning programs, and a great deal of research has been done to determine machining features from a geometric model. Due to the sheer number of different approaches, it is far more practical to present survey papers on the subject of feature recognition than to list all relevant work. Particularly thorough reviews have been written by Shah et al. in 2001 [10] and more recently by Babic et al. in 2008 [11]. As

originally presented, such methods of feature recognition are not particularly applicable to this line of research because they assume a geometric model with no existing feature information. Under such approaches, there is no way to guarantee that the features generated match those of the original design. In feature-based exchange, the goal is to take existing feature data from a history tree and translate it to be compatible with a different system. However, by using the geometry created by each individual feature being translated, such recognition techniques can be applied to determine which features in the target CAD system may be compatible. Of particular interest is work done by Henderson and Anderson [12] and Prabhakar and Henderson [13], which used rule-based recognition of features and made use of Prolog, the logic programming language. This thesis proposes a similar rule-based approach that is meant to match specific feature types to one another instead of trying to recognize features from an arbitrary geometric model. Because this approach takes feature-based models as the input, it is more appropriate to review feature mapping approaches.

Historically, feature mapping has been researched to translate from one domain to another, most commonly converting design features to manufacturing and process planning features. Only recently, as feature-based modeling become dominant, has feature mapping been applied to convert from one CAD system to another. Early work by Shah [14] created a framework for understanding feature mapping which proposed features as analogous to vectors, describing a feature space as a domain defined by product type, application, and level of abstraction. A feature space that contained all feature types for all product life-cycle activities could be defined, and any feature would be considered a subspace. Features could be transformed between subspaces as long as the subspaces overlapped. Other early work often involved methods that attempted to standardize feature types for ease in exchanging data between different application domains. Bettig and Shah [15] proposed a standard set of geometric constraints for parametric modeling and data exchange. These geometric constraints were classified as

algebraic, logical, or dimensional. Ovtcharova et al. [16] discussed a need for clarifying the definition and classification of features. They focused on form features as fundamental feature types, proposing a classification schema based on complexity. They also proposed that feature-based design be considered a process with multiple levels linked by mappings. These mappings would relate application features, form feature definitions, form feature representations, and geometric models. Other work on feature conversion methodology was proposed by Rosen and Dixon [17] and involved a three-step process consisting of filtration, selection and aggregation. This process would first filter out irrelevant information, then relationships among design features were computed in the selection stage, and the aggregation would identify secondary features from the filtered design features, geometry, and computed relationships. Rosen and Peters [18] investigated applying mathematical concepts to product representation conversion, concluding that conversions without knowledge of the target application can be difficult and sensitive to small changes. Bettig, Summers, and Shah [19,20] discussed the use of design exemplars, a pattern of topological, geometric, algebraic, and semantic relationships with high level engineering significance that go beyond the capabilities of features. With these exemplars, part of the pattern corresponds to actual geometry, while part of it is inferred. These exemplars also included a second pattern for the false condition for use in re-write rules. Exemplars go beyond features by incorporating information from multiple design domains, attempting to unify models to avoid feature mapping and redesign.

Editable Representation (E-Rep) [21,22] was an early attempt at exchange of construction information. It specified models as a sequence of feature insertion, modification, and deletion processes. Project ENGEN (Enabling Next GENeration design) [23] involved extending the STEP standard to more than purely geometric data. The focus of project ENGEN was the exchange of geometric constraints which convey design intent, and demonstrated the exchange of 2D data containing constraint

information. Others have sought to add constraint data to the STEP standard. Kim et al. [24] focused on using recent enhancements to STEP standard to exchange construction history shape models with parameterization and constraints. They note a common problem that most researchers in this field experience when attempting to create exchange programs using a CAD systems application programming interface (API), stating "the APIs of commercial CAD systems are not primarily intended as an interface for model exchange" and indicate that future research should "adopt a ontological approach for the semantic mapping of modeling elements between CAD systems", basing their methodology on work by Patil et al. [25]. Rappoport et al. [26,27] describe a representation of features using a B-Rep structure. They use a concept called "feature rewrite", which computes the changes in geometry before and after a feature operation. Their research focused on the retention of geometric information and is being implemented in the commercial CAD translator offered by Proficiency Inc.

At KAIST (Korea Advanced Institute of Science and Technology), the approach being taken involves capturing the construction history by means of the journal file created by the CAD system [28-30]. This journal file contains a record of the commands utilized by the user, which it then converts into a non-STEP neutral format. This approach to interoperability is achieved by converting the construction history into instruction information for the target system, instead of exchanging actual model data. More recently, the neutral macro format has been updated to work with geometry-based data to avoid problems with part references based on creation order [31], but is now facing problems with persistent naming. Li et al. [32] established a real-time collaborative design environment based on use of neutral modeling commands. APIs of the source and target systems were used to exchange construction information across a network in real-time through use of neutral commands. Research into translating feature data across heterogeneous systems using XML files and the API of various CAD systems has been done recently at Wuhan University in China [33-35], although it appears that

feature mapping is done manually. Similar work with neutral XML files and feature mapping has also been done recently at Dalian University of Technology in China [36,37].

The use of ontologies has become increasing favored in approaches aimed at exchange of semantic feature data. Ontologies are used as a way to create a consistent vocabulary of concepts across a domain, as will be discussed in further detail in the next section of this chapter. A common language is essential to a neutral format, which reduces the number of translators and helps resolve semantic differences. Some ontology languages have also been created to support inferencing, which is a highly useful tool when working with heterogeneous data sets. Dartigues et al. [38] exchanged data between a CAD ontology and a computer aided process planning (CAPP) ontology through use of a common domain ontology. However, the CAD ontology was only used to store geometric data and not construction history or parameters. Seo et al. [30] added semantic data to the macro-parametric approach through an ontology using the F-Logic format. The VRCIM laboratory at Washington State University is also active in ontology-based interoperability. They have illustrated interoperability between product design and assembly simulation domains [39,40], with further discussion of how to translate between different domains through use of a bridge ontology, but not how to translate in cases besides a one-to-one match [41].

The approach to improve interoperability proposed in this research differs from the other ontology based approaches described by examining the individual features from a more conceptual viewpoint, relying more on describing features as interactions between basic parameters and the types of geometry that result instead of rigorously defining features in a shared domain ontology or separate ontologies which then must be mapped. This approach builds off of the automated feature mapping approach proposed in previous work by our research group [42,43] addressing some of the main issues encountered with that method. It uses techniques similar to the rule-based feature

recognition proposed by Henderson et al. [12,13], but instead of applying them to a general geometric model from one system to find instances of features in another system, this approach asserts that because the features from the source system are already defined, it is more appropriate to use such recognition techniques on each individual feature shape as initially defined to preserve design intent. Converting the features in this way would not normally be done, because research in this area is mostly concerned with mapping to different application domains, and feature in one domain may not correlate to a feature in another. For example, a design form feature may not be compatible with a machining feature. However, because this research is concerned with exchanging data to different software within the same domain, maintaining the features as they were is essential to maintaining design intent. The reasoning behind using a feature recognition technique is to ensure the shape of the feature is recognized in the target system's feature library before any automated feature mapping based on semantic data occurs. This is an important distinction from other automatic mapping approaches, which rely solely upon semantic data, because it can accurately identify a feature instance as an object of a particular feature class even when there are semantic differences. This is an important characteristic because features from heterogeneous systems do not always have one-to-one matches, even when they represent the same concept. In cases such as this, automatic semantic approaches will conclude that there is no match, which may not be true. Additionally, such an approach also immediately rules out all features which are not matches, which illuminates unnecessary similarity calculations and semantic comparisons. Further discussion of these differences will be discussed in the next chapter.

Use of Ontology for Data Representation

In order to understand the feature model proposed, as well as how it addresses the limitations of other research, one must first understand what an ontology is and why it is beneficial to store feature data using a common ontology. In computer science, an

ontology is a formal representation of a set of concepts and the relationships between those concepts within a given domain. It is basically a way to create a common vocabulary to model different objects or ideas, their properties, and their relationships. The concept of an ontology originates from philosophical study, where it is concerned with describing entities that exist, and how entities can be grouped, classified into hierarchies, and organized according to similarities and differences between concepts. Ontologies are used by people in a number of fields in order to better organize knowledge into taxonomies. Ontologies have become very important in many fields of computer science as a form of knowledge representation because they characterize concepts in a consistent way, such that a computer program can make inferences. As an example to illustrate this, consider describing a car as being "red". A computer program would have a hard time understanding that information because it lacks the ability to infer relationships that humans do. However, if I were to describe "red" as a "paint color", and "paint color" as an "appearance property" of the car, the program would be able to make several inferences about the car because the data is now less ambiguous. Rules can be applied so that a car can only have one color, and so that only certain colors are valid on certain cars. By using a common ontology to represent CAD data, we can give a computer the ability to compare features using tools developed in computer science.

To describe why an ontology-based approach is being taken by our research group, it its easiest to compare the process of feature mapping to the process a human would taken when manually recreating a model in the target system. Manually recreating a model one feature at a time in the target system is by far the most reliable way to translate the data with as little data loss as possible. A person with knowledge of both systems can easily identify which features, concepts, and parameters are equivalent to each other, and can simply copy the data accordingly. Knowledge of both systems is analogous to the more ad hoc approaches taken by commercial translation solutions. However, a more general approach will have to assume no specific knowledge of the

source system. Imagine if a person needed to manually recreate a part from a CAD system they have never used before. If they are not familiar with either system, they might look for features with similar pictures or names that are equivalent, such as extrude in one system being called extrusion in another. However, if the names do not match and there are no pictures, a person could try features that use the same types and numbers of attributes and use the results displayed on the screen to determine the equivalent feature through trial and error. A person with extensive knowledge of the target system would likely be able to discern which features are needed based only on recognizing the shape of the feature, and then from there try to discern which parameter values need to be matched based on name or datatype to create the same geometry. However, without a common ontology to describe the concepts and a way to verify results, a computer would not be capable of any of these reasoning techniques. By using an ontology, the meaning of concepts is more flexible but still unambiguous.

When an ontology is used to represent data, it can be useful to visualize the data in a graph, with nodes representing concepts and properties, and lines connecting the nodes representing relationships. By comparing a feature graph in one system to that in another, the computer can determine the similarity between features. If the library (list of all features) of two different CAD systems were represented through a common ontology, features could be mapped between systems automatically, allowing for easy determination of features with one-to-one matches, identification of features that are most similar to each other in cases without a direct match, and identification of features with no equivalent feature in the target system. Information not directly related to the feature definition, such as constraints, tolerances, surface finish, material data, and other markups could also be stored in an ontology representation. The biggest benefit of the ontological representation is that it only specifies a common language set, allowing features to retain the information that makes each unique and limiting the loss of information relaying design intent to only that which is not supported by the target system. In addition, there is

a set of concepts that is fairly universal between CAD systems, meaning a common expressive data structure should be understandable to anyone familiar with CAD systems. From an implementation standpoint, the process of exporting files into this format should also be fairly simple from a CAD company's standpoint, because it is only an organization of the data used to define a feature, as no computation should be necessary.

The Hybrid Semantic Feature Model and Feature Mapping

The hybrid semantic feature model is the initial ontology representation of CAD feature data proposed in our group's previous research [42,43]. This thesis describes an approach which was heavily influenced by this previous approach, so it is helpful to understand the basics. The hybrid semantic feature model focused on storing the semantic data used to define a feature and using similarity calculations to compare the features from one system to another and automatically map the best matches to each other. It represented a feature with a directed, labeled, and attributed graph. The model was associated with eight types of attributes, which were classified as individual, interfacial, or alias. Individual attributes were used to characterize the attributes that only belong to one feature, including Parameter, Sketch, and BooleanSign. Interfacial attributes are supplied to define the boundaries of features that belong to more than one feature. Four interfacial attributes are Point, Line, Surface, and SolidBody. The alias attribute is used to capture the possible alias name of a feature, either from different systems or from different naming methods. For example, an extrusion of a fixed cross-sectional profile is referred to as Extruded Boss/Base in SolidWorks software and Extrude in its API. But the same feature is referred as Protrusion in SolidEdge. Thus Extruded Boss/Base and Protrusion are the aliases of the same feature in different systems. The alias attribute was used to facilitate the mapping process by storing the multiple names for the same feature

after mapping had successfully been achieved. An example feature graph is displayed in Figure 1.

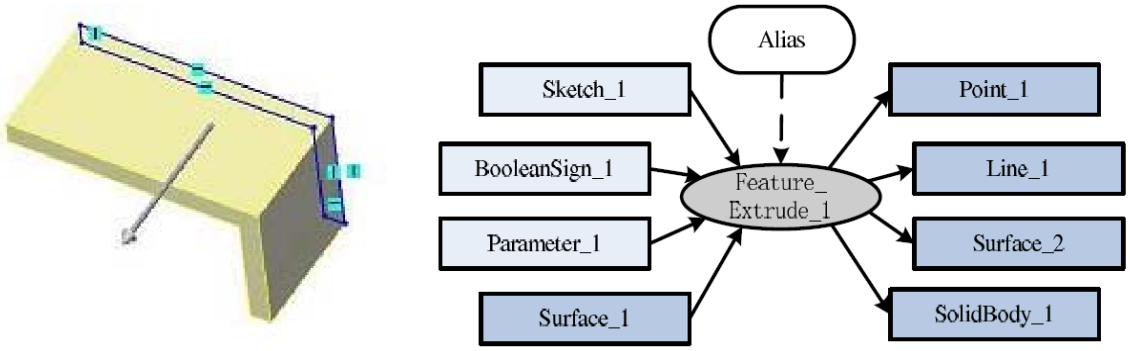

Figure 1: Feature Graph example of SolidWorks Extrude Feature [43]

Interoperability between programming languages is an important issue in computer science research, so treating CAD features as language types and applying a systematic approach to map features semantics based on computer science approaches was used to automatically map features. Features described using the hybrid semantic model were able to be automatically mapped based on similarity calculations and type checking. In the process of feature mapping, one feature in the source system is compared to the features in the target system. A feature mapping algorithm can compare the graph structures and calculate the semantic similarity between the source and target feature. Similarity is calculated for both the source and target features by dividing the total number of common attributes by the total number attributes in each respective features definition. This led to several cases, each with their own conclusion, as illustrated below.

Case 1: Both source and target similarity calculations equal 1. This indicates that all the attributes of both the source and target features have a match, and that the features are equivalent. This is the ideal scenario.

Case 2: Both source and target similarity calculations are less than 1. This indicates that both the source and target features have attributes that do not match each other, and are therefore not equivalent. This does not necessarily mean that the features cannot represent the same geometric construction, but simply that their attributes may be defined differently. It may be possible to resolve the differences through some conversion process and create an identical feature.

Case 3: Source similarity calculations equal 1, but target similarity is less than 1. This indicates that the source feature is an equivalent subset of the target feature and that the target requires additional attributes to be fully defined. This can mean that these additional attributes may need to be computed or that the target feature may offer an additional attribute that is not supported by the source feature. In such a case, it may be necessary to include default values in the definition of the target feature for attributes that may not be used in other systems.

Case 4: Target similarity calculations equal 1, but source similarity is less than 1. This indicates that the target feature is an equivalent subset of the source feature and that target feature does not require or support all of the attributes required in the source feature definition. This can indicate that there are redundant attributes in the source feature or, more problematically, that the target feature may be lacking a parameter needed to fully reproduce the source feature.

In the above, only the first case demonstrates an adequate translation. The remaining three cases present non-trivial problems that must be handled to ensure interoperability. These three cases are indicative of the inherent differences between CAD systems, which can signify features that cannot be replicated in the target system or

require some calculation or conversion. Unfortunately, the hybrid semantic feature model lacked any means to automatically handle any issues arising from the three non-trivial cases. In cases where similarity did not equal 1, it would pick the feature pair with the highest similarity or defer the choice to the user. It also had no means of storing the resulting geometry of the feature operation, meaning there was no way to verify that the feature translation successfully recreated the original geometry. Without a means to automatically handle cases of dissimilarity or to verify the results, the computationally expensive mapping processes were limited in usefulness.

It is also important to note that there are two different mapping processes, each with their own advantages and disadvantages. One method involves the static mapping of 2 class-level libraries to create a translator for one-to-one matches. This translator could then be used to directly translate an instance-level file from the source system to the target system. This approach can be seen in Figure 2.

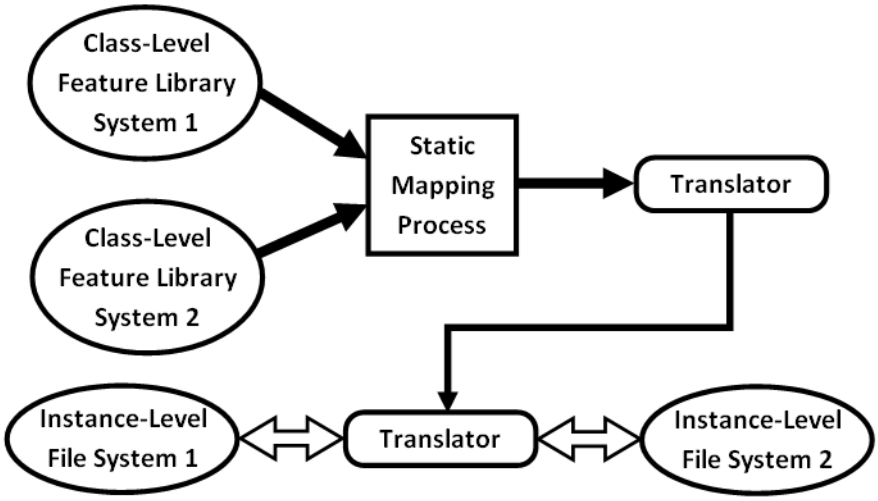

Figure 2: Static Mapping Approach

The other approach is a dynamic mapping process, where an instance-level file from the source system would be compared to the class-level library of the target system,

with an instance-level file being generated in the target system once the match is found. This approach can be seen in Figure 3.

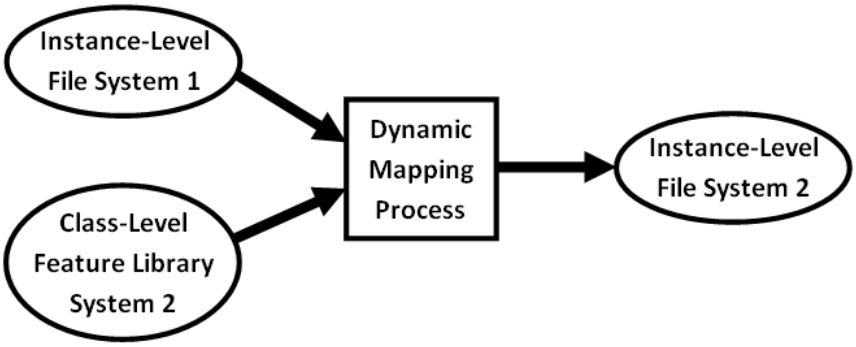

Figure 3: Dynamic Mapping Approach

Of the two, the static mapping approach is the most efficient. In static mapping, the similarity calculations would only have to be done once, the features could be stored as matching pairs, and the resultant translator could then directly convert from one instance-level representation into another without any calculations. The dynamic mapping would be inherently less efficient because similarity calculations would have to be made every time a file was translated, unless some method to store and distribute the match was implemented. However, even in this case, a full translator would not be achieved until every possible combination of a class from the target system was created and mapped at the instance-level, which is somewhat comparable to a static mapping process. Finally, for full two-way exchange of CAD data, both system libraries would be necessary, which would be all that is required for static mapping. However, there are major obstacles with static mapping that makes the use of a dynamic mapping approach far more practical.

The first major obstacle in static mapping is that it requires full access to both of the systems' class-level libraries and they must be kept up to date. This may be very

difficult task, as it would likely require reverse engineering the entire library, because CAD companies are not always forthcoming when it pertains to their proprietary data. The other major obstacle is the lack of a practical method to verify that each match between the source and target feature classes is valid. Unless there is a perfect one-to-one match, there is absolutely no guarantee that the use of the class pair with the highest similarity will result in a correct recreation of the feature. This is because the resulting geometry of a feature is so dependent on the prior geometry and the input parameters that there is no way to describe it fully in a general, class-level sense. Therefore, any static mapping approach would require the two class-level libraries to be very similar to each other, because without any verification tools, the only way to ensure a match would be human involvement, which offers few benefits over the existing ad hoc approaches. Fortunately, with the use of a dynamic mapping approach, it is possible to implement a verification process. When working with an instance-level source file which is converted into an instance-level target file, it would be possible to verify the geometry is recreated so long as the original geometry data is stored in the source file. Additionally, because dynamic mapping only requires the library of the target system, it works better when the goal is a neutral exchange format, as the dynamic mapping process could be integrated into the file import process of each CAD system. A developer working for a specific CAD company would have a much easier time getting access to or constructing a class-level library for their particular platform, and because the dynamic mapping process needs no knowledge of the source system library, they do not need access to any other companies libraries.

Boundary Representation and Euler Operations

This section describes the basics of boundary representation (B-Rep) in CAD systems, which are currently the standard of exchange and would be necessary in implementing any geometry verification scheme. In general, when one exports a file to be

opened by a different CAD system, they are exporting some type of B-Rep model. A B-Rep model is very useful as a medium of exchange when one is only concerned with the final geometry of the part, because regardless of the differences in feature types between different CAD systems, any valid part should be able to be expressed in terms of the volume enclosed by a bounding shell. Several neutral formats were developed for the express purpose of creating an industry standards for exchange and are supported by every CAD system. However, in modern CAD systems, features are the main tools of geometric construction and generally are what convey the intent of the designer. Due to the prevalence of feature-based CAD design, B-Rep is an insufficient tool for CAD interoperability in cases where modifications to the file may need to be made. However, it is still an essential tool for error checking when dealing with feature-based exchange because the resulting geometry must be the same (within reasonable error) to consider the translation correct.

B-Rep is essentially representing a model in terms of its bounding surface or shell, such that only the volume on one side of the shell is considered part of the model. This bounding surface can be divided up into individual faces. Each face is described as a surface bounded by a loop of edges. Each edge acts as the intersection between 2 adjacent faces. Each edge is bounded by 2 vertices. The data structure is distinguished into two groups called the topology and the geometry. The topology serves to describe the structure of the model, while the geometry describes the shape. The topology only defines the structure of the model by describing which vertices are used to define the end points of each edge, which loop of edges defines the face, and which faces enclose a shell. Take a simple cube for example (Figure 4). The topology describes the 6 faces that enclose the shell of the cube, the 12 edges that bound those faces, and the 8 vertices that bound the edges.

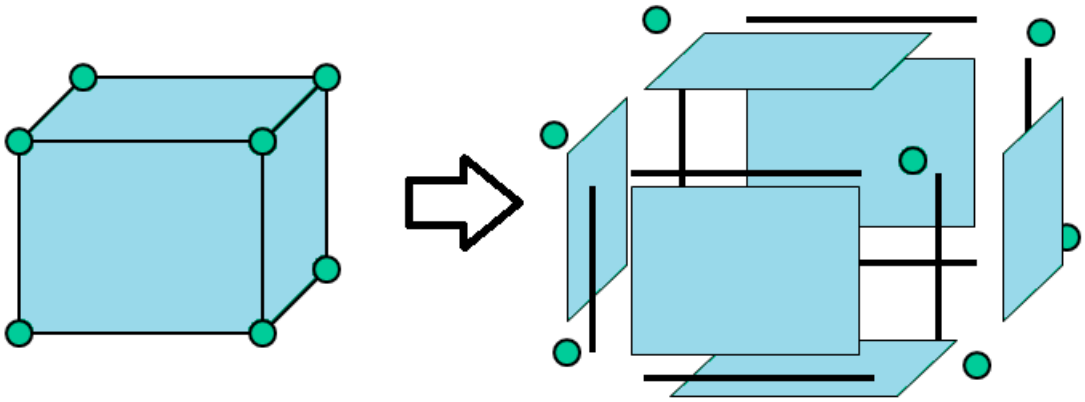

Figure 4: Cube Topology

Each element of the topology has a corresponding element of the geometry that is used to describe it. Each vertex corresponds with a point, which gives its coordinates. Each edge corresponds with a geometric curve, which describes its shape in 3D space. In the cube example, each edge is defined by a geometric line, and the two vertices are used to define the endpoints on that line to create a line segment. Each face is associated with a geometric surface, with the loop of edges defining the boundaries of the surface and defining the face. In the cube example, each geometric surface is a plane, but a surface can be described by non-planar entities as well. A diagram of the basic B-Rep data structure is displayed in Figure 5.

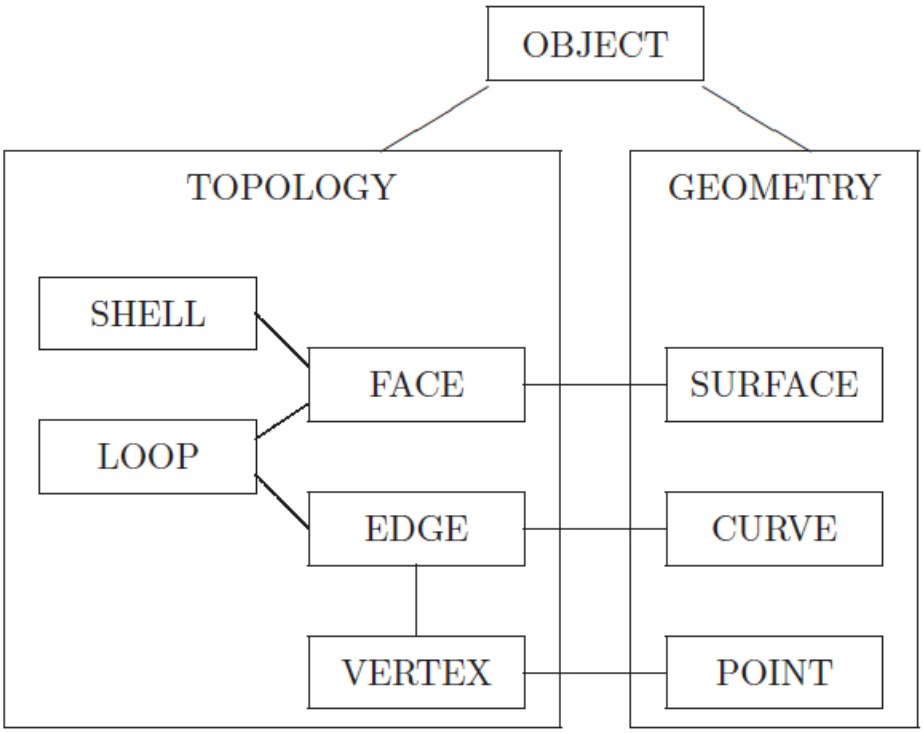

Figure 5: B-Rep Data Structure [44]

Some data structures also chose to represent edges as a pair of half-edges. This is because every edge is shared by two faces, so one half-edge belongs to one surface and its compliment half-edge belongs to the other. This is the structure used by the Spatial Corporation's ACIS standard [45,46] and will be used in this thesis as well. ACIS refers to half-edges as coedges, and loops are composed of coedges. Each coedge has a direction, as well as a pointer to the next coedge and previous coedge in the loop. All coedges must point in the direction of the next coedge and point in the opposite direction as their compliment coedge. Coedges always traverse the bounding edges of a face in a counterclockwise manner when viewed from the external surface, as demonstrated in Figure 6.

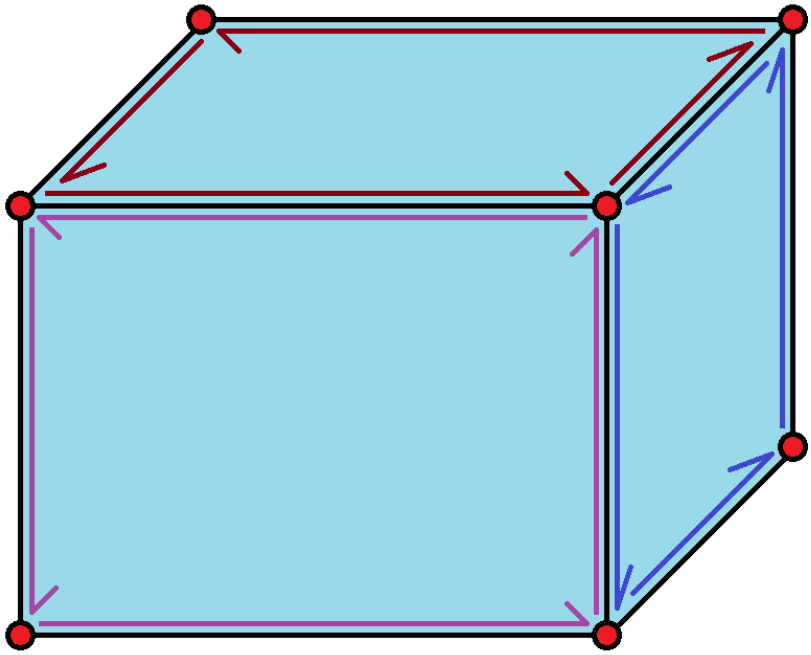

Figure 6: Illustration of Coedges

One important property of the standard B-Rep data structure is that any manifold object can be described as an Eulerian object. What this means is that the number of elements must follow a set of rules and that object can be created through combinations of Euler operators. To be a valid object, the set of variables representing the number of each element ("v" for vertices, "e" for edges, "f" for faces, "h" for hole-loops, "g" for genus, and "b" for shells or bodies) must follow these rules described by Braid et al. [47]:

1. $v, e, f, h, g, b >= 0$
2. $if\ v = e = f = h = 0,\ then\ h = b = 0$
3. $if\ b > 0\ then,\ v >= b\ and\ f >= b$
4. $v - e + f - h = 2(b - g)$

The first three rules are fairly obvious. Rule 1 states that negative numbers of elements are not possible, Rule 2 states that it is impossible to have a shell or genus

without any of the other elements, and Rule 3 states that each shell must have at least one vertex and one face. Rule 4 is the Euler-Poincaré formula that all valid manifold objects must follow. An Euler operator is any operation that changes the number of elements by at most 1 while still satisfying the Euler- Poincaré formula. For instance, the Euler operator MEV (make edge and vertex) or in array form (+1,+1,0,0,0,0) is valid because the net value of the left-hand side of the equation would not change. Likewise, a ME (make edge) operator, (+1,0,0,0,0,0) would not be valid because it would unbalance the equation. There are a total of 99 valid Euler operators, and any valid feature operation must be some linear combination of them. However, any of those 99 valid operators can be described as a linear combination of other valid Euler operators, meaning there is no unique combination for each feature operation. However, a useful property to consider is that the net sum of operators must satisfy the Euler-Poincaré formula, meaning that whatever changes a feature makes to the existing B-Rep must also satisfy the Euler-Poincaré formula.

The concept of an operation to describe the changes each feature operation makes to the B-Rep would be very useful in the ontology-based file format. By following the concept of "feature rewrite" as described by Rappoport et al. [26], one could store the changes to the B-Rep as the difference between the B-Rep entities before and after the feature operation. By examining the difference in B-Rep for each feature as opposed to the overall model geometry, it is possible to observe what each feature operation does given different inputs. This is an important component to how a feature is conceptualized. If it is assumed that the source file has a valid B-Rep and the target system is producing valid B-Rep data, then it can be assumed that the models are made using valid Euler operators without having to determine them. The best way to implement this is to record each instance of B-Rep that is deleted and each instance that is created. By separating the net change into individual deletion and creation operations for each type of entity, it is easier to determine what each feature is capable of doing and ensure that the feature in

the target system generates the same results. However, there is a major problem with this approach. There is a lack of a persistent naming scheme for B-Rep data. The labels assigned to each entity will not be the same between each system, meaning it would be difficult to compare them directly. Additionally, any B-Rep entity referenced by a feature in one system must be correlated to the equivalent entity of the target system for the data to be translated. This would require using the geometry data to uniquely identify each entity. However, persistent naming of B-Rep data is a complicated research topic by itself, and will not be addressed in this research.

With the background material addressed, the next chapter will describe the general approach taken by this research. It will discuss the motivation for a new ontology-based approach and give a general overview of how it works. It will describe the limitations of previous approaches and how this new approach attempts to address them.

CHAPTER 3

MOTIVATION AND OVERVIEW OF PROPOSED ONTOLOGY-BASED APPROACH

The main driving factor behind this research topic was determining a method which could address the shortcomings of other interoperability solutions. Wache et al. [48] explains that data interoperability problems that arise within a given domain are due to structural and semantic heterogeneity. Structural heterogeneity represents data incompatibility that occurs due to the data structures being different. Semantic heterogeneity represents the data incompatibility that occurs due to naming or terminology differences. The problem that most interoperability approaches focus on is mainly semantic heterogeneity because of their prevalence, but it is important to note that structural heterogeneity is a big problem in data interoperability that must be addressed. It is fairly easy to automatically map feature data from one system to another when equivalent features are structurally identical with some semantic differences through use of ontologies. The real challenge is determining how to automatically map features that are equivalent but are defined with a different data structure, a task which is usually delegated to a human user in interoperability solutions. However, to address structural heterogeneity, semantic differences must first be resolved. By using ontologies, one can explicitly define a domain of concepts, thereby describing the specific semantics that must be used. But there are many different ways to represent data of a particular domain using ontologies, so it is important to determine which method is the best for the CAD domain.

Types of Ontology Representation

Wache et al. [48] describes three types of approaches through which ontologies can be used to integrate data and specify semantics. These three types of approaches are identified as single ontology, multiple ontology, and hybrid ontology approaches. Figure 7 illustrates the main information structures of these three approaches.

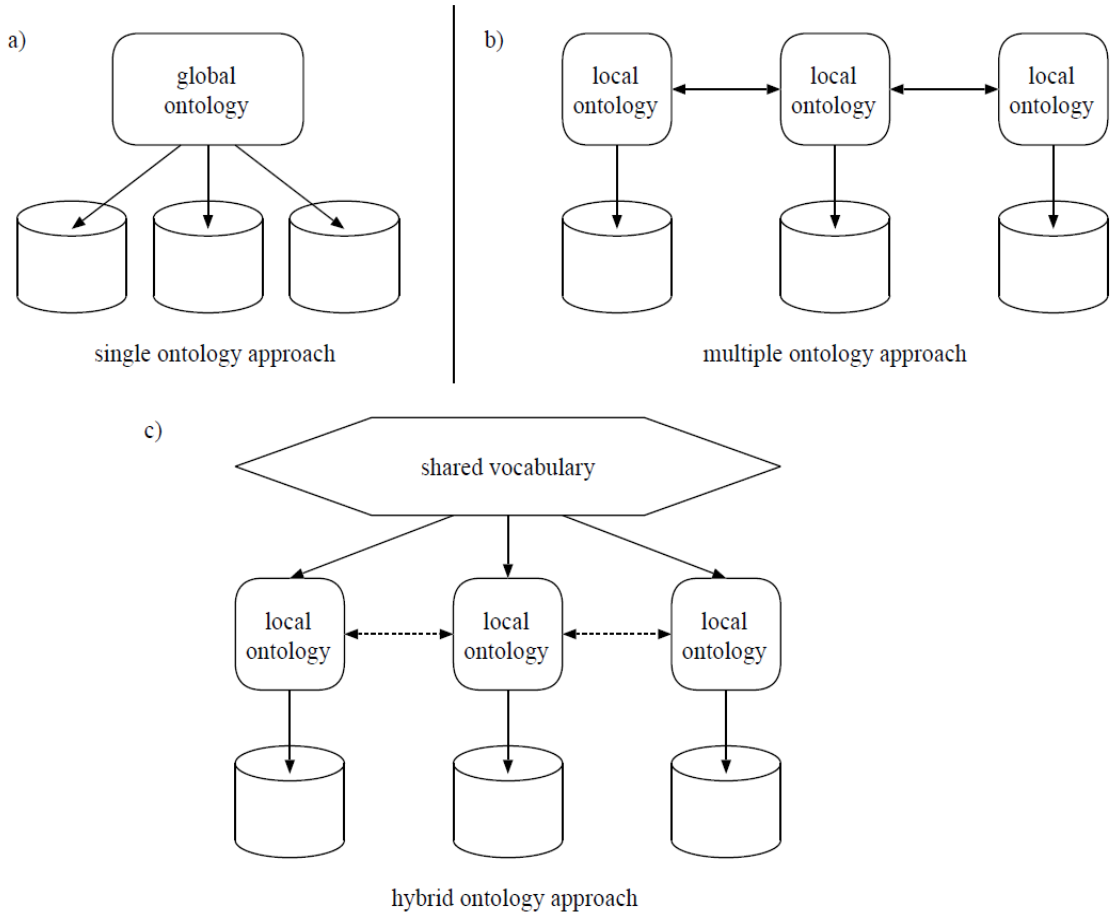

Figure 7: The Three Ontology Approaches [48]

Single ontology approaches (Figure 7a) specify a single global ontology for the entire domain. Each source of information, in this case each CAD program, is related to

the global ontology and their information can be stored using the global semantics. In this way, the global ontology is analogous to a neutral format, where all data must be stored in the prescribed format. Unfortunately, problems arise when different data sources have unique types of data, or different views or structures on the same concept. For example, a feature in SolidWorks may not exist in Pro/Engineer, or it may be a combination of multiple features, or it may exist but is defined differently. Creating such a global ontology would be difficult, as it would require each system to agree on a standard shared set of features, a universal way in which to define them, and as a result would greatly limit the amount of data which could be shared with this approach. Decisions on how the global ontology should structure feature classes, the minimum set of features that must be supported, how detailed each feature definition must be, etc. would be difficult to make, because there is no "right" way to define a feature, it is a matter of personal opinion. A simplified feature set would not permit the use of highly specialized advanced features, and a very robust feature set would be difficult to implement and result in a lot of concepts unique to specific CAD systems, defeating the purpose of a global ontology.

Multiple ontology approaches (Figure 7b) specify a separate ontology for each source of data within a domain; in this case each CAD system would have its own ontology with its own vocabulary. Multiple ontologies have the advantage of allowing each system to store data as they see fit, and does not require limiting the feature types. Unfortunately, the lack of a shared vocabulary among the ontologies means that mapping data between different systems is difficult, as they may be semantically and structurally different. Lack of a shared vocabulary also means that mapping is a direct and single-directional process, so full exchange between *N* systems would require *N(N-1)* mapping processes. As the number of systems increases, this becomes unmanageable, as these mapping processes are usually ad hoc. An example of this can be seen in Figure 8. This essentially represents the state of CAD system interoperability currently, as each CAD

system has its own internal data structure and usually a means to extract that data via the API, but differs too greatly from any other system to be exchanged.

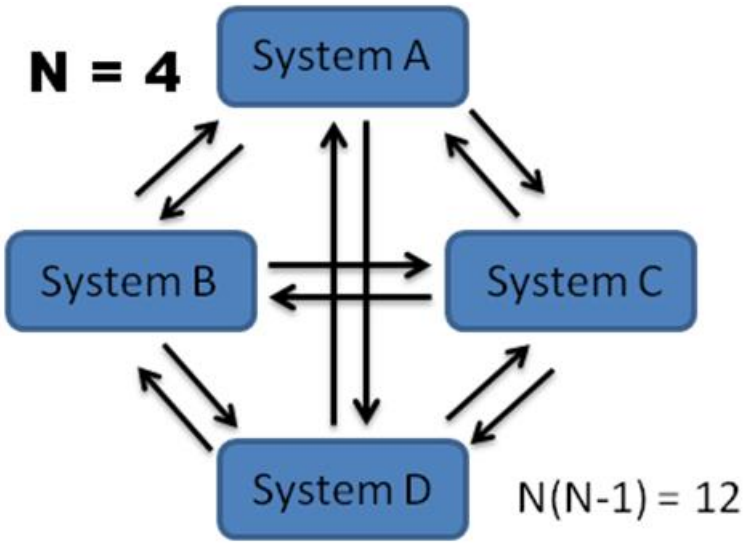

Figure 8: Direct Mapping Limitation

Hybrid ontology approaches (Figure 7c) resolves the issues of both single and multiple ontology approaches by combining them. A hybrid ontology approach is essentially a multiple ontology approach in which the multiple ontologies are built from a shared vocabulary. The shared vocabulary contains the basic concepts of the domain, which are combined in each local domain ontology to create more complex concepts. The shared vocabulary can also be represented with its own ontology to create a shared base ontology, which can represent specific semantics and data structures of the basic concepts. Using a shared base ontology allows information to be exchanged in a format that each local ontology can interpret, effectively creating a neutral exchange format without forcing the restrictions of a single ontology approach. Another benefit of the

hybrid ontology approach over multiple ontologies is that the shared language reduces the mapping processes of *N* systems from *N(N-1)* down to *2N*, as demonstrated in Figure 9.

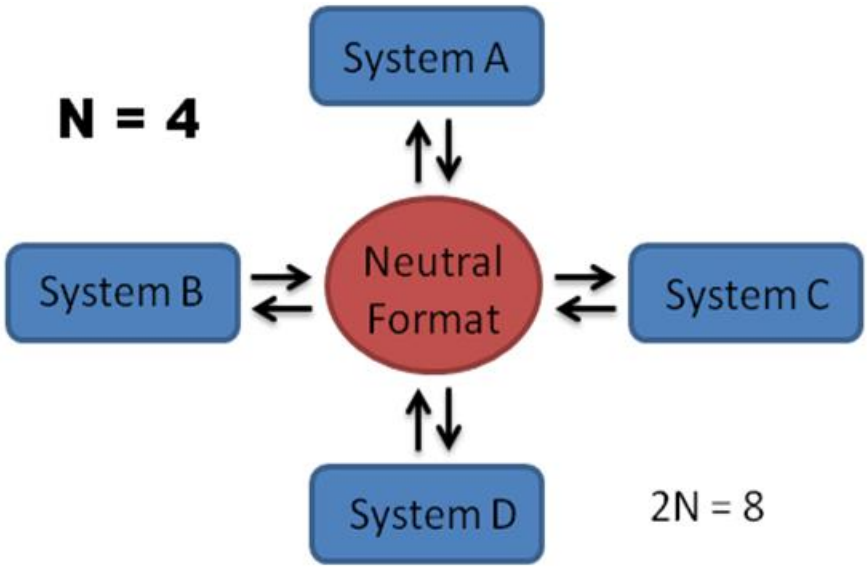

Figure 9: Mapping Advantages of a Neutral Format

Not only is the number of mapping processes reduced when dealing with more than three different systems, but the complexity of the mapping processes is greatly reduced as well. If any concept in a local ontology can be described as combinations of the basic concepts described in the shared base ontology format, then the mapping process from the local ontology is simply a matter of parsing the data into the shared base ontology, and the only mapping process that requires reasoning is converting from the base ontology to a specific local ontology. Additionally, as the shared base ontology becomes more detailed and thorough, the amount of semantic heterogeneity decreases significantly, and the cases that do exist are much easier to resolve. For that reason, the approach described in this thesis implements a hybrid ontology approach in which the shared base ontology is as robust as possible, containing detailed and structured

definitions of all the base concepts of the CAD domain, while the local ontologies only contain definitions of their feature operations as specific combinations of these base concepts. Additionally, features will be structured using a particular format, the three-branch CAD feature model, which is described in the following chapter. The three-branch CAD feature model prescribes how the base concepts should be used to define feature as combinations of reference attributes, parameter attributes, and B-Rep Operations. This eliminates most of the semantic heterogeneity problems, but there are still problems with approaches based solely on using semantic similarity to map data, as will be discussed in the next section.

Deficiencies of using Semantic Data to Conceptualize a Feature

The approach implemented with the hybrid semantic feature model [42,43] used a dynamic mapping procedure which compared a feature name to those of the target system library, when a match was found, it would map data to that definition, and when a match was not found, it would attempt to determine the best match by running similarity calculations that compared labels and attribute types. However, this approach required systems with high levels of semantic similarity, and problems could arise when dissimilar features had similar graph structures. Fundamentally, the approach was lacking because it was only defining features in terms of the parametric data that defined it, and not correlating that to how a feature uses the data to affect geometry. In general, most other ontology-based approaches that attempt to resolve data interoperability take a similar approach to resolving semantic heterogeneity. This can be highly problematic when dealing with high level concepts, as oversimplification often occurs when defining a concept as only a combination of certain properties. Problems can also occur when there is structural heterogeneity between equivalent concepts.

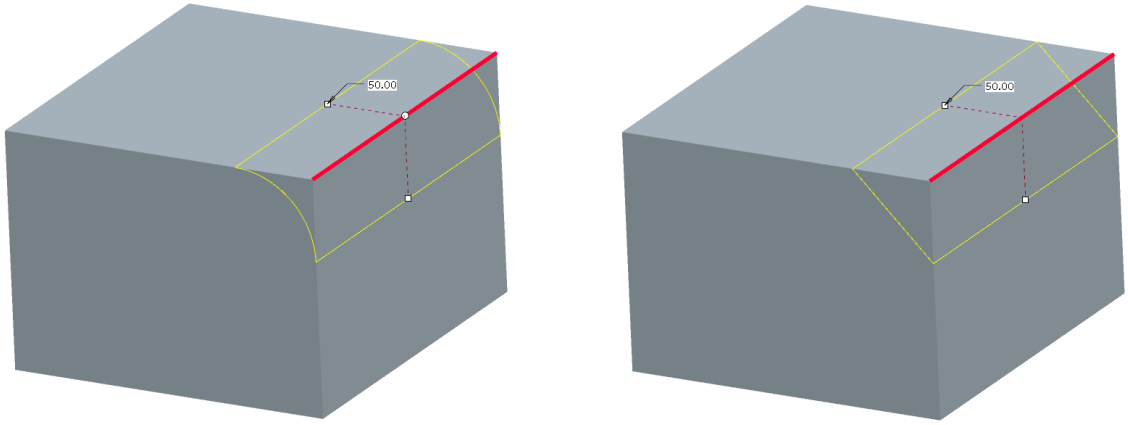

Figure 10: Round Feature vs. Chamfer Feature

A simple example of a semantic heterogeneity problem that could not be automatically resolved would be trying to map an edge fillet from one system to another. In some systems, a fillet feature is called a round, so if an alias had not previously been established, the semantic approach would have to resort to attribute type comparison. In the target system, a round and chamfer feature would have a very similar set of attributes, as demonstrated in Figure 10. In this example, both features take a float value to describe the dimension, and a single edge as a reference. The mapping process would be unable to determine which one is correct, because the feature graphs for both round and chamfer features would be a match for the fillet feature. Without existing knowledge that a fillet is analogous to a round in terminology, the mapping process would not be able to resolve this issue and would have to rely on user input. If the user was unfamiliar with the terminology, they would have to examine the resulting geometry of each to determine which was correct. This problem illustrates how semantic data is not fully encapsulating the concept of the feature.

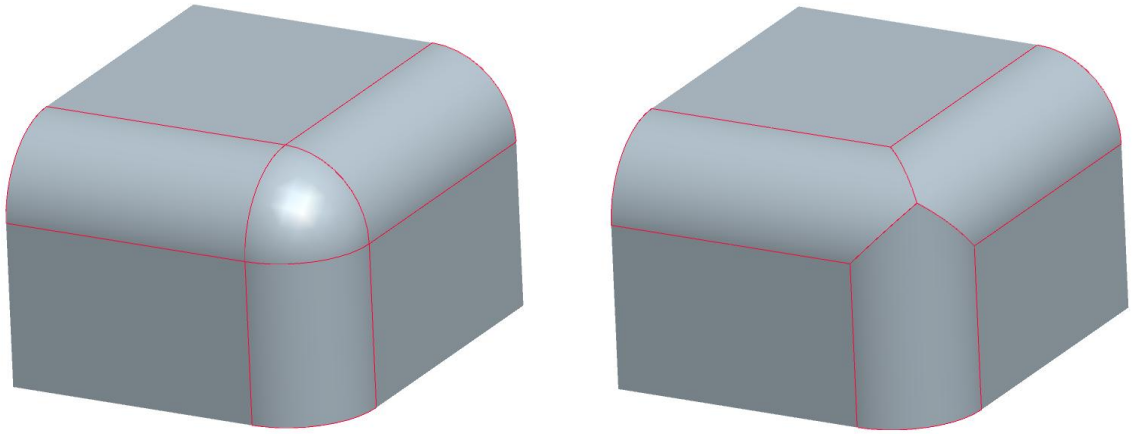

Figure 11: Round Intersection Geometry

Another simple problem that would be problematic involves two types of round features, as demonstrated in Figure 11. The difference may be as small as a single option within the feature definition that is not shared between different systems, introducing structural heterogeneity. Suppose there was a "blend intersection" attribute stored as a Boolean value in a feature in one system, while another system separated the concept into its own feature. Because the second system does not store this value, the mapping process becomes more difficult. When mapping to the feature with the additional Boolean value, the mapping process would have to prompt the user for a default value, being totally incapable of handling cases in which the number of attributes does not equate. Likewise, when mapping to the system that separates the concepts into two different features, the standard mapping process would calculate equal similarities, and not reason that the Boolean value corresponds to each. The system also had no way of verifying that the translated feature recreated the correct geometry, so the user would have to verify that manually. In both examples, once the correct map was established, the alias could be stored, but in cases of highly different systems, mapping would require a great deal of user input, which would negate the benefits of using ontologies over a more ad hoc

translator. Again, the problem can be traced back to the feature definitions not fully illustrating the concept of each feature.

Using Rules to Improve Feature Conceptualization

To overcome the problems with semantic and structure heterogeneity, it was necessary to examine the problem from more than just a semantic viewpoint. In order to reduce the amount of user input and better automate the mapping process, it was essential to relate the semantic input to expected geometric output. For sake of comparison, it is useful to consider how a human would manually translate a part from one system to another. The previous semantic mapping approach only replicated the process in which a human would search for a feature with a similar name, and then copy values of similar type and description from one system to another. Clearly, this is not sufficient, because the human would be able to observe and understand the changes each setting is making to the geometry, whereas a purely semantic approach does not. By adding B-Rep data into the ontology-based feature representation, it is possible to give the computer a basic means to verify geometry and enable feature mapping by process of elimination for a particular instance, but this is still not an efficient approach because it would require a large number of tests and feature creations to recreate a specific geometry. The ideal approach would emulate a human translator who conceptually understands features in the target system, allowing them to predict the geometry output without actually having to create it. The person might not know exactly how different attributes need to be mapped, but they could still reduce their search to only those features which they know can generate similar geometry.

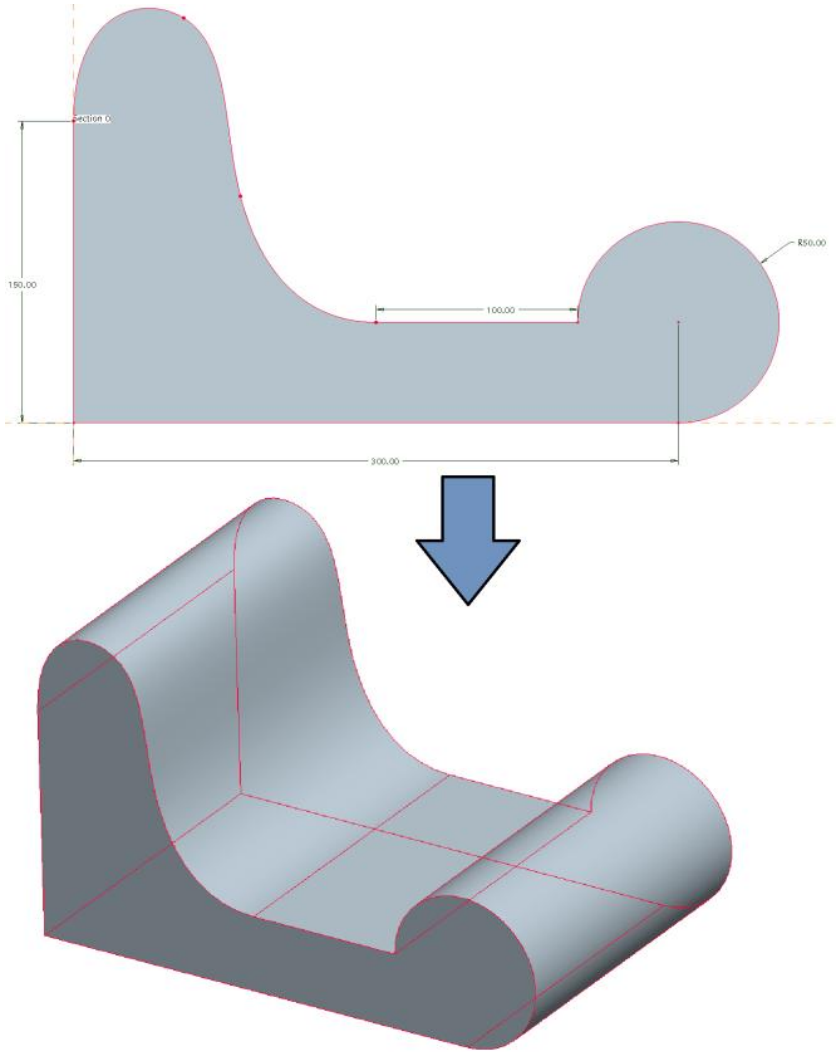

Figure 12: Extrusion Feature Conceptualization

The goal then is to conceptualize a feature in a way that an ontology reasoner could understand. But this is somewhat problematic, because the resulting geometry of a feature depends very heavily on the type of input parameters that define it. For an example, consider a simple solid extrude feature. Conceptually, a person knows that it creates a 3D object by projecting a 2D sketch linearly in a direction normal to the sketch plane, as demonstrated in Figure 12. A human can look at the input sketch and the resulting geometry, and piece together the general idea of what the feature does. The person can also imagine what types of geometry could be expected from different types

of input. The feature is an abstract concept that conveys the general idea of a shaping process, and through reasoning a person can understand the relationships between input and output, and infer the set of rules automatically. Therefore, to describe the concept of the feature in terms an ontology reasoner could understand, rules that correlate specific inputs types to specific output types must be used in the feature definition as necessary and sufficient conditions. In the extrude example, it is clear that every entity in the sketch is going to create a surface and there will be two planar surfaces on each end. To specify further, it is also true that every line entity in the sketch is going to create a planar surface, each circle or arc will create a cylindrical surface, and each spline in the sketch will create a spline surface. To generalize the feature more to include cases where the solid extrude is intersecting another object, the rule can be modified so that for every sketch entity, the feature must create at most N+2 surfaces. In essence, the feature is being described by a set of rules, or necessary conditions, which must always hold true for a valid instance of that feature. The rules can be more technical, by correlating specific sketch inputs to specific geometry output values, such as if a cylindrical surface entity has radius *R*, then there must be a circle or arc entity with the same radius value. This can be extended to all types of features defined in feature-based design because all features must have a set of rules that correlate input to a predictable behavior. This can be used to create class hierarchies to allow a reasoner to automatically determine which classes the feature being translated can belong to without relying on exactly matching semantic data. Taken to the logical conclusion, a comprehensive set of rules that define a feature in its local ontology would be the same rules that are being implemented internally within the specific CAD system to automatically convert the user input to the valid part geometry.

From an implementation standpoint, the process of data exchange would work as a dynamic mapping process. First the data for a specific instance from the source system is extracted from the system format. Once extracted, it is parsed into the shared base

ontology format. This data contains the sequence of features in the three-branch CAD feature model, which stores all relevant data pertaining to the feature. Once the data has been exported, it could be used by any CAD system which supported importing of data from the shared base ontology format. The mapping and reasoning only becomes necessary during the import process. During the import process, the reasoner of the target system would examine each feature being imported from the shared base ontology file, and run a series of tests on these features. For each imported feature, these tests check to see which feature subclasses in the target feature library have rules that are not being violated, and therefore could possibly recreate the feature. Once these tests have eliminated all feature classes which cannot reproduce the source feature, then more traditional similarity mapping can resolve any remaining semantic problems. In cases of structural heterogeneity, additional rules could verify when types of feature data are superfluous, when additional information is required and what form it must take, or when a feature has no equivalent and cannot be reproduced. Additionally, with the B-Rep stored with the file, the resulting geometry could be used to verify the match. Once a match has been found, the mapping can be stored, and the translation could be performed without the testing from then on. This process is illustrated in Figure 13.

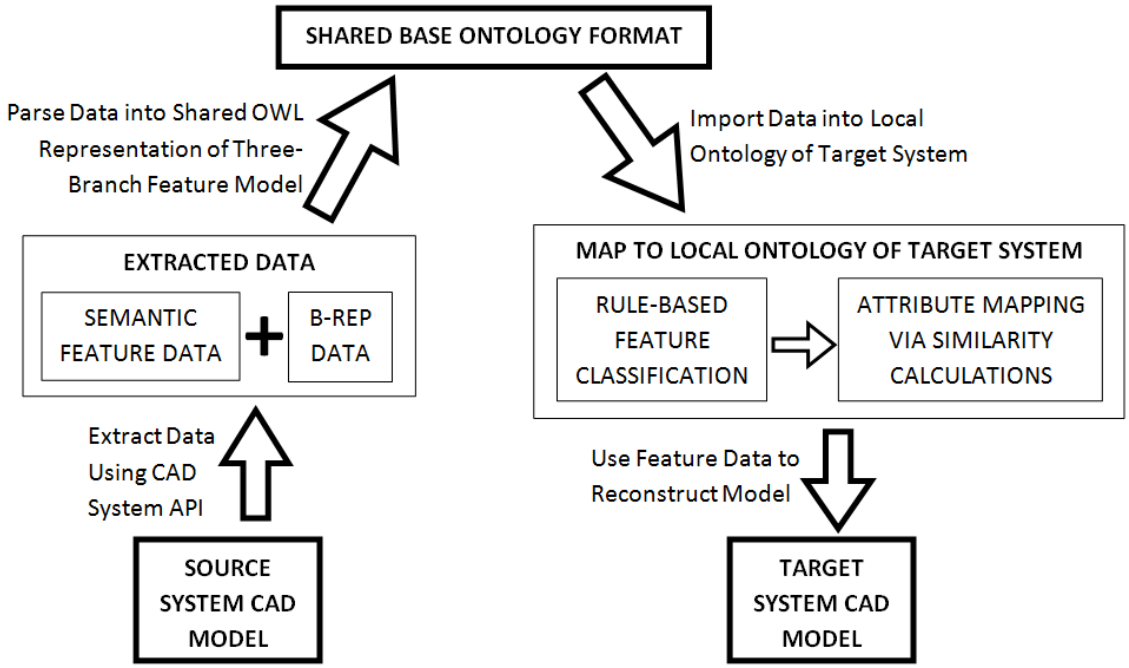

Figure 13: General Approach to Feature Interoperability

It is important to note, however, that this approach would really only be effective in the case of a distributed environment, where it is actually the CAD companies developing the export and import algorithms to work with their systems. This approach requires very high level understanding of the entire library of the target system to implement these rules effectively, and exporting all of the necessary data via the API is somewhat problematic. If a third party company set out to make a translation program to work with the various CAD programs, it would require extensive knowledge of all CAD libraries, and with that knowledge it would be far more efficient and reliable to manually determine and code the mapping of one feature onto another instead of trying to describe a series of rules to define each feature. The strength of this approach lies in the fact that no knowledge of other CAD systems is required for interoperability so long as the shared base ontology is used. Once a set of classification rules is determined for a specific CAD system's local ontology, then any feature can be imported from any other system so long

as it is described using the shared base ontology. It should also be noted that this approach would only be appropriate when the goal is to maintain the original design intent of the model. If the goal is only to recreate the geometry with parametric data, then simply using feature recognition techniques on a geometric model would be more practical. The advantage of this approach is that it attempts to recreate a model in a new system using the same steps, constraints, and parameters as were used in the original design of the part.

Use of OWL format

In order to construct an ontology in computer science, an ontology language is used. There are numerous formal knowledge representation languages which can be used to represent an ontology. The choice of the OWL (Web Ontology Language) [49] was made because it is powerful tool for describing ontologies and as a World Wide Web Consortium (W3C) recommended standard, there are various tools and support available for this language. OWL is a vocabulary extension of the RDF (Resource Description Framework) [50] that was used in previous research by our group. The RDF data model is a way to store information as a set of statements. Each statement is in the form of a triple, which has a subject, predicate, and object. For example, a statement made stating that a person "John Smith" has mother "Jane Smith", then "John Smith" would be considered the subject, the predicate is "has mother" and the object is "Jane Smith". The RDF Vocabulary Language, RDF Schema (RDFS) [51] is used to define a set of standard RDF resources and properties. This provided a mechanism for grouping related resources into RDF classes. An RDF class is a generic concept of a specific type or category. The vocabulary of an RDF ontology is created by defining a standard set of RDF resource classes and relating them via property classes. For example, one could create a class called "human" which would have subclasses "man" and "woman". Because "man" is a subclass of "human", declaring "John Smith" as an instance of "man" also implies that he

is an instance of “human” as well. A graph of this data could be constructed to relate instances of classes to the properties that relate them. For example, the class “human” can be defined to have two properties, “has birth mother” and “has birth father”, the objects of those properties would be the classes “woman” and “man” respectively. RDFS allowed the creation of ontologies with subclass hierarchies, but lacked means to express relationships between properties and classes.

OWL expanded upon RDFS in many ways. In OWL, classes could be defined as logical combinations such intersections, unions and complements of other classes. Classes could also be defined as disjoint, meaning individuals could not be instances of both classes. It also enhanced properties by allowing them to be stated as transitive, symmetric, functional, or the inverse of another function. The most important enhancement was the addition of constraints and restrictions that can be applied to classes. Using the previous “human” class example, it is possible to add the restriction that every instance of “human” must have one and only one instance of the class “woman” for the property “has birth mother”. Additionally, OWL can be used to specify types and values that must be obeyed. For example, the property “has age” could be added to the class “human”, and could be specified such that every instance of “human” must have one and only one instance of “age”, and the value of “age” must be an integer that is greater than or equal to 0. Addition of these restrictions allowed OWL to be a much more expressive language while also reducing ambiguity among class definitions.

The formal specifications of the OWL language were influenced by over 10 years of Descriptive Logic research as stated by Horrocks et al. [52]. Description Logic [53] is a family of formal knowledge representation languages used to express domain-specific concepts and the relationships between them. In Description Logic, classes are referred to as concepts, and properties are referred to as roles. The fundamental concept of Description Logic is the use of axioms, which are logical statements relating roles and concepts. Axioms are defined using combinations of constructors to build complex

concepts from simpler ones and place emphasis on the decidability of reasoning problems. The goal of OWL was to create a language with increased expressiveness while retaining reliable and efficient reasoning support. To maintain that balance, OWL was based on the $\mathcal{SH}$ family of Descriptive Logics [54]. The $\mathcal{S}$ designates a family containing constructors of the Attributive Concept Language with Complements ($\mathcal{ALC}$) family, which includes intersection ($\cap$), union ($\cup$), complement ($\neg$), universal restriction ($\forall$), and existential ($\exists$) constructors, as well as transitive roles, which become transitive properties in OWL. The $\mathcal{H}$ designates the support for role hierarchies, or subproperties in OWL. Other members in the $\mathcal{SH}$ family are $\mathcal{SHIQ}$ [55] and $\mathcal{SHOQ}$(**D**) [56] Descriptive Logics, where $\mathcal{I}$ indicates the use of inverse functions, $\mathcal{Q}$ allows the use of generalized cardinality restrictions, $\mathcal{O}$ allows the creation of a class as enumerated instances, and (**D**) indicates the use of datatype properties. There are three versions of OWL, which include OWL Lite, OWL DL, and OWL Full. In OWL DL and OWL Lite, only certain constructors are allowed, and they can only be combined in certain ways, essentially making them expressive Description Logics. OWL DL can be best described as a $\mathcal{SHOIN}$(**D**) Descriptive Logic, where the $\mathcal{N}$ denotes cardinality restrictions ($\geq$, $\leq$, and =) on properties. OWL Lite is similar to the $\mathcal{SHIF}$(**D**) Description Logic, where the $\mathcal{F}$ denotes that functional properties are allowed, which is equivalent to only allowing cardinality restrictions that equal 1. OWL Full allows all types of RDF graphs, making it even more expressive than OWL DL, but no longer decidable as a consequence.

It should be clear how the addition of Descriptive Logic techniques to an ontology by means of restrictions allows for a much better representation of a feature as an ontology class. By defining features in terms of the numbers and types of properties that define them, an ontological reasoner can automatically group features into different hierarchies through use of necessary and sufficient conditions. For example, consider the property “hasSketch” applied to a feature. An ontological reasoner would automatically separate those features that do not require a sketch from those that do. Therefore, when it

becomes necessary to map a feature to another, the algorithm would simply traverse the branching hierarchy until it is either matched with an equivalent feature, is identified as a unique subclass of an existing feature, or is one of several subclasses of a class of features.

In OWL, object properties relate an individual of one class to individuals of another class. The domain and range of each object property can be specified to only apply to certain types of classes. Likewise, datatype properties are used to relate an individual of one class with a value of a particular datatype. These datatypes can be Boolean, integer, float, or string, etc. It is also possible to specify the cardinality (number of required properties) and types of each property a class is allowed to have through property restrictions. However, OWL's limited expressiveness excludes property-chaining or axioms with variables [52]. For example, OWL permits cardinality restrictions that specify whether a given class has greater than, less than, or exactly a certain number of a property. However, this number must be an integer specified at class definition. An extrude feature could have the property restriction without any problem:

$$Cardinality(hasSketch) = 1$$

But the cardinality cannot be a variable or a number that must be determined, so rules such as the following cannot be implemented in OWL:

$$Cardinality(createsPlaneSurface) \leq Cardinality(hasLineEntity)$$

These property restrictions also do not allow for any limitations on values of datatype properties, nor do they allow the inference of new properties. They can be used to infer when an instance is a member of another class, but even those restrictions are limited because OWL uses an open world assumption. In an open world assumption,

something cannot be determined to not exist until explicitly stated. For example, if there is a class *Parent* with a restriction that any individual with property *hasChild* is a *Parent*, then only conclusions can be made about individuals that have children. If the individual does not have any instances of the *hasChild* property, it **cannot** be concluded that the individual is not a *Parent*, it remains unknown. The open world assumption works this way to allow for introduction of new data over time. These rules prevent new information from invalidating previous conclusions. OWL uses the open world assumption because it was mainly designed for knowledge representation and assumes that one cannot have full knowledge of a domain and that new data may be added over time. However, when used as an exchange file, all instances that will exist are present, and no new data will be introduced without creating an entirely new file, so using a system with an open world assumption offers no benefits over a system with closed world assumptions and limits the types of inference rules which can be implemented.

Despite these limitations, OWL was still chosen to create the CAD ontology because it is the most developed ontology language and better suited to create such a large and expressive ontology. The construction of such a large CAD ontology would have been much more difficult without the tools that have been developed to support OWL. To overcome the limitations in expressiveness of OWL, the Semantic Web Rule Language (SWRL) [57] was used. However, SWRL also adopts the open world assumption, so it is also somewhat limited. This means that it cannot make inferences based on the number of a particular class or property if that number may change and invalidate the inference. This also means that it only supports monotonic inferencing, meaning existing information in the ontology cannot be edited or removed using these rules. The monotonicity also means SWRL rules cannot support negation (NOT operator) and disjunction (OR) statements. Therefore, from a basic logic standpoint, only conjunction (AND) statements can be used to create rules. One of the reasons why OWL and other Description Logic languages have not extended to include non-monotonic and

closed world reasoning is because there are workarounds such as the ones proposed in this thesis, but they usually require the introduction of additional data that is introduced outside of the reasoner. For example, it is possible to impose closed world reasoning with use of OWL datatype properties that explicitly declare the number of each instance of a class and imposing SWRL rules, but the number of instances must determined externally through ad hoc processes. Future work would require a more appropriate language approach than the one proposed in this thesis to be developed, but such a development is beyond the scope of this project and is a significant research topic by itself. The inclusion of non-monotonic constructs in Descriptive Logic is still an active research area with recent works by Grimm et al. [58,59], Katz and Parsia [60], Hosain and Jamil [61], and Knorr et al. [62]. Further description of SWRL limitations and how they are overcome will be discussed in Chapter 6.

CHAPTER 4

THREE-BRANCH CAD FEATURE MODEL

The three-branch CAD feature model that is proposed is meant to model a feature in terms of the individual settings and parameter values selected by the user during feature definition. To better classify the types of attributes used to define a feature, they are separated into two main categories which will be called *reference* attributes and *parameter* attributes. The reference attributes pertain to information that is necessary to the feature definition, but is defined externally from the feature. This can be a pointer to a reference datum, existing face, edge, or vertex, and sketch data for sketch-based features. Conversely, the parameter attributes refer to information that belongs exclusively to the feature, such as an option or number that the user specifies, which is independent of other features. This distinction is important because the parameter attributes can be compared and converted directly and use established datatypes like floats and integers, but the reference attributes refer to existing entities, which will likely have different identifiers in different systems. For example, consider a feature that rounds an edge. The parameter attributes would identify the type of round and value of the radius, but the reference attribute would be the specific edge identifier, which would differ between CAD systems without a persistent naming convention. These two sets of attributes make it possible to describe a class-level representation of the feature, as the types and amount of reference attributes should be independent of the specific values entered in the instance-level.

The third branch of the feature model represents the *B-Rep operations*, which stores the changes the feature makes to the B-Rep at the instance-level representation of the feature. This cannot be described in a class-level representation because the changes a feature makes to the B-Rep can only be determined when the reference and parameter attributes have specific values. The B-Rep Operations information is similar to the

"feature rewrite" described by Rappoport et al. [26]. These data are mainly used as a verification measure, because values of B-Rep cannot be determined without specific instance data. However, the feature definition rules can be used to verify the number and type of certain B-Rep entities. The *feature type* attribute is used to store the class type of the feature from its source system, so that it can be used as an alias once a match has been established. Strictly speaking, it is unnecessary for feature classification purposes, but it allows the target system to know what type of feature it has mapped, so future instances of the same feature type can be recognized. A general diagram of the three-branch CAD feature model can be seen in Figure 14.

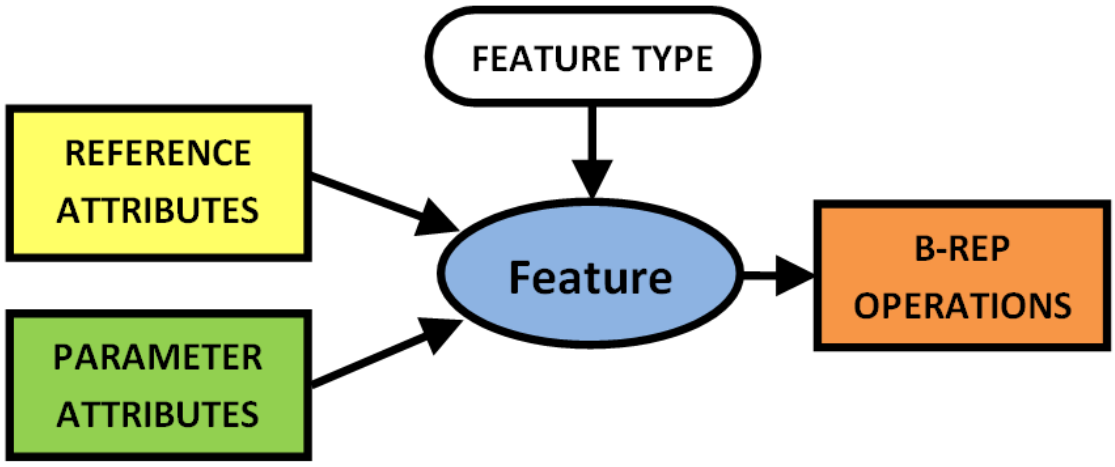

Figure 14: General Structure of Three-Branch CAD Feature Model

The three-branch CAD feature model serves as a template for describing a model using the shared base ontology language. The goal of the three-branch feature model is to strike a balance between prescribing a neutral format through which all features can be described and maintaining the expressiveness and individuality of each CAD system. Classes for all three branches are well defined in the global ontology, so the only thing that should be unique to the local ontologies of each CAD system should be the types and

quantity of class instances described by each feature. Descriptions of the classes from the shared base ontology will be provided in the next chapter. This template is necessary to reduce the amount of semantic and structural heterogeneity as much as possible without constraining the expressiveness of the feature definitions from the CAD systems. This has several advantages over the hybrid semantic feature model. The added expressiveness allows the comparison of parameter attributes on more factors than just name and datatype, because the base ontology can contain definitions for various common parameter classes, such as radius and length, will ensure that semantic heterogeneity is avoided when possible. Similarly, the definitions of reference attributes and B-Rep Operations can be standardized because they are based on geometric concepts that have been well established and are fairly universal amongst different CAD systems. Specifying a standard structure and nomenclature for these reference attributes and B-Rep operations does not restrict the expressiveness of features and in most cases would be a direct process or require a simple conversion. This approach does not attempt to mandate that standard features follow a shared definition, but only that the existing feature definitions be parsed into a particular format, so it should be much easier to implement. Problems with persistent naming will still be a problem with reference parameters and B-Rep operations, but those can be resolved using tests to determine equivalency.

The following three figures demonstrate simplified graphs of how the three-branch CAD feature model might look for different types of features. Figure 15 shows an example of an extrude feature. The reference attributes are the reference plane used to define the orientation and the sketch used to create the shape. Parameter attributes determine whether it creates a solid or surface model, the option to set the depth type, the depth value (if blind or symmetric is chosen), whether to flip the direction of the sketch normal, and whether the extrude feature is being used to add or remove material (Boolean union or difference).

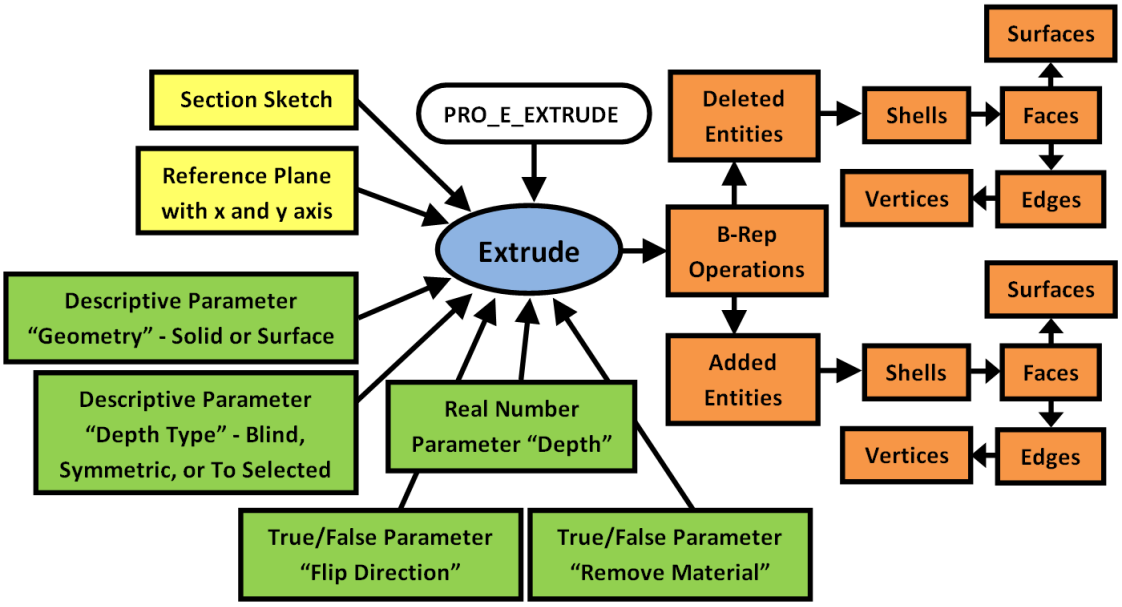

Figure 15: Three-Branch Extrude Feature Model Example

Figure 16 shows an example of a revolve feature, which is like an extrude in that it uses sketch data, but instead rotates it about an axis to create geometry. As such, it has the same reference attributes, with the addition of an axis of rotation to determine the line about which the sketch is rotated. It has fewer parameter attributes, because revolves are specified using the angle of rotation rather than depth, and thus have fewer options.

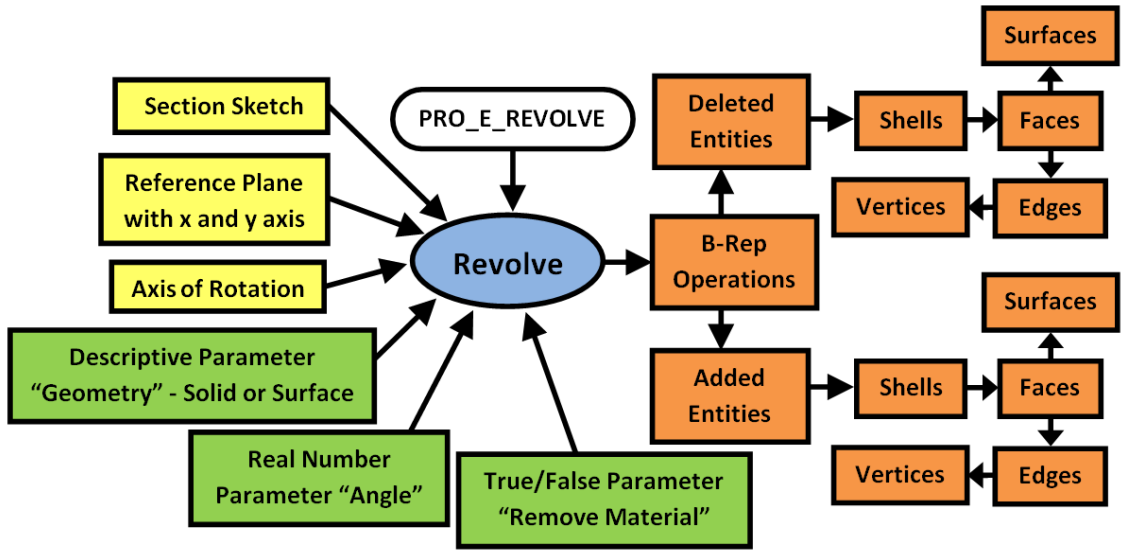

Figure 16: Three-Branch Revolve Feature Model Example

Figure 17 illustrates an example of an edge rounding feature. The edge round takes only existing edges in the part and rounds them to a given radius. Therefore, its only reference attributes are edge entities created by earlier features in the construction history tree. For parameter attributes, it requires a number to define the radius of the round, and the user must specify how the transitions of intersecting rounded edges should be handled, as was demonstrated in Chapter 3.

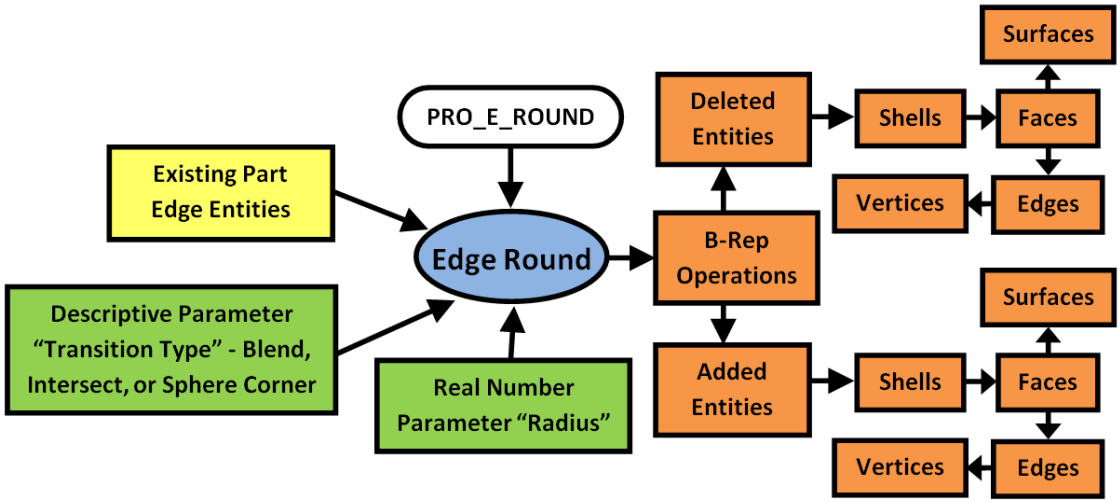

Figure 17: Three-Branch Edge Round Feature Model Example

These three examples show the general structure specified by the three-branch feature model. The examples show only the basic classes of B-Rep operations because without specific instance information, they are not defined. With this general structure of how features should be defined, the next step is to generate the shared base ontology language through which these three-branch CAD feature models can be described. This is discussed in the next chapter.

CHAPTER 5

OWL REPRESENTATION OF SHARED BASE ONTOLOGY

To create the shared base ontology used to represent CAD data in OWL, Protégé-OWL [63] was used. Protégé-OWL is an extension of the open source ontology editor Protégé that supports the OWL language. Protégé-OWL is a tool that allows creation of OWL files visually using tools to create classes, properties, and restrictions and the actual OWL file is created automatically. This is an invaluable tool because manually coding all the classes and relations for a large ontology would be very difficult. The OWL file containing the shared base ontology that has been included in Appendix A, and should give a clear indication of how complicated manually writing an ontology file would be. Protégé-OWL also allows the easy modification of class definitions as well as tools to check the consistency of the file. Because the shared base ontology created in Protégé is so large, it will be broken up it into several main categories corresponding to the various concepts used in the three-branch CAD feature model. It should be acknowledged that this ontology is not meant to be definitive or all-encompassing, but simply to illustrate an example of how a shared base ontology language could be constructed for the CAD domain. It would be more appropriate for a definitive CAD base ontology to be established by an organization such as the National Institute of Standards and Technology (NIST) with input from the various CAD vendors.

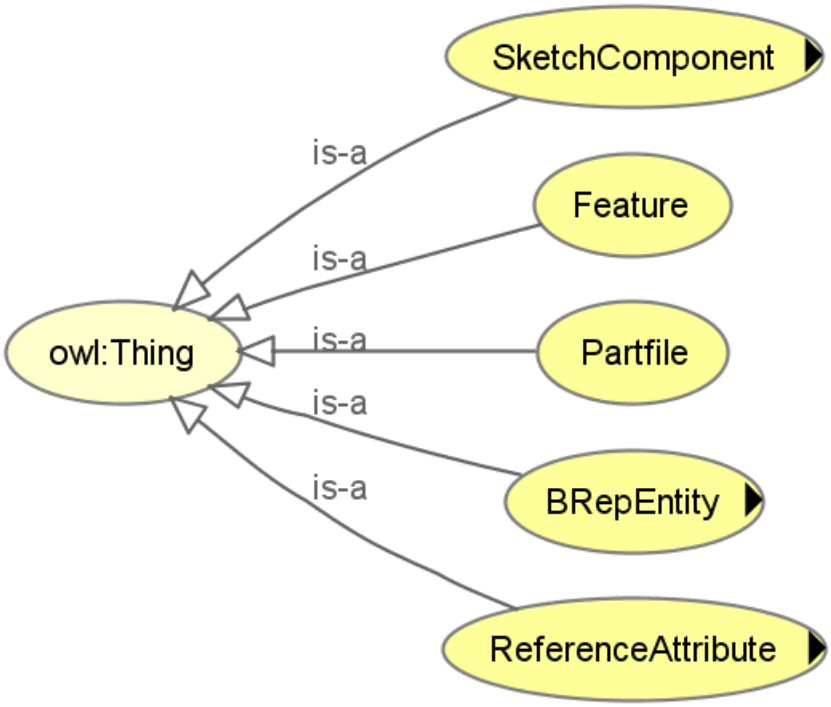

Figure 18: The Main Classes of the CAD Ontology

Figure 18 shows the top level classes used to define the shared base ontology. Each arrow represents an "is-a" relationship in this taxonomy. As illustrated, each of the five main classes are considered a subclass of "*owl:Thing*". Every OWL class is ultimately some type of thing or concept, so each the "*owl:Thing*" class is always the top-most level. If the ontology were further to further expanded to include other domains, "*owl:Thing*" could have a subclass such as *"EngineeringConceptDomain"* that could have a subclass *"CADDomain"* and so on, but for the purposes of this project, such distinctions are not necessary. Additionally, each of the five main classes is disjoint with its sibling classes. What this means is that a *ReferenceAttribute* cannot be a *BRepEntity*, *Feature*, *Partfile*, or a *SketchComponent*. Of these five main classes, the *Feature* and *Partfile* classes serve only to act as place holding classes. The *Partfile* class is the class in which the feature tree is stored, and must be included in the base ontology for parts to be recognized between different CAD systems. The restrictions on this class state that it

must have at least one coordinate system, must have at least three reference datum planes (Front, Right, Top), and must have at least one feature. The *Feature* class serves as the parent class for all feature classes in specific CAD system local ontologies. Every instance of a specific local ontology feature class must also be an instance of the shared base feature class. The only restrictions on the *Feature* class are that it must have at least one B-Rep operation and at least one reference attribute.

The remaining three classes are used to define specific concepts that should be universal to different CAD systems and create the vocabulary and structure that the local ontologies should use to define these concepts. Sketches are prevalent in feature based modeling, as several different types of features use 2D drawings to create 3D objects. The *BRepEntity* class describes the various topology and geometry elements and how they are related and store data through specific properties. The *SketchComponent* class defines the various types of sketch entities, constraints, and dimension types that compose a sketch and the properties each should have. Finally, the *ReferenceAttribute* class describes the various types of references, such as sketches, reference planes, coordinate systems, and existing part reference types, that can be used in a feature definition. These will be described in further detail in the following sections.

BRepEntity Class

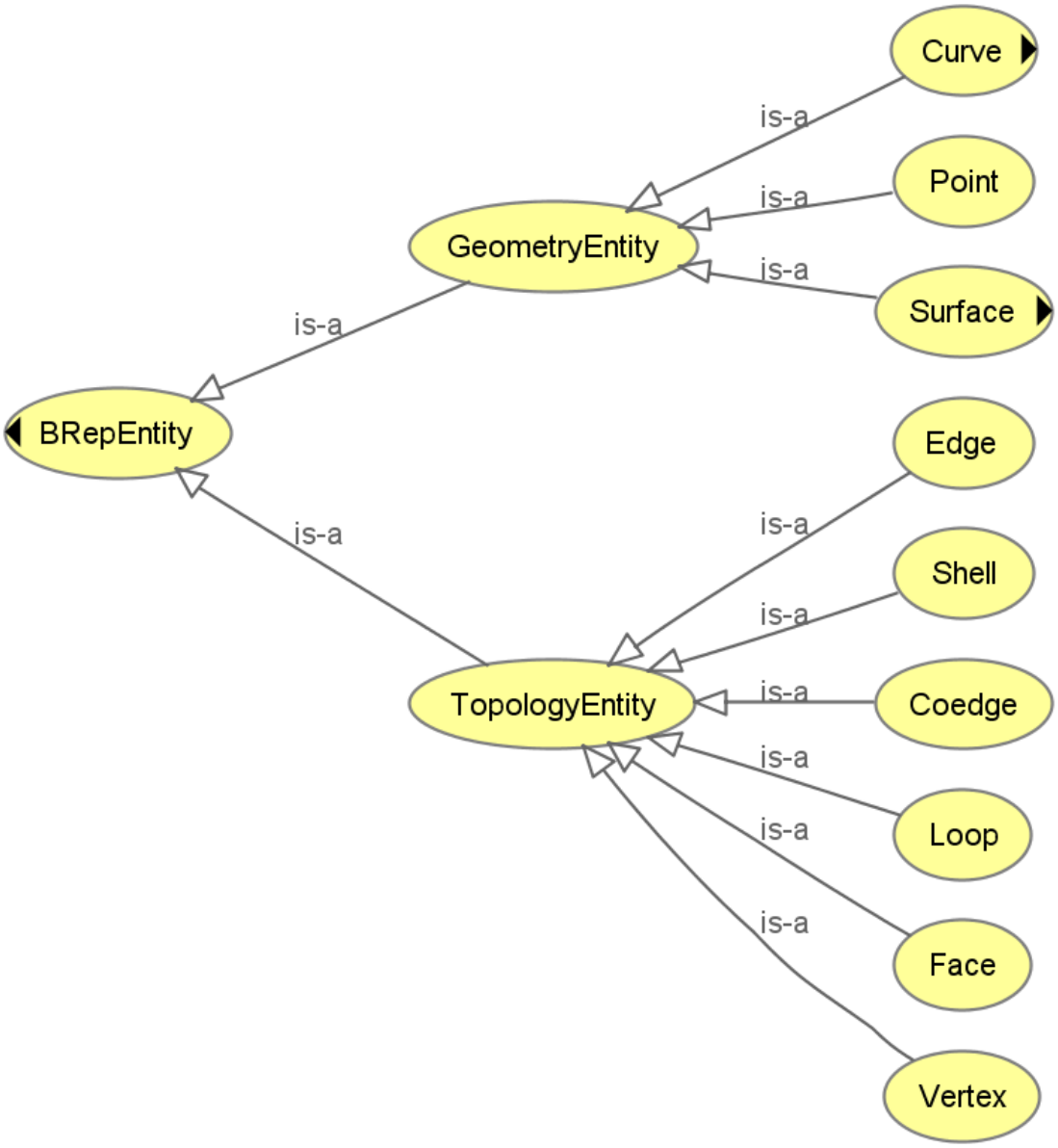

Figure 19: *BRepEntity* Class Structure

The basic structure of the *BRepEntity* class is displayed in Figure 19. The *BRepEntity* class is separated into Topology and Geometry entities. Please note that this diagram only shows the "is-a" relationships. Each class also has a number of properties

associated with specified cardinalities associated with them that must be satisfied to create a valid instance of the class. Also note that both the *Curve* and *Surface* geometry subclasses also have subclasses that are not displayed, as evidenced by the black arrows. The structure of the BRepEntity subclasses was based on the structure of the ACIS neutral format [45,46]. This was chosen because it is a well known industry standard that is easy to extract data from. To understand the properties that are used to define the between the topology subclasses, see Table 1.

Table 1: Properties of Topology Subclasses

Domain	Property	Range	Cardinality
Shell	*hasFace*	Face	≥1
Face	*isFaceOf*	Shell	=1
	hasLoop	Loop	=1
	hasGeometryOfSurface	Surface	=1
Loop	*isLoopOf*	Face	=1
	containsCoedge	Coedge	≥1
Edge	*hasVertex*	Vertex	=2
	hasStartVertex	Vertex	=1
	hasEndVertex	Vertex	=1
	hasCoedge	Coedge	=2
	hasForwardCoedge	Coedge	=1
	hasReverseCoedge	Coedge	=1
	hasGeometryOfCurve	Curve	=1
Coedge	*isCoedgeInLoop*	Loop	=1
	isCoedgeOf	Edge	=1
	isForwardCoedgeOf	Edge	0 or 1
	isReverseCoedgeOf	Edge	0 or 1
	hasNextCoedge	Coedge	=1
	hasPreviousCoedge	Coedge	=1
	hasOppositeCoedge	Coedge	=1
Vertex	*isVertexOf*	Edge	≥1
	isStartVertexOf	Edge	≥0
	isEndVertexOf	Edge	≥0
	hasGeometryOfPoint	Point	=1

Note that these properties must apply to every instance of the entity and that the cardinality specifies the number of instances that are objects of that property. Each property has a domain, which is a list of classes that can be the subject of the property, and a range, which is a list of classes that can be the object of the property. In the cases presented, each property only has one class type for both the domain and range because the properties were designed to represent the specific relationships between the B-Rep entities. Each shell must have at least one face, but can have any number, so the cardinality is ≥ 1. Each face is defined by one unique loop and has the geometry of one surface, so the cardinality for both must = 1. A loop contains at least one coedge, but can contain many more. An edge must have exactly two vertices (a start and an end vertex), two coedges (one pointing in the same direction, and another opposite), and one curve geometry to describe it. A coedge must have one previous coedge, one next coedge, and one opposite coedge that it shares with its edge. Finally, a vertex is defined by 1 point. Additionally, there are inverse functions that correspond to the functions above, with their own cardinality. A face can only belong to one shell, a loop can only belong to one face, a coedge can only belong to one loop and one edge, and a vertex must belong to at least one edge although it is usually three or more. Note that a coedge cannot be both a forward and backwards coedge, it must be one, and its opposite coedge will be the other.

The geometry subclasses can be divided into more descriptive subclasses that mirror the different classes of surfaces and curves that are stored in the ACIS format. These are displayed in Figure 20. The *Surface* class is defined as being a planar, conical, spherical, toroidal, or spline surface. Similarly, the Curve class can be defined as a linear, elliptical, helical, or interpolated curve. By defining these basic geometry terms, a standard way to store the exact geometry in a standard format, such as those employed in the ACIS or STEP standards, can be developed. Each surface belongs to one face, each curve belongs to one edge, and each point contains the x, y, and z coordinates of a single vertex.

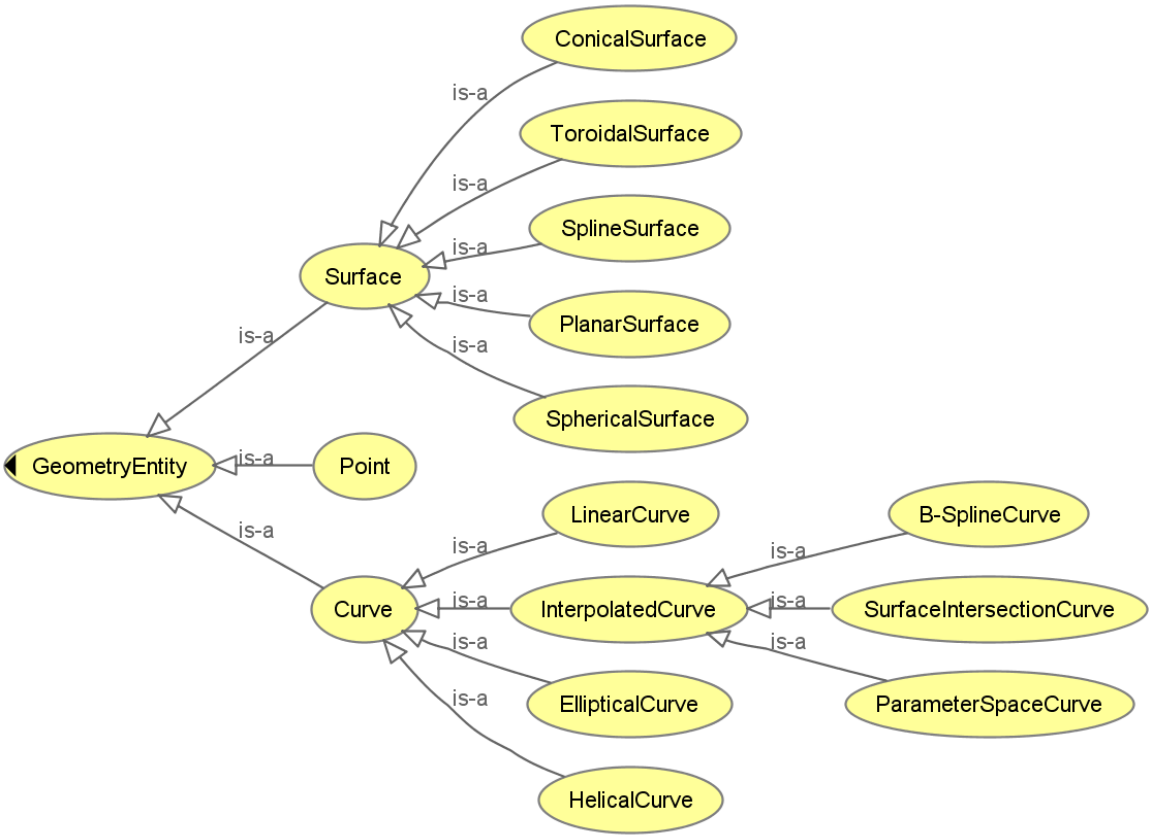

Figure 20: *GeometryEntity* Subclasses

SketchComponent Class

The *SketchComponent* class was created to describe a common vocabulary for the various sketch entities, constraints, and dimensions that are fairly consistent between different CAD systems. The class was deemed necessary because sketch-based features are one of the most common ways to create a model in a CAD system, and adding dimensions and constraints in a sketch are common ways to convey the design intent of the solid model. The *SketchComponent* class is defined by three main subclasses, grouped as *SketchEntity*, *SketchConstraint*, and *SketchDimension*. These are displayed in Figure 21-23.

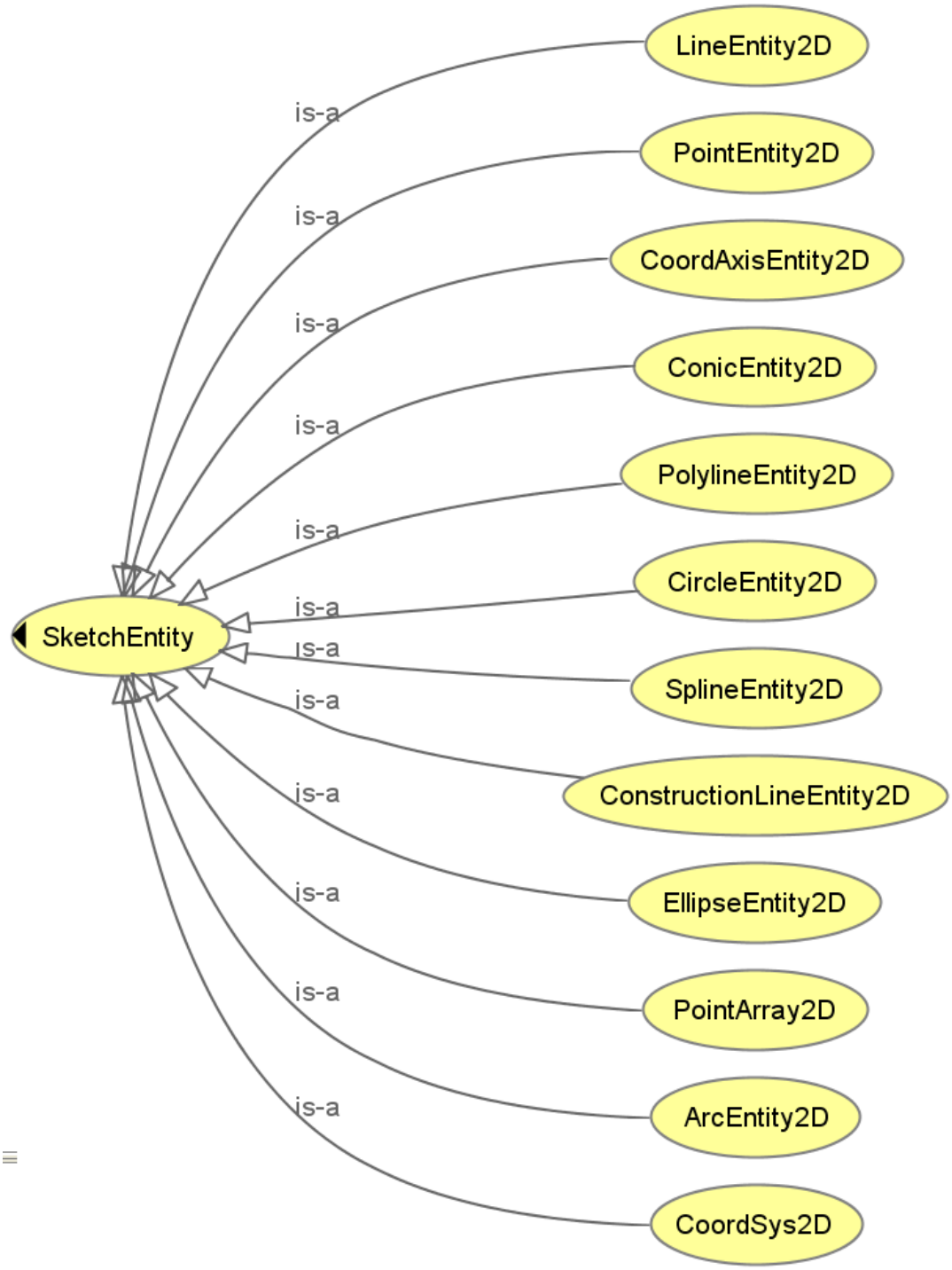

Figure 21: Subclasses of *SketchEntity*

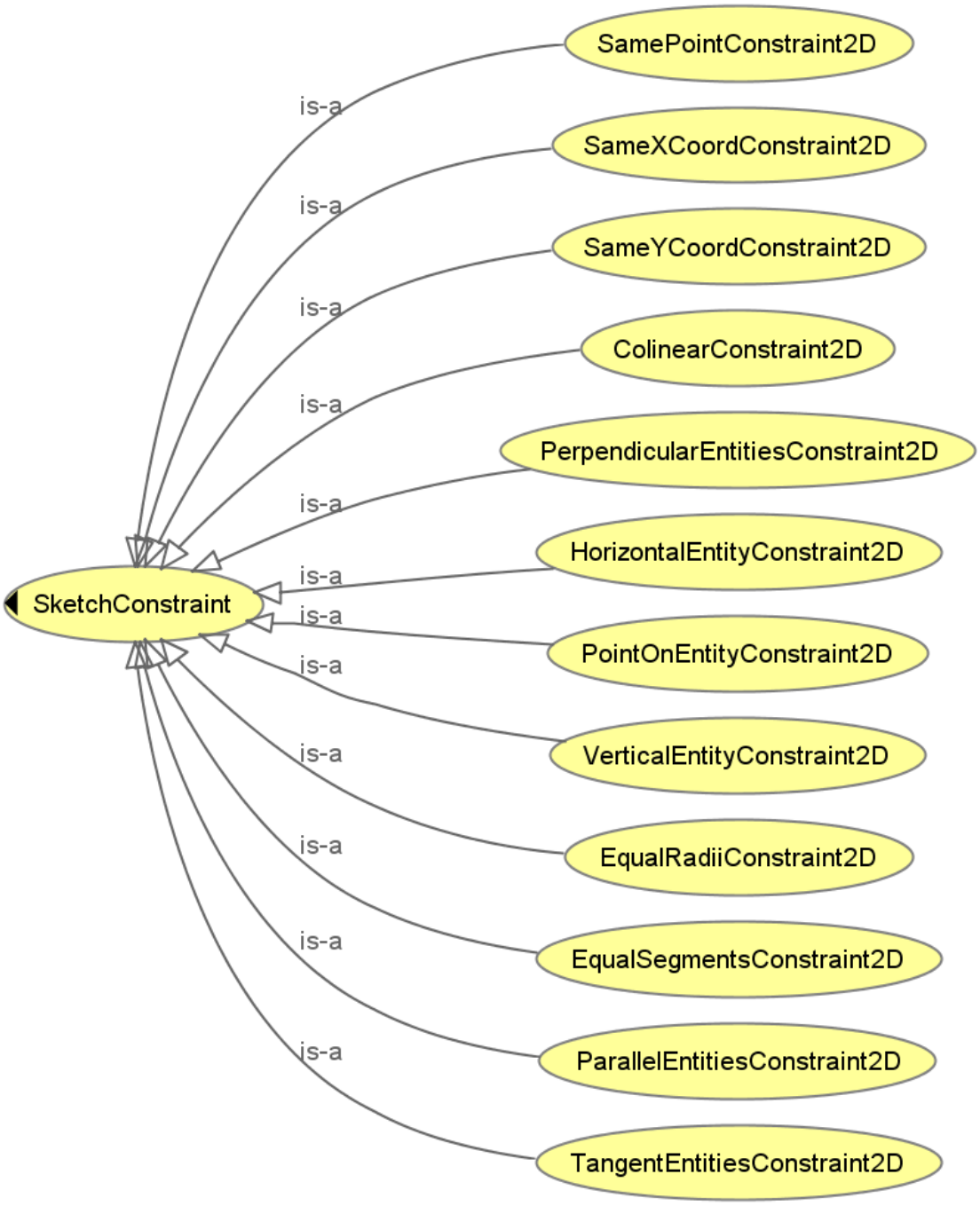

Figure 22: Subclasses of *SketchConstraint*

Figure 23: Subclasses of *SketchDimension*

The different subclasses of *SketchConstraint* and *SketchDimension* only serve to describe the constraints and dimensions placed on instances of *SketchEntity* during the construction of the sketch. All the data required to reconstruct the sketch is stored in the instances of *SketchEntity*, so the constraint and dimension data is only used to convey the original design intent of the sketch. If the sketch were to be reconstructed with only the *SketchEntity* data, the CAD system would have to automatically apply constraints and dimensions for the system to be fully constrained, which would more than likely not be the same as the set used by the original designer. Table 2 lists the properties for each subclass of *SketchEntity*. Note that in this table, there are both *object properties* and *datatype properties*. *Object properties* link an individual to another individual, while *datatype properties* link an individual to a specific type of value. In the table, *Datatype properties* will be italicized and in green and their domain will be the type of value it requires in brackets.

Table 2: SketchEntity Subclasses

Domain	Property	Range	Cardinality
PointEntity2D	*hasXCoord*	[float]	=1
	hasYCoord	[float]	=1
CoordSys2D	hasPoint	PointEntity2D	=1
CoordAxisEntity2D	hasStartPoint	PointEntity2D	=1
	hasEndPoint	PointEntity2D	=1
ConstructionLineEntity2D	hasStartPoint	PointEntity2D	=1
	hasEndPoint	PointEntity2D	=1
LineEntity2D	hasStartPoint	PointEntity2D	=1
	hasEndPoint	PointEntity2D	=1
ArcEntity2D	hasCenterPoint	PointEntity2D	=1
	hasStartAngle	[float]	=1
	hasStartPoint	PointEntity2D	=1
	hasEndAngle	[float]	=1
	hasEndPoint	PointEntity2D	=1
	hasRadius	[float]	=1
CircleEntity2D	hasCenterPoint	PointEntity2D	=1
	hasRadius	[float]	=1
EllipseEntity2D	hasCenterPoint	PointEntity2D	=1
	hasXradius	[float]	=1
	hasYradius	[float]	=1
ConicEntity2D	hasStartPoint	PointEntity2D	=1
	hasEndPoint	PointEntity2D	=1
	hasShoulderPoint	PointEntity2D	=1
	hasConicParameter	[float]	=1
PointArray2D	*hasN_Points*	[integer]	=1
	hasPoint	PointEntity2D	=N
PolylineEntity2D	hasPointArray	PointArray2D	=1
SplineEntity2D	hasPointArray	PointArray2D	=1
	hasStartAngle	[float]	=1
	hasEndAngle	[float]	=1

The entities in the Table 2 should be understood fairly easily by anyone familiar with basic geometric curves and shapes. The only shape definition that requires some additional information is the *ArcEntity2D* class. This class was made specifically with redundant information to better facilitate the use of constraints and dimensions. This is because the minimum required information for defining an arc is a center point, a radius, and the two angles that bound it. However, the start and end points of the curves can be easily derived from this information, and are more often used in constraints and dimensions, so they were added to the definition. The constraint and dimension entities are fairly straight forward, as the names themselves explain the use for each one. The only properties the constraints have are *isConstraintOfEntity*, which are restricted to the types of entity according to the constraint type. Likewise, dimensions have the property *isDimensionOfEntity*, which is restricted to the appropriate entities for the dimension and a float datatype property *hasDimesnionValue* which stores the value of the dimension.

ReferenceAttribute Class

This class is used to define the attributes that are necessary to creating a feature but are external to the feature definition, meaning they must already be defined in order for the feature to use them. This class of attributes is usually the first thing that must be assigned in the creation of a feature and is one of the three main components of the three-branch CAD feature model. Figure 24 shows the basic structure of this class.

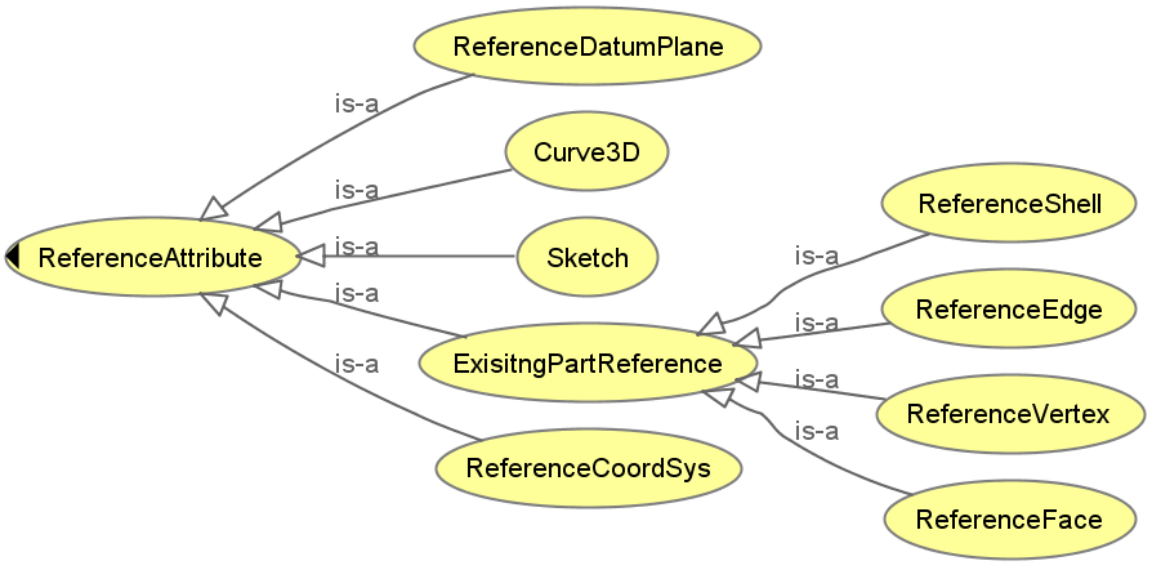

Figure 24: *ReferenceAttribute* Class Structure

The figure illustrates common references required in the creation of a feature, but is far from comprehensive. The *ExistingPartReference* would contain B-Rep entities that already exist in the part model. For example, if the user wanted to extrude a shape from the face of an existing part or to chamfer an edge, the B-Rep entity that is selected as the reference input would be stored here. The *Sketch* subclass is where sketch data is stored as instances of the *SketchComponent* subclasses. A sketch must have at minimum one instance of the *hasSketchEntity* property to be valid. The *Curve3D* class would be where a curve for a sweep feature would be stored. Because this class is dedicated to referencing existing parts, it will be necessary to resolve the naming persistency problem to make full use of this in an exported file.

Shared Ontology Property Types

The classes discussed previously describe concepts used in feature definitions, but in order to actually define a feature, those concepts have to be related to the feature through a series of properties. Previous sections have shown how properties are used to

define concepts, an example being the LineEntity2D class requiring both an instance of both the *hasStartPoint* and the *hasEndPoint* properties for an instance of the class to be valid. Similarly, object and datatype properties are also used to define features within the three-branch CAD feature model. The way in which features use properties in their definition is a major source of heterogeneity, so it is important to create a framework to describe a standard set of properties to avoid semantic incompatibility.

Of the three branches, it is more straightforward to describe the properties relating the reference attributes and B-Rep operations because they are object properties, as opposed to the parameter attributes, which are datatype properties. When relating features to reference attributes, the basic framework creates a property for every class of reference attribute, with the domain being a top level feature class and the range being the respective reference attribute. The feature is named after the reference attribute by appending the prefix "has". For example, to describe a feature that uses a sketch in its definition, the feature must have an instance of the *hasSketch* property, which specifies an instance of the Sketch class. In cases where the property must be more specific, then a subproperty can be defined. This is basically a more specific form of the parent property, for example a *hasParent* class could have subproperties *hasMother* and *hasFather*. To demonstrate an example of this in a CAD context, consider defining an extrude feature in Pro/Engineer, where the user must select a primary reference datum plane on which the sketch is created and a secondary reference datum plane to describe the orientation. In this case, the *hasReferenceDatumPlane* property is insufficient, since the feature must have two instances of the same property that are used in different ways. To resolve this, two subproperties which inherit the properties of the *hasReferenceDatumPlane* property but represent two distinct concepts are created. These subproperties are differentiated by name, *hasPrimaryReferenceDatumPlane* and *hasSecondaryReferenceDatumPlane*, which convey the difference semantically. Every instance of these subproperties is also an

instance of the *hasReferenceDatumPlane* property, which conveys how the concepts are related.

Properties relating the B-Rep operations are also fairly easy to describe. Every B-Rep property is described as a subproperty of the *hasBRepOperation* parent property, which has the *BRepEntity* class as its range. This parent property splits into the *hasTopologyOperation* and *hasGeometryOperation* subproperties, with ranges of the *TopologyEntity* and *GeomertyEntity* classes respectively. These subproperties are then divided into creation and deletion subproperties, which are then further specialized to refer to the various subclasses of topology and geometry entities, following the class structure illustrate in Figure 19. For example, *createsPlanarSurface* is a subproperty of *createsSurface*, which is a subproperty of *createsGeometry*. The ranges also follow this hierarchy, as the range of *createsPlanarSurface* is the *PlanarSurface* class, which is a subclass of *Surface*, which is then is a subclass of the *GeometryEntity* class. B-Rep operations were divided into purely creation and deletion operations because without a persistent naming convention, B-Rep entities can only be uniquely identified by their specific definition, so tracking changes is much simpler if you just consider the modification a deletion followed by a creation.

By now it should be more apparent why parameter attributes are slightly more problematic. Parameter attributes are defined using datatype properties, so instead of linking the feature to an instance of a particular class, the properties are instead linking to a specific value of a given datatype, such as an integer, float, string, or Boolean value. Like the reference attributes, a major problem arises with how to handle cases when more than one property is using data from the same range. This problem is compounded in parameter attributes because features often have multiple parameters that use the same data type and a datatype contains very little conceptual information. Fortunately, datatype properties can also have subproperties, so long as the subproperty has the same datatype as its range. Therefore, multiple subproperties can be used to distinguish between

different parameter types that use the same datatype. The problem then becomes a matter of creating a parameter attribute subproperty hierarchy that defines the common types of parameters. It is important to note, however, that even establishing a set of common parameters may not prevent semantic heterogeneity. For example, one program may describe the distance of an extrude feature using a float parameter called “depth” while another program uses “length”, “D1”, or any other type of name. In this case, setting a common vocabulary may become difficult. This problem has to be resolved through collaboration to determine a specific set of concepts, and through clever use of subproperties. For example, the *hasDimensionAttribute* property which stores float values could include subclasses such as *hasLengthTypeDimension* and *hasAngleTypeDimension*, which are used to distinguish between values which are expressed in units of length from those expressed in units of degrees or radians. The *hasLengthTypeDimension* property could have further subproperties such as *hasDepth*, *hasRadius*, *hasDiameter*, *hasThickness*, *hasLength* and other common ways in which lengths can be recognized. The benefit of this approach is that when mapping occurs, a semantic similarity mapping process would have more information to work with than simply the data type. When attempting to compare a feature which uses *hasLength* to one that uses *hasDepth*, it could move up one level and see that both are instances of a property that measures length, meaning a higher similarity than simply two float values. Other relationships could also be included, such a simple conversions. Common terms could be related through conversion rules, such as diameter equals 2 times the radius, or arc length equals radius times angle (in radians). The main point is to use property hierarchies to provide more information about the data which is stored. Other general practices which could be followed is using Boolean values in cases when there is a choice between two options, such as a check box, and string values when there is multiple options, such as a dropdown menu. Integers should only be used when dealing with options that require whole numbers, like features that copy parts or create patterns. Using integers to store options

types like in an enumerated list, or Boolean operation should be avoided, because it makes the information ambiguous, and is counterproductive for data interoperability. The goal is to present data in a way that can be easily understood both by a human and ontology reasoner with as little knowledge of the source system as necessary. The last property to note is the *feature type* property assigned to allow for faster mapping once a valid match has been already established. This *feature type* property is stored as a simple string datatype property. Once a feature match has been established via the dynamic mapping process and verified as correct, a new rule could be created to automatically map all features with the same value for the *feature type* property directly to the matching target feature, forgoing the computations used in dynamic mapping.

With the shared base ontology established, the framework for exporting CAD feature data from any system into a neutral format is complete. The remaining parts of the export process only require the data from a given CAD system be parsed into instances of the shared base ontology classes. However, to complete the data exchange process, the feature data must still be mapped from the shared base ontology format into the feature classes defined by the local ontology of the target CAD system. This process is described in the following chapter.

CHAPTER 6

FEATURE CLASSIFICATION IN LOCAL ONTOLOGIES

As stated in Chapter 3, classes in OWL are defined not just by their names, but by the properties that can be applied to them and the types of restrictions placed on those properties. Classes in OWL are defined by two sets of properties, those that are necessary, and those that are both necessary and sufficient. By default, all property restrictions to define a class are considered necessary, as the user is specifying that the axiom must evaluate to true in order for an instance of that class to be valid. A set of restrictions can be defined as necessary and sufficient, meaning that not only must these restrictions evaluate to true, but if they do, then that is all that is required for any resource to be inferred as an instance of that class. OWL describes classes with at least one necessary and sufficient condition as a defined class, while those without are described as primitive classes. Therefore, if one were to represent a feature in a local CAD ontology as an OWL class, and properly defined the set of necessary and sufficient conditions based on restrictions of global ontology properties, then any feature that has the set of properties meeting those conditions could automatically be inferred as a member of that class, regardless of the source system.

However, the types of restrictions that can be used to define a class in OWL are somewhat limited, as are the types of inferences that can be made using those rules. OWL can only impose quantifier, cardinality, and hasValue restrictions on properties, or groups of properties connected with either union (∪) or intersection (∩) operators. The quantifier restrictions that can be used are the existential quantifier (∃), which states that a class must have some from the restricted property, and the universal qualifier (∀), which states that the class must only have values from the restricted property. The cardinality restrictions are used to specify a minimum (≥), maximum (≤), or exact (=) cardinality

which indicates a feature must have greater than, less than, or exactly a specific number of instances of a property. However, as stated before, property restriction cannot be chained and they do not support variables. The number must be specified explicitly in the restriction definition. Finally, the hasValue restriction (∋) allows the restriction of properties that only have a specific individual or data value defined by the property. This means that if the hasValue restriction is used on an object property, it must point to a specific individual, and if it is used on a datatype property, it must be a specific data value. Just like the cardinality restrictions, the value used by the hasValue restriction must be a constant stored in the class definition.

Unfortunately, due to the open world assumption used in the OWL language, when most of these restrictions are used as necessary and sufficient conditions, the reasoner will be unable to infer that an individual is a member of the class because the information must not be invalidated by the addition of new information. This means that only the existential qualifier and the minimum cardinality restrictions can be used to infer that an individual is a member of a class. This is understood most easily if you consider the existential qualifier as equivalent to having a minimum cardinality of 1, any additional instances of the property added will never invalidate the minimum cardinality rule once it has been evaluated as true. For example, consider a class that represents a hole feature, which could include a counterbore. One of the necessary and sufficient conditions of this feature would be that every instance of *HoleFeature* must contain a minimum of 1 *createCylindricalSurface* property, but could contain more. This necessary condition could be described as:

$$\text{Cardinality}(\textit{createCylindricalSurface}) \geq 1$$

Clearly, once there is one instance of the property, the axiom will always be true. The existential qualifier works the same way. The same concept could be described as the

property *createsGeometry* ∃(someValuesFrom) *CylindricalSurface* class. Either of these will remain true, regardless of additional data. However, this is not true of the other property restrictions, because additional instances of properties could later invalidate inferences made during reasoning, causing an inferred individual of a class to no longer satisfy the necessary conditions of that class. For example, consider a hypothetical class *SimpleHoleFeature*, which would be a feature that creates a single cylindrical hole in an object, such that it does not allow for a counter bore. One of the necessary and sufficient conditions for that class could be described as:

$$\text{Cardinality}(\textit{createCylindricalSurface}) = 1$$

Clearly, any feature that creates a single cylindrical hole would evaluate true under this axiom, but the feature would **not** be automatically classified as a instance of the *SimpleHoleFeature* class. This is because if another instance of the *createCylindricalSurface* property was added to the feature later, then the axiom would no longer be true. Likewise, the max cardinality, universal qualifier, and hasValue restrictions can also be invalidated by the addition of new property data. Even though addition of information would be impossible in the use of this data as a neutral exchange format, there is no way to bypass these limitations of the open world assumption using OWL alone. For that reason, SWRL rules had to be implemented.

SWRL rules allow new property relationships to be inferred between existing individuals and can explicitly declare an individual to be a member of a class, overcoming two major limitations of OWL. For example, consider an instance of the class *HoleFeature* called “Hole_1”. If a datatype property is added to “Hole_1” that corresponds to the number of cylindrical surfaces it creates, then that number could be used to determine if it is also an instance of *SimpleHoleFeature*. A SWRL rule could test all instances of *HoleFeature* and checked to see if the number of cylindrical surfaces of

each instance equals 1. If this test on “Hole_1” evaluates to true, then the SWRL rule make “Hole_1” an instance of the *SimpleHoleFeature* class. Similarly, new property values can be created using this rule. For example, a SWRL rule could be created that states if an instance of a *HoleFeature* class *hasRadius* equal to a variable “R”, then that *HoleFeature* can have a new property *hasDiameter* and set the value of that property to equal 2·R. Any number of rules that can be evaluated as true or false can be used to construct SWRL rules, and there are various tools available to manipulate the data. This overcomes many of the limitations of the OWL language. However, SWRL shares OWL’s open world assumption, which restricts some reasoning abilities. SWRL only supports monotonic inference, so SWRL rules cannot modify or remove existing information from the ontology, but it can add new information without problem. The monotonicity also means SWRL rules cannot support negation (NOT operator) or disjunction (OR) because new information may invalidate the statement. This means that from a basic logic standpoint, only conjunction (AND) statements can be used. SWRL rules are written as a combination of what are essentially Boolean functions and all must evaluate to true in order for the declaration to be made. An example of the general structure of SWRL rules are explained below.

$$\text{Class1}(?A) \wedge \text{hasProperty}(?A,?B) \wedge \text{swrlb:equal}(?B,4) \rightarrow \text{Class2}(?A)$$

The above shows examples of each of the type Boolean functions. In SWRL, variables are designated with the “?” prefix. The statement “Class1(?A)” stores instances of *Class1* in the variable “?A”, and will evaluate to true so long as at least one instance exists. The second statement “hasProperty(?A,?B)” stores all objects of the *hasProperty* triple with subject “?A” into the variable “?B”. This statement will evaluate as true so long as “?A” has at least one instance of *hasProperty*, and the variable “?B” is not already defined. If “?B” is already defined, then it will only evaluate to true is “?B” is the

object of the *hasProperty* triple. The statement "swrlb:equal(?B,4)" is a SWRL built-in relation. In this case, it evaluates to true if the values within the parentheses are equal. Here, it is clear that in order for the statement to be true, *hasProperty* must either be a float or integer datatype property equal to 4. If all three statements are true, then the inference is made, and the individual stored in variable "?A" is declared as a member of *Class2*. Because all of the statements are connected by AND (^) operators, if one fails, the declaration is not made. The declaration can also be used to create new feature instances and can make multiple inferences at once, for example:

Class1(?A) ^ hasProperty(?A,?B) ^ swrlb:equal(?B,4) → hasProperty2(?A,"Four")

It is possible to work around these limitations due to lack of closed world reasoning support, but as demonstrated above, the SWRL language is not the most convenient method of reasoning. Unfortunately, due to the emphasis on open world reasoning in OWL, it is the only widely available and supported rule language that works with OWL. To count the number of instances of a class to allow closed world reasoning, an integer datatype property was created for each class that needed instance data. These integers must be instantiated and exported by the program that creates the OWL exchange file. This is because SWRL rules are capable of counting instances, but cases with zero instances evaluate as false, and this cannot be resolved in SWRL with only AND operators. By creating integer properties for every class, it is essentially forcing a closed world assumption. These numbers represent how many individuals belong to each class, and because they cannot be modified, new information cannot be added without invalidating them. Because any OWL rule that is necessary and sufficient can be recreated with SWRL, this means that cardinality restrictions are now able to be used with variables, and these variables can be formed from any mathematical combination of property values. Using OR operators is somewhat possible by making multiple rules with

the same outcome but different tests, but such an approach is incredibly impractical. So, while these SWRL rules can be used to demonstrate automatic feature classification based on property restrictions, being limited to writing statements without OR and NOT operators is a big limitation, and any more advanced implementations of this approach would require a more suitable rule engine. Unfortunately, ontologies and tools that use closed world assumptions have not been developed as prominently as those built around OWL, so such a tool would likely need to be built from scratch. Despite the limitations of SWRL, it is still capable of demonstrating this approach is feasible, albeit impractical with current reasoning tools. The next section will describe the general approach to using rules based on feature conceptualization to create a set of necessary and sufficient conditions to define a feature.

Defining and Classifying Local Feature Classes

To allow for automated mapping of any feature stored in the three-branch CAD feature model built using the shared base ontology, the local ontology of the target system should contain a feature hierarchy with the *Feature* class from the shared base ontology as the topmost base class. From there, the local ontology should create subclasses corresponding to different families of features, in whatever organization is the most logical for the target system, effectively creating a comprehensive feature set. There is no correct or single way to do this, but each CAD system has its own internal class hierarchy through which features are defined, so that would be a suitable structure to emulate. Examples of different ways a feature hierarchy could be constructed are represented in the following figures. These figures are not meant to show complete feature hierarchies, they are meant only to show how common features can be conceptually grouped into different families based on different viewpoints. Figure 25 demonstrates one simple class hierarchy based loosely on Pro/Engineer's feature types.

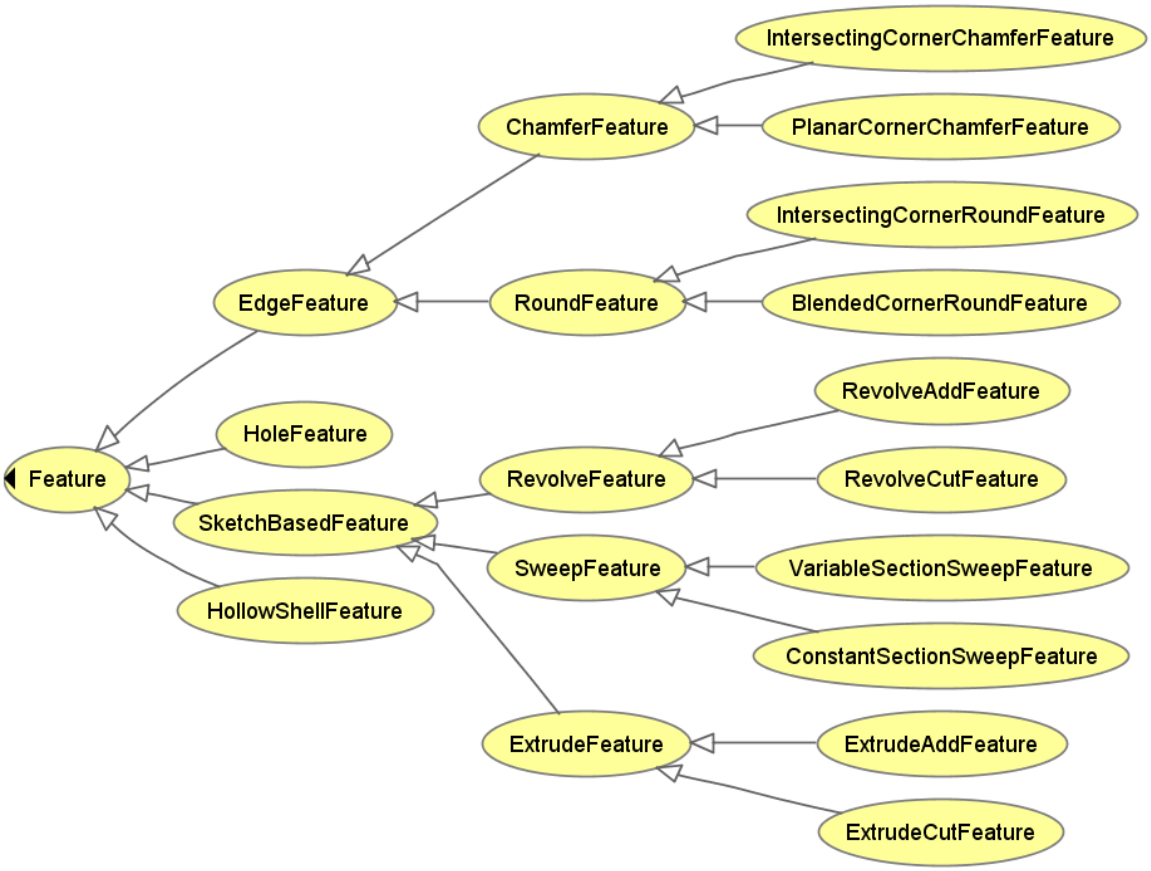

Figure 25: Simple Feature Hierarchy Example Based on Pro/Engineer

In the feature hierarchy presented in Figure 25, the main feature class is divided into groups of features, each representing a specific conceptual grouping. All features that require a sketch are grouped as *SketchBasedFeatures*, and the features that effect edge geometry are grouped as *EdgeFeatures*. The benefits of grouping features in this way is that they follow the structures laid out in the Pro/Engineer API. A simple test can instantly determined whether a feature is a *SketchBasedFeature* or not by checking for instances of the *hasSketch* property. Likewise, any feature that uses reference attributes besides edges could instantly be eliminated as a possible edge feature. However, this feature hierarchy may not be suitable for systems other than Pro/Engineer. For example, Pro/Engineer treats revolve, extrude, and sweep features as separate features, but this may not be the case for all CAD systems. Consider that a constant section sweep takes a

two dimensional sketch and extends it along a designated path. One can consider the extrude and revolve feature as simply special cases of the sweep feature. An extrude can be considered a sweep along a linear path normal to the sketch, while a revolve can be considered a sweep along a circular path. In that case, it is conceivable that a CAD system may want to treat the extrude and revolve features as subclasses of the constant section sweep. Similarly, very simple hole features can be considered a special case of an extrude feature in that sketch is replaced with a circle of given radius. If a CAD system were set up this way, it would be better to use a local ontology with a feature hierarchy similar to the one displayed in Figure 26.

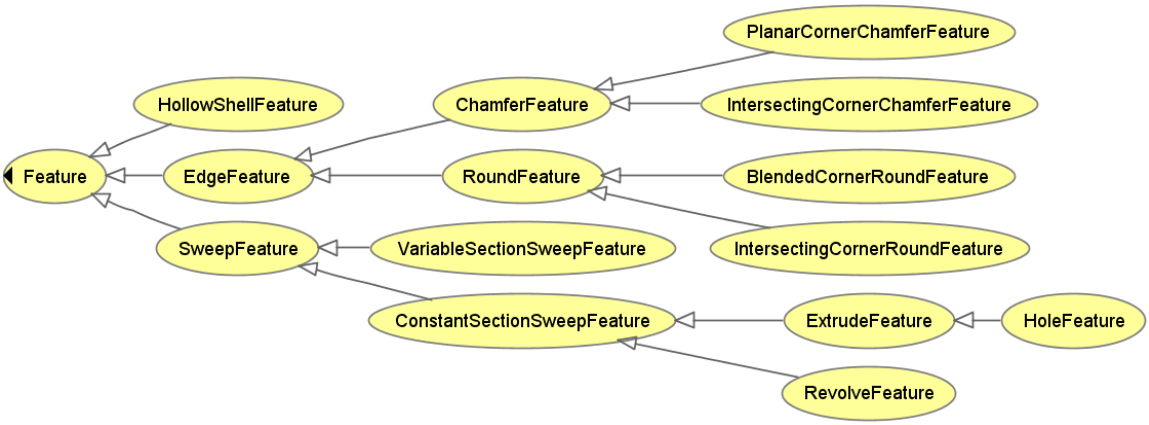

Figure 26: Simple Feature Hierarchy With Extrude and Revolve as Subsets of Sweep

Figure 26 illustrates how a feature hierarchy created in the local ontology should be suited to the specific CAD program it is meant to represent. The groupings presented in Figure 26 clearly would not work with Pro/Engineer, as all sweep features require a path in Pro/Engineer, meaning an extrude could not be a subclass without that information. However, another CAD system, such as an open-source program, may not deem it necessary to code separate extrude and revolve features if the sweep included

easy options to generate linear and circular paths. With this import process, the goal is to determine how best to sort an arbitrary feature into the target system, so the local ontology must be designed to best represent the grouping of feature concepts in the target system. Consider the feature decomposition hierarchy proposed by Dartigues et al. [38] as displayed in Figure 27.

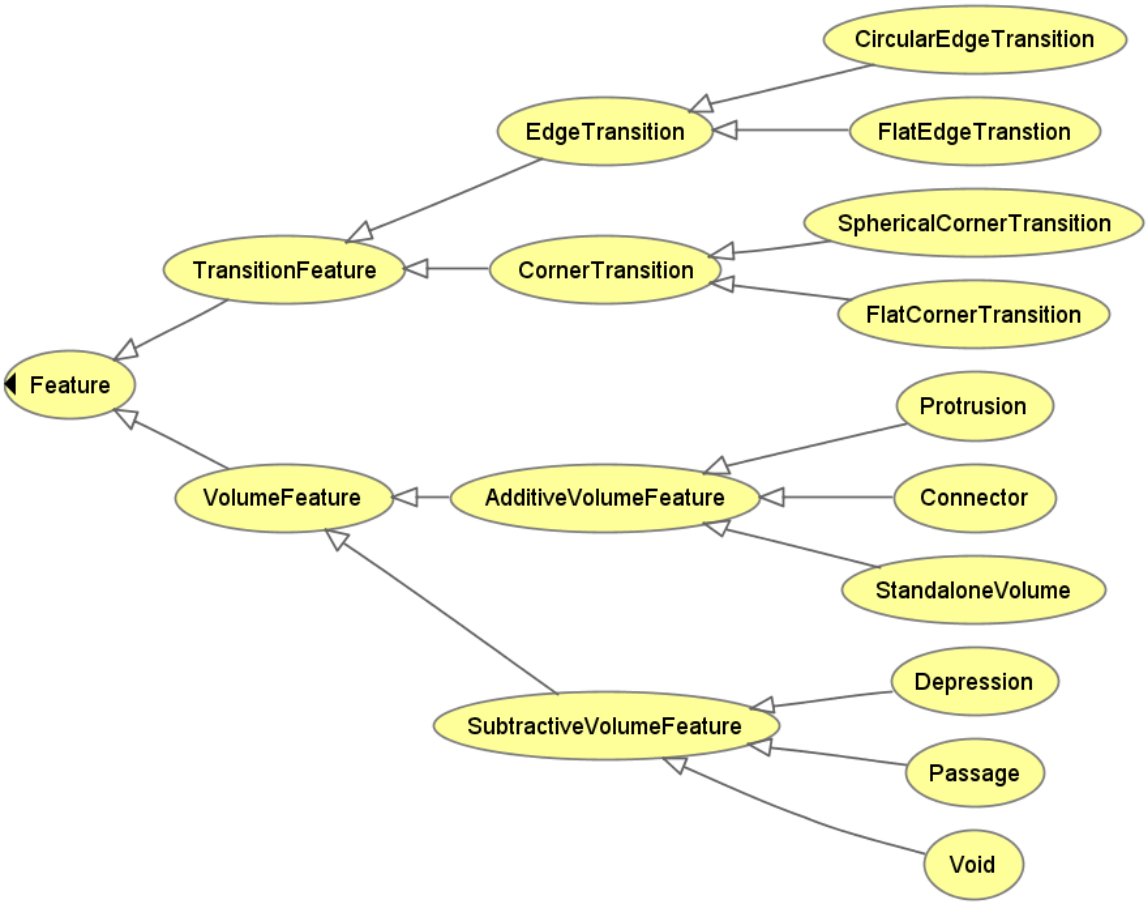

Figure 27: Feature Decomposition Hierarchy Proposed by Dartigues et al. [38]

In the feature decomposition proposed in Figure 27, features are separated into classes which affect volume directly and those that deal with face transitions. This type of hierarchy would work with systems that were built around a constructive solid geometry

(CSG) system. The additive features would be used to represent features that make use of the Boolean union operations, while the subtractive represent Boolean differences and intersections. Again, the use of feature hierarchies is meant solely to facilitate feature mapping, so each local ontology should be tailored to the CAD system it represents.

To classify a feature imported into the target system, a series of rules are used to progress down the feature class hierarchy in the local ontology. For these examples, the hierarchy presented in Figure 25 will be used, however the same general procedure could be used with any class structure. The feature from the source system is imported as a general, unclassified instance of the top level *Feature* class. From there, the feature is tested with SWRL rules to determine which of the next level of subclasses it can belong to. Each subclass is disjoint with each other, so once a feature is classified as a member of one subclass, it is eliminated from the others. For example, the test to see if a feature is a *SketchBasedFeature* would check the number of *hasSketch* properties, if the number is one or greater, it is assigned to the subclass, and no longer tested for subclasses along the other branches.

To further classify the *SketchBasedFeature*, more tests are run on every instance of that class. The test for a basic *ExtrudeFeature* checks every instances of *SketchBasedFeature* class to see if the number of surfaces is equal to the number of sketch entities plus two, the number of planar surfaces equal to the number of line entities plus two, the number of conical surfaces equal the total number of circle and arc entities, and finally that the total number of spline surfaces equals the total number of 2D polyline, spline, ellipse, and conic entities. If all of the above is true, then the *SketchBasedFeature* instance becomes a member of the *ExtrudeFeature* class.

Similarly, the rules to check if it were an instance of the *RevolveFeature* class could check to make sure that specific sketch entities correlated to specific surface types. Line entities that are perpendicular to the axis of rotation would create planar surfaces and all other lines would create conical surfaces. Circular arcs centered on the axis of

rotation would create spherical surfaces while those that are off center would create toroidal surfaces. All other 2D curve entities should result in spline surfaces and all points not on the axis of rotation should create circular curves. Members of the *SweepFeature* class would have to be composed of surfaces that correlated to the swept path, with planar surfaces on both ends and a curve duplicating the path for every 2D point entity in the sketch.

However, these rules are far from being the full set of necessary and sufficient conditions for classifying a feature completely without error. They are a simplification of the much more rigorous process that would be needed for true feature to feature recognition. By only checking the types of surfaces and curves generated and not the exact parameters, such rules could produce false positives. Additionally, conditional statements would need to be introduced for when the feature is interacting with existing geometry. In cases where a union, intersection, or difference is occurs, such rules would no longer be valid, because some surfaces may not be made. However, with enough information, a reasoner could not only check that the right types of surfaces are being created, but also confirm that each surface is defined with the right parameters. However, such rules would require very detailed knowledge of how the feature is defined. In short, the feature recognition rules would have to emulate the internal feature creation and validation rules as defined by the target system. Additional rules would also have to be implemented to allow for differences in the way some geometry is represented. For example, cylindrical and spherical surfaces can be divided into multiple surfaces in different CAD systems. For example, in Pro/Engineer, all full cylindrical surfaces are separated into two halves. A system that subdivides such periodic surfaces a different way would need some method to check that both of those surfaces share the same geometric definition.

Similar tests can be done with any feature, as features are defined by the predictable way they transform user input into geometry based on an internal rule

scheme. Take for example the *EdgeFeature* subclass from Figure 25. The test to determine if a feature is a member of this subclass, rules would determine if all the instances of reference attributes of the feature are members of only the *ReferenceEdge* class. It would also check to make sure that every *ReferenceEdge* selected is accompanied by at least one surface creation and that the selected edge is deleted from the B-Rep by the feature operation. To distinguish between *RoundFeatures* and *ChamferFeatures*, the types of surfaces would again be tested. To be considered an instance of the *RoundFeature* class, there must be a conical surface creation for every linear curve selected, a toroidal surface creation for every circular curve selected, and a spline surface created for every elliptical and interpolated curve selected. Similarly, to be a member of the *ChamferFeature* class, there must be a planar surface creation for every linear curve, a conical surface for every circular curve selected, and ribbon like spline surfaces for all other curve types. To test for the different corner blending options for both of these features, the rules would simply check for additional surfaces. If more planar surfaces are created than linear curves were selected, then the *ChamferFeature* will most likely have corner planes, and if there are spherical or additional spline surfaces in a *RoundFeature*, then there was likely some corner blending. Once the list of possible feature matches have been narrowed down in this way, the mapping process becomes much easier. Again, these serve as simplified examples, as more rigorous tests could be performed to ensure that the curvature of the resulting surfaces match those of the edges, and there would have to be conditional rules for when the geometry of the part causes exceptions, but it is important to again stress that this is knowledge encoded into the programs that create these features in the target system.

The final feature example that will be discussed is a *HoleFeature*. One of the best tests to run on any type of hole feature is one to ensure that all the surfaces created share a common central axis. Regardless of whether the hole has a counter bore, countersink, or tapered end, all conical surfaces created will share the same central axis, and all new

planar surfaces will centered on and normal to that axis. From there, the number and types of surfaces would be tested to determine what kind of hole options were used. These rules may be difficult to express with the limited functionality of SWRL, but clearly such distinctions can be made because holes are easily distinguished in current automatic feature recognition software.

The simple tests described above require very little calculation and are quick, but are only capable of partially determining which feature class are capable of replicating the imported feature and which are not. In an ideal approach, the feature rule testing would be as rigorous and complete as those used to define and validate the features in the target system, thus ensuring that no false conclusions are made. In order to do that, highly detailed knowledge of how the CAD system operates must be known, and would be a costly and laborious task for those without direct access. This is what makes a CAD program unique and is highly proprietary, so it would be very difficult for a third party to obtain without extensive reverse engineering. Therefore, such an ideal approach could likely only be implemented effectively by the CAD vendor themselves. However, unlike the more ad hoc approaches, this approach only requires knowledge of the target system, meaning this could be implemented so long as the CAD companies agree on the shared base ontology format and are willing to develop a hierarchy for their system's features with a series of tests that always evaluate to be true for each type of feature class.

The biggest advantage of this approach is that if the classification rules properly reflect the necessary and sufficient conditions the target feature, it should always work provided the target system has a feature that can replicate the geometry of the source feature. This approach of course will fail when there is no feature in the target system that adequately resembles the shape concept conveyed by the original feature, but such an instance would cause problems for any semantic based approach as well. However, if the local ontology uses a branching hierarchy as in the examples above, this approach has a benefit over the standard mapping processes, because it still classifies the source feature

as the lowest feature subclass that the rules proved was valid, which could narrow down the number of choices for manual mapping. Other times when this approach would fail is when one system uses a compound feature, which would have to be replicated by more than one feature in another system. For example, Solidworks allows users the option to include a draft angle in their extrude feature definition which tapers all sides in by the given angle. Pro/Engineer has no such option in their extrude feature, and to replicate the design intent, the user would have to first extrude the shape, and then use a separate draft feature. Here, the rules based approach would fail, because no version of the Pro/Engineer extrude would be able to create B-Rep in which surfaces were not perpendicular to the sketch plane. In this case, the classification rules stop after listing the feature as an instance of the *SketchBasedFeature* class and would be unable to proceed. A purely semantic based approach may be able to classify it as an *ExtrudeFeature*, but it too would be unable to reconcile the difference in shape type. This also has an advantage over the modifications to the STEP format proposed by Kim et al. [24] in that it is not mandating a single ontology approach, which has limitations as described in Chapter 3. Finally, with the full B-Rep being included in the exchange file, the geometry of any feature that is incapable of being mapped could still be recreated by inserting "dummy" surface that are not defined parametrically, which is an approach taken by some commercial translators.

CHAPTER 7

IMPLEMENTATION

To demonstrate that this approach is viable, a shared global CAD ontology was created using Protégé-OWL. A sample set of feature classes were created to demonstrate creation of a local ontology feature hierarchy. The SWRL rules were implemented in Protégé-OWL's SWRLTab and run using the Jess rule engine [64]. PTC's Pro/Engineer CAD software was used to demonstrate the export of feature and B-Rep data using the API and subsequent parsing into the CAD ontology format. The exported CAD data is then opened in the example local ontology format in Protégé-OWL, where the Jess rule engine is run on the data to automatically classify imported data into the sample feature classes. Figure 28 displays a diagram of the general process with the steps that were implemented within the area bounded by the dashed line. This implementation serves only as a proof of concept, as the export process only supports single feature models, and the rule-based classification is severely limited by the lack of non-monotonic reasoning in OWL and SWRL. Despite these limitations, the classification of extrude features from Pro/Engineer to a sample local ontology of another system demonstrates that this approach works and could be viable given a more robust rule-based reasoning language. The next sections will discuss how the shared global ontology was created in Protégé-OWL, how SWRL rules were implemented, how data was exported from Pro/Engineer, how it was converted into OWL format, and finally how the data was classified.

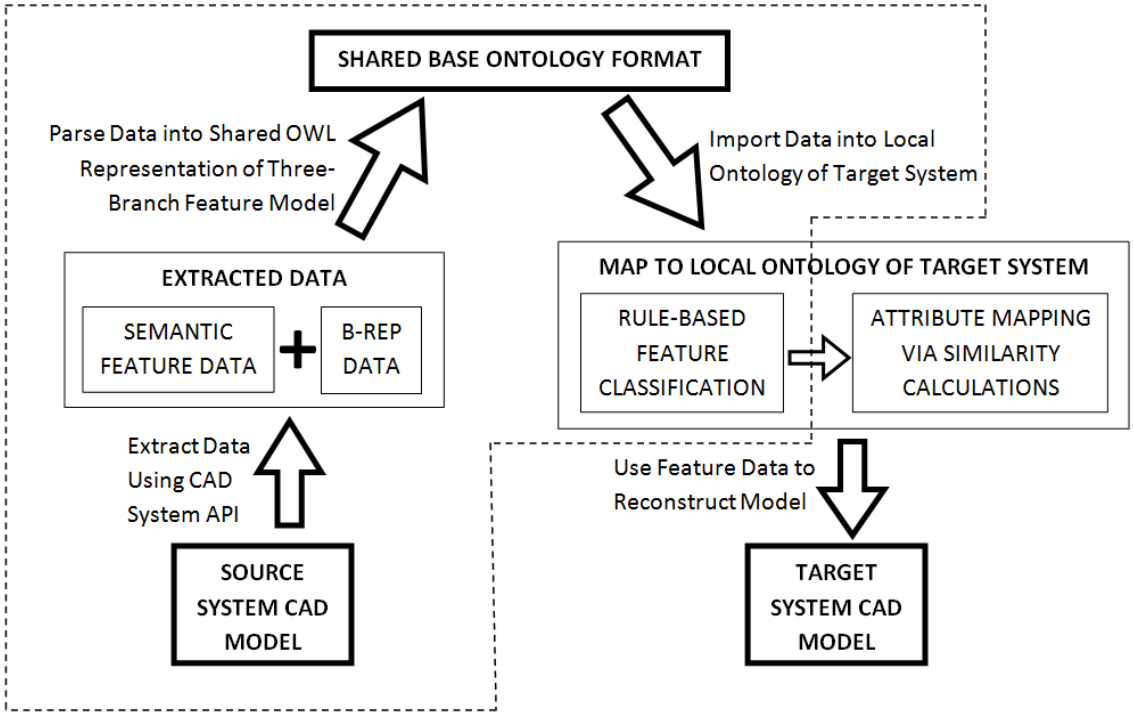

Figure 28: Implemented Parts of the General Approach

Construction of Shared Ontology in Protégé-OWL

Classes and properties in the OWL format are fairly easy to create using the Protégé-OWL interface. As stated in Chapter 5, five top level classes were created. These classes include the *Partfile* and *Feature* classes used to store the exported part and the features of the feature tree, the *ReferenceAttribute* class used to define the various types of reference attributes, the *SketchComponent* class, where various sketch entities, constraints, and dimensions are defined, and finally a *BRepEntity* class, where the different types of B-Rep concepts used by the ACIS format are defined. Figure 29 displays the OWLClasses tab in Protégé-OWL. Classes are defined in the left panel, property restrictions are displayed in the center right panel, and disjoint classes are displayed in the bottom right panel. The data stored in the actual OWL format can be located in Appendix A.

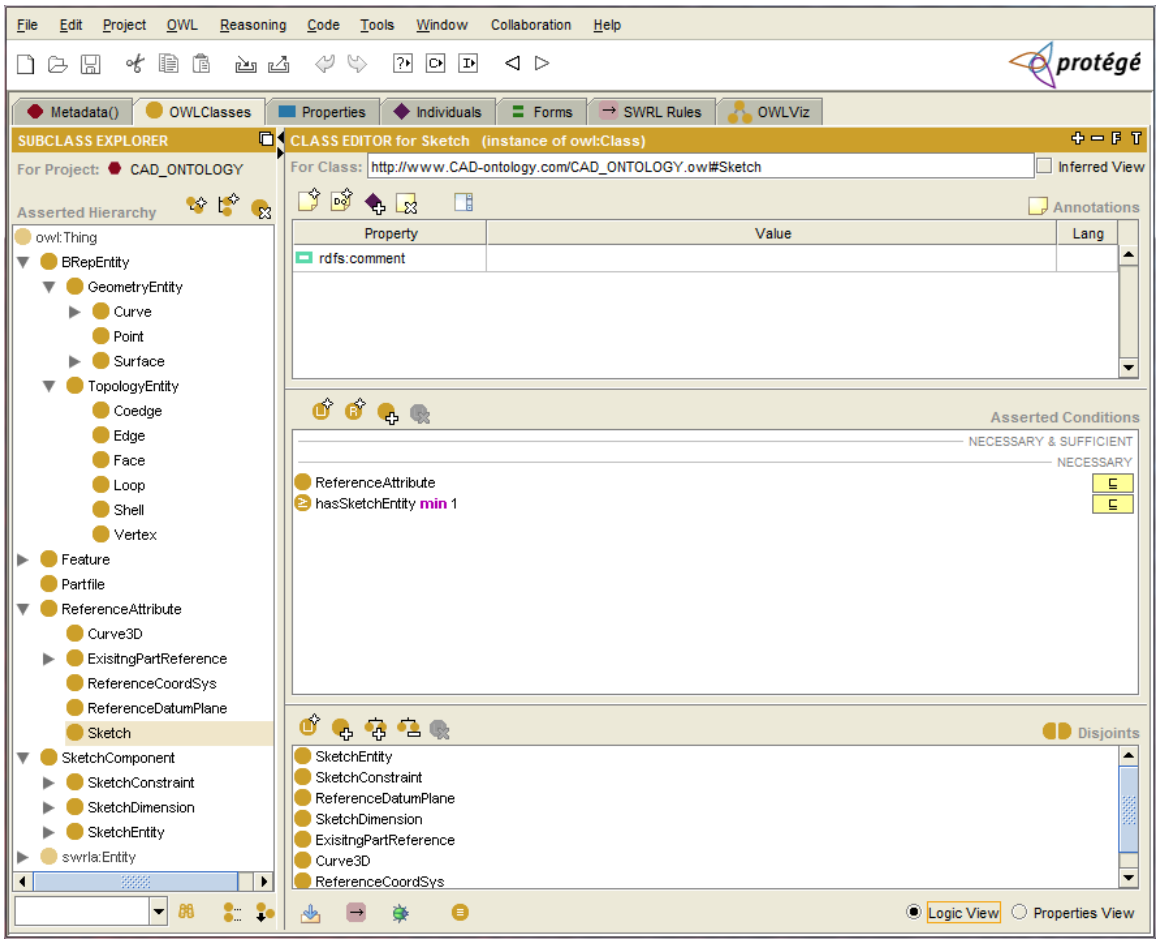

Figure 29: Protégé-OWL OWLClasses Tab

Once the classes are created, the next step is to create the properties that relate the different classes to each other and are used to define the features. This is handled in the Properties tab of Protégé-OWL. Figure 30 displays the object properties view of the Properties tab. On the left, the properties and subproperties are defined. The right panel is used to define the domain and range of the property, and to assign it as functional, inverse functional, symmetric, or transitive. An inverse functional property could also be created, or assigned to an existing property as an inverse. Figure 30 shows the *createsGeometry* property selected, which shows the domain as the *Feature* class and the range as the *GeometryEntity* subclass of the *BRepEntity* class.

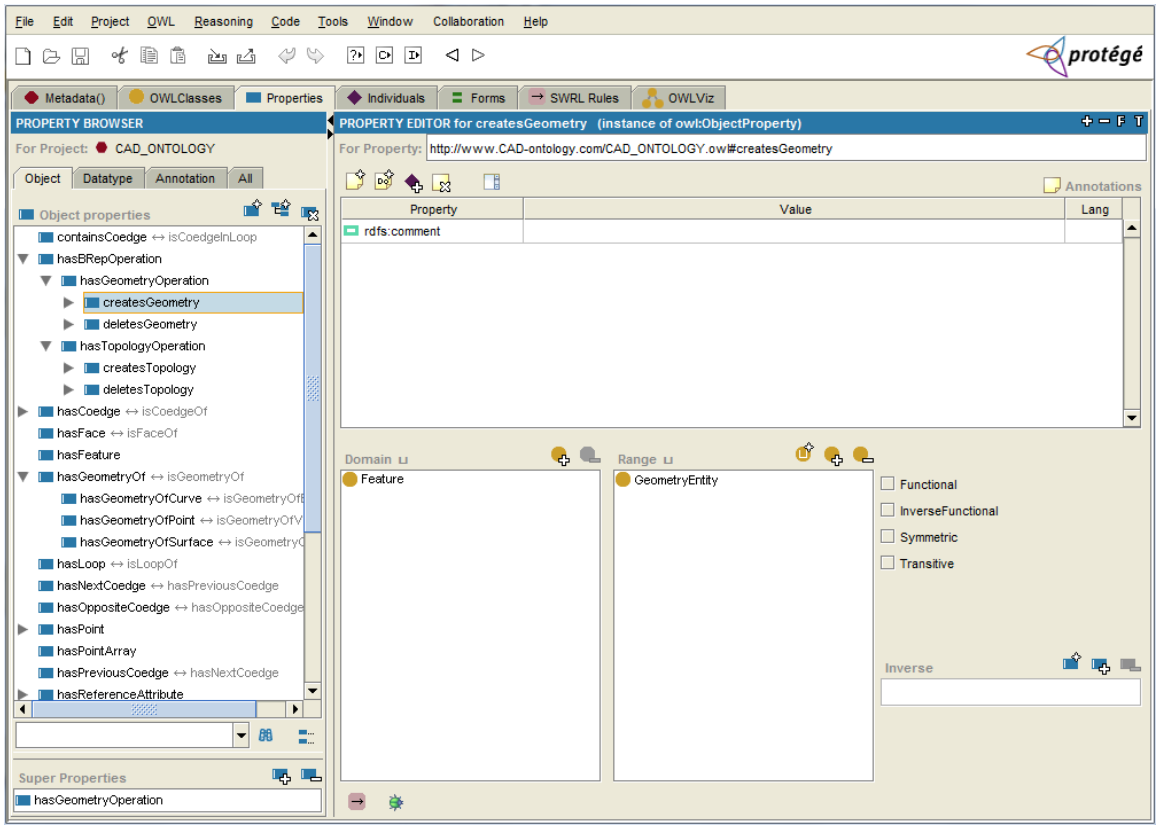

Figure 30: Protégé-OWL Properties Tab, Object Properties View

Figure 31 displays the datatype properties view of the Properties tab. This is similar to the object properties, with a few differences. The range definition area is changed from a class selection menu to a dropdown menu where the specific data type can be chosen. A new box has been added to allow for the designation of a set of allowable values for the datatype property. The datatype property only has the functional property option, because a datatype cannot have an inverse property, nor can it be symmetric or transitive. In Figure 31, the property *hasDepthDimension* is selected. It has the domain of the *Feature* class, and the range of a float value, both of which it inherited from its parent property *hasLengthDimensionAttribute*, which were inherited from the superproperty *hasDimensionAttribute*. No allowed values have been specified, because depth can be any value. Unfortunately, OWL currently does not support ranges on

datatype properties, so if one wanted to specify a range, such as the dimension must be a positive number, it would be impossible with the current language. This has been fixed in OWL2, among other things.

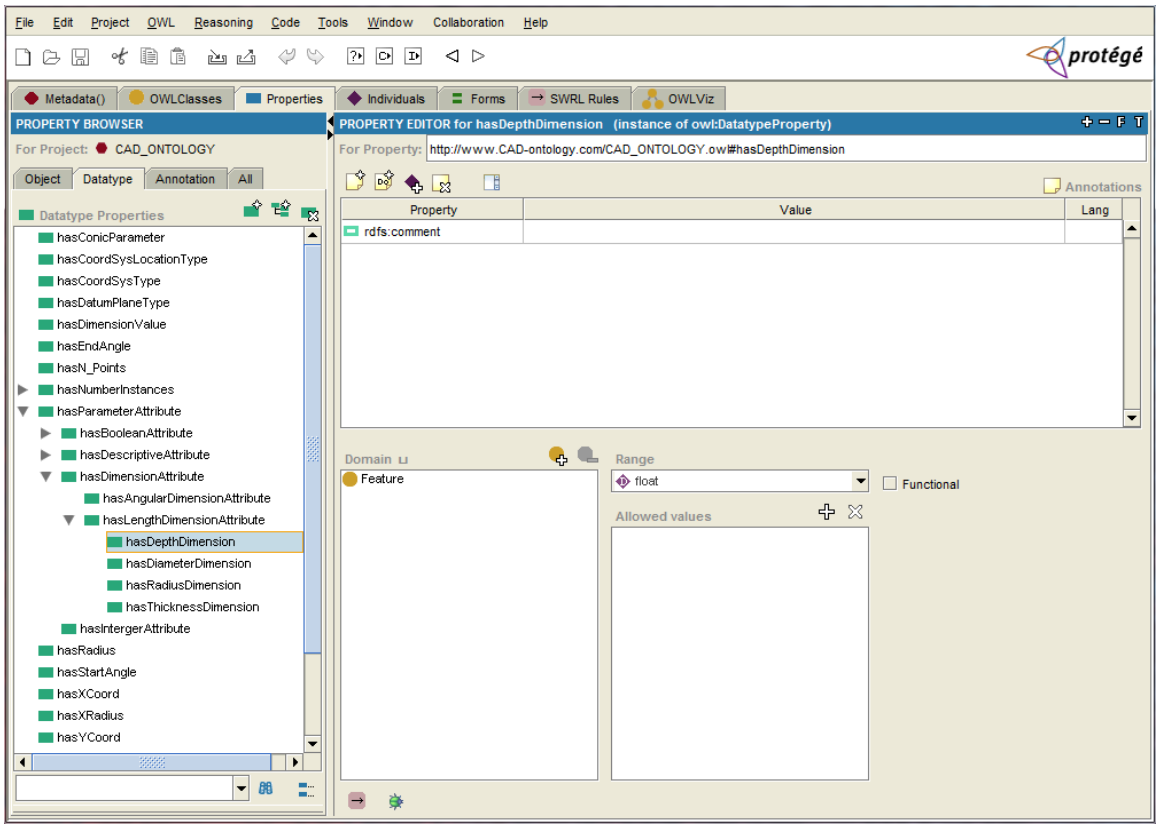

Figure 31: Protégé-OWL Properties Tab, Datatype Properties View

Local Ontology Feature Definitions using SWRL Rules

Feature classes were created in a simplified hierarchy similar to the one previously described in Figure 25. These were created in an ontology that automatically loads and uses the shared base ontology. Due to the limitations of using SWRL, only a few features were defined using SWRL rules. The *SketchBasedFeature*, *EdgeFeature*, and *ExtrudeFeature* classes were the only ones that were defined. The *EdgeFeature*

subclasses could be defined as well, but due to limitations in the export of data from Pro/Engineer, testing them would be impossible. This will be discussed further when describing the export program. Figure 32 illustrates the SWRL tab in Protégé-OWL. Besides rules to define features, additional rules were created to compute the inverse properties of some of the B-Rep properties. This was necessary because when instances are imported from an external file, inverse properties are not automatically created unless previously defined in the imported file. Oddly, the ontology reasoners built into Protégé-OWL do not automatically infer the inverse properties either, which meant the rules had to be added to ensure some of the B-Rep entities were fully defined.

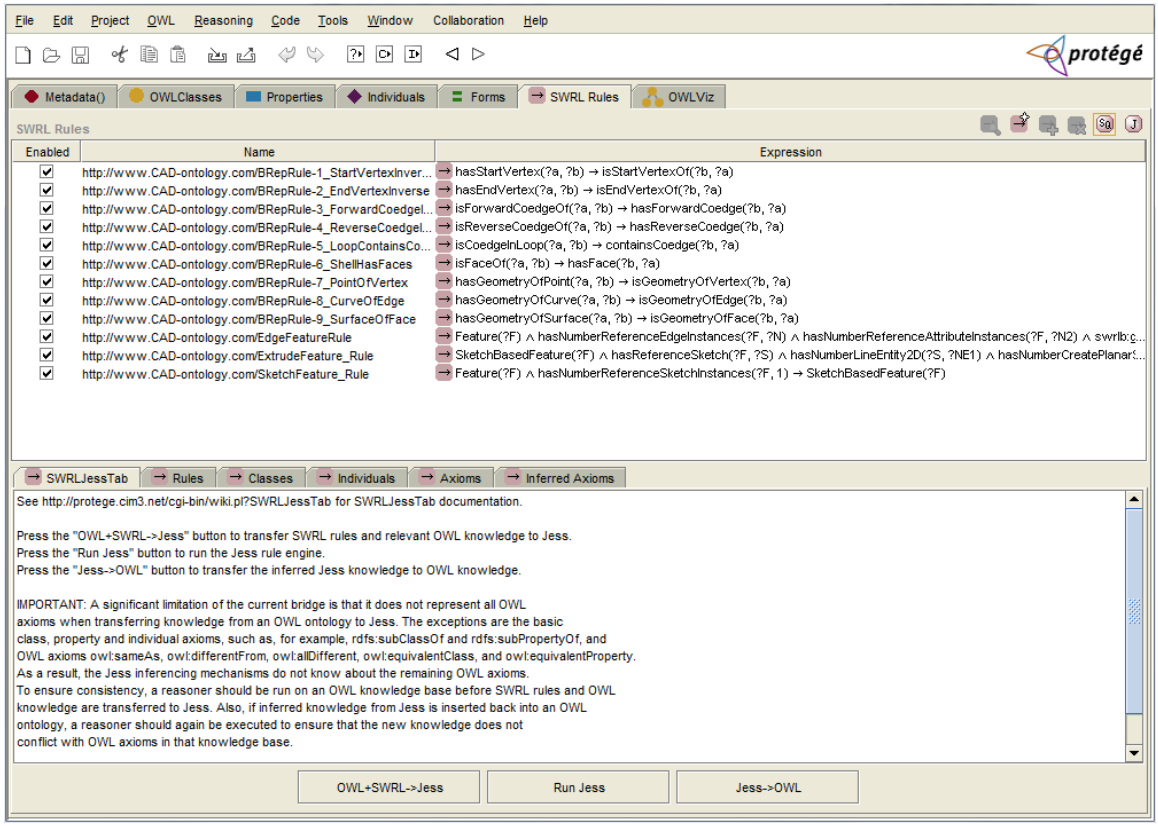

Figure 32: Protégé-OWL SWRL Tab

Extracting Feature Data from Pro/Engineer

Feature data is extracted from parts modeled in Pro/Engineer through use of their API, Pro/TOOLKIT. The exportmodel C++ program was developed and used to export CAD feature data and sketch data as a series of XML files directly from Pro/Engineer. Pro/TOOLKIT provides functions for automatically exporting each feature in the history tree as its own XML file, however, it did not support exporting of sketch data at all. To export sketch data, the exportmodel program originally read each entity, constraint, and dimensions sketches stored within the Pro/Engineer part file and exported that data into a single XML file for each sketch. I modified the code to export the sketch data into the sketch component classes defined in the shared base ontology format. Additional code to create a log file of each feature operation and to export the B-Rep data in ACIS format was also added to the exportmodel program. Unfortunately, the amount of testing that can be done is limited to single feature part models, because the ACIS data that is exported represents only the final geometry of the part. Additional code will have to be developed to determine how to export the resulting B-Rep data after every feature operation in order for parts composed of multiple features to be analyzed in future work. Because the XML files are exported using an existing Pro/TOOLKIT function, it was much easier to use a separate program to read in the XML files and convert the feature data to the shared base ontology format. The code to extract the semantic feature information from the Pro/Engineer XML export file had to be written such that each property needed was read and classified, the data retrieved and converted to the OWL format, and then finally exported with the proper OWL tags. This was a very time intensive task because it required a good deal of trial and error to reverse engineer the meaning of all the tags in the exported XML files, primarily because the Pro/Engineer API guide provided little documentation on how feature data is stored internally. Another major problem was that once the data was deciphered, it had to be converted to OWL format, and in order to do this, ad hoc functions had to be written for every XML tag to convert it into a valid OWL

tag. Figure 33 demonstrates a sample XML file exported directly from Pro/Engineer when viewed in an XML editor. The full code of this example feature can be found in Appendix B. Although the data is somewhat human readable, it is not very well structured. The feature type and feature form properties are both stored as integer data types, when it is clear that the data stored is not. Additionally, the "external surface cut solid type" is given as an integer number, and without access to the Pro/TOOLKIT help files, conveys no useable information. Even with the API help files, much of the data structure is left unexplained, which made deciphering it particularly taxing. Clearly, in its current form, this data is not useful for interoperability purposes.

```
<?xml version="1.0" encoding="UTF-8"?>
<PRO_E_FEATURE_TREE  AppName="Pro/ENGINEER" AppVersion="Wildfire_4.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="ProTKFeature.xsd" type="compound">
  <PRO_E_FEATURE_TYPE type="int">PRO_FEAT_PROTRUSION</PRO_E_FEATURE_TYPE>
  <PRO_E_FEATURE_FORM type="int">PRO_EXTRUDE</PRO_E_FEATURE_FORM>
  <PRO_E_STD_FEATURE_NAME type="wstring">EXTRUDE_1</PRO_E_STD_FEATURE_NAME>
  <PRO_E_EXT_SURF_CUT_SOLID_TYPE type="int">917</PRO_E_EXT_SURF_CUT_SOLID_TYPE>
  <PRO_E_REMOVE_MATERIAL type="int">-1</PRO_E_REMOVE_MATERIAL>
  <PRO_E_IS_SMT_CUT type="int">0</PRO_E_IS_SMT_CUT>
  <PRO_E_SMT_CUT_NORMAL_DIR type="int">0</PRO_E_SMT_CUT_NORMAL_DIR>
  <PRO_E_SKETCHER></PRO_E_SKETCHER>
  <PRO_E_STD_SECTION type="compound">
    <PRO_E_STD_SEC_METHOD type="int">0</PRO_E_STD_SEC_METHOD>
    <PRO_E_SEC_USE_SKETCH type="selection"></PRO_E_SEC_USE_SKETCH>
    <PRO_E_STD_SEC_SELECT type="compound">
      <PRO_E_STD_CURVE_COLLECTION_APPL type="collection">
        <PRO_XML_COLLECTION type="curve">
        </PRO_XML_COLLECTION>
      </PRO_E_STD_CURVE_COLLECTION_APPL>
      <PRO_E_SURF_CHAIN_CMPND type="compound">
        <PRO_E_SURF_CHAIN_METHOD type="int">0</PRO_E_SURF_CHAIN_METHOD>
        <PRO_E_SURF_CHAIN_REF_SURFS type="array">
          <PRO_E_SURF_CHAIN_SURF type="compound">
            <PRO_E_SURF_CHAIN_REF type="selection"></PRO_E_SURF_CHAIN_REF>
          </PRO_E_SURF_CHAIN_SURF>
        </PRO_E_SURF_CHAIN_REF_SURFS>
      </PRO_E_SURF_CHAIN_CMPND>
      <PRO_E_STD_SEC_BLN_VERTS type="selection"></PRO_E_STD_SEC_BLN_VERTS>
    </PRO_E_STD_SEC_SELECT>
```

Figure 33: Sample Section of PRO_EXTRUDE XML File

Converting XML Feature Files into a Single Ontology File

To read XML files and create OWL files more effectively, the open source C++ XML parser, TinyXML was used. TinyXML allows a file to be read, stored, and provides tools to search for specific tags and extract the information. Using the TinyXML functions, I was able to create the ProEtoOWL C++ program. This program reads in the XML feature files, the OWL sketch files, and the ACIS B-Rep files and appends them all into a single global CAD ontology part file stored in the OWL format. To append the feature data, the XML file is read in, the program determines what type of feature it is based on from the "PRO_E_FEATURE_TYPE" tag, and then runs a specific function for that type of feature. Because the reading, parsing, converting, and exporting process had to be written separately for each feature, only the simple extrude feature and the features that define the reference datum and coordinate system were implemented. Once all of the features have been appended to the OWL file, the program then checks the sketches and adds them to the appropriate extrude feature based on the reference given. Finally the B-Rep data is read in, line by line, and each ACIS class type examined, the important information is stored, and then appended to the OWL file as a series of B-Rep entity creations. Any example section of the resulting fully formatted OWL file is presented in Figure 34. A full listing of the exported file from an example feature can be found in Appendix C. Clearly, the data is much easier to read and the properties more directly convey what options and parameters were used to define the features. Additionally, now the data can be examined from within Protégé-OWL, and the SWRL rules can be applied.

```
<CAD:hasFeature>
    <CAD:Feature rdf:ID="EXTRUDE_1">
        <CAD:hasReferenceSketch>
            <CAD:Sketch rdf:ID="S2D0002">
        </CAD:hasReferenceSketch>
        <CAD:hasExtSurfaceType rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Solid</CAD:hasExtSur
        <CAD:removesMaterial rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">False</CAD:removesMat
        <CAD:isThinShellPart rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">False</CAD:isThinShel
        <CAD:hasDepthDimension rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100.000000</CAD:hasDe
        <CAD:hasPrimaryReferenceDatumPlane rdf:resource="#FRONT" />
        <CAD:hasSecondaryReferenceDatumPlane rdf:resource="#RIGHT" />
        <CAD:hasNumberReferenceSketchInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</CAD:
        <CAD:createsShell>
            <CAD:Shell rdf:ID="BRepID-3" />
        </CAD:createsShell>
        <CAD:createsFace>
            <CAD:Face rdf:ID="BRepID-4">
                <CAD:hasLoop rdf:resource="#BRepID-6" />
                <CAD:isFaceOf rdf:resource="#BRepID-3" />
                <CAD:hasGeometryOfSurface rdf:resource="#BRepID-7" />
            </CAD:Face>
        </CAD:createsFace>
        <CAD:createsFace>
            <CAD:Face rdf:ID="BRepID-5">
                <CAD:hasLoop rdf:resource="#BRepID-9" />
                <CAD:isFaceOf rdf:resource="#BRepID-3" />
                <CAD:hasGeometryOfSurface rdf:resource="#BRepID-10" />
            </CAD:Face>
        </CAD:createsFace>
```

Figure 34: Sample Section of OWL Exchange File

Classification with SWRL Rules in Protégé-OWL

When the OWL partfile is opened in the local CAD ontology file in Protégé-OWL, the feature has not been classified and it is merely an instance of the *Feature* parent class. To classify it, the SWRL Rules must be run using the Jess rule engine. The test to see if a feature is an instance of the *SketchBasedFeature* class simply tests to see if the feature has one reference sketch associated with it. If this is true, then the feature becomes an instance of the *SketchBasedFeature* class. This test in clearly trivial, because it is very simple to check if a feature was created using sketch using the shared based ontology language. Likewise, the test to determine if an instance of the *EdgeFeature* class is similarly simple. This rule checks to see if all reference attributes are of type *ReferenceEdge*. The test to determine if an individual of the *SketchBasedFeature* class is a member of the *ExtrudeFeature* class is slightly more difficult to express. As stated in the previous chapter, this test compares the number of 2D sketch entities to the number of

specific surface types. For this test, I assumed the extrude feature was being used as the first and only feature of the file, indicating that only B-Rep creation operations would be performed, and the exact number of surface creations could be predicted. The test for the *ExtrudeFeature* as written in Protégé is displayed in Figure 35. For clarity, the rule will also be expressed in Table 3. Note that due to the open world assumption of OWL, the rules are nothing more than a series of AND logical conjunctions, and if any part of the rule fails, the inference will not apply. If the feature were interacting with an existing model, these rules would no longer hold true. Such rules would have to included exceptions and equivalent tests that incorporated the feature creation and validation rules that are employed when features are created that interact with existing B-Rep. Currently, it is unclear how such rules could be modeled, or if it would be possible at all without non-monotonic reasoning.

Full Representation of Extrude Feature SWRL Rule:

SketchBasedFeature(?F) ∧ hasReferenceSketch(?F, ?S) ∧
hasNumberLineEntity2D(?S, ?NE1) ∧
hasNumberCreatePlanarSurfaceInstances(?F, ?NS1) ∧ swrlb:add(?TNE1, ?NE1, 2) ∧
swrlb:equal(?NS1, ?TNE1) ∧ hasNumberCircleEntity2D(?S, ?NE2) ∧
hasNumberArcEntity2D(?S, ?NE3) ∧
hasNumberCreateConicalSurfaceInstances(?F, ?NS2) ∧
swrlb:add(?TNE2, ?NE2, ?NE2, ?NE3) ∧ swrlb:equal(?NS2, ?TNE2) ∧
hasNumberEllipseEntity2D(?S, ?NE4) ∧ hasNumberConicEntity2D(?S, ?NE5) ∧
hasNumberSplineEntity2D(?S, ?NE6) ∧ hasNumberPolylineEntity2D(?S, ?NE7) ∧
hasNumberCreateSplineSurfaceInstances(?F, ?NS3) ∧
swrlb:add(?TNE3, ?NE4, ?NE4, ?NE5, ?NE6, ?NE7) ∧ swrlb:equal(?NS3, ?TNE3) ∧
swrlb:add(?TNE, ?TNE1, ?TNE2, ?TNE3) ∧ swrlb:add(?TNS, ?NS1, ?NS2, ?NS3) ∧
swrlb:equal(?TNS, ?TNE) → ExtrudeFeature(?F)

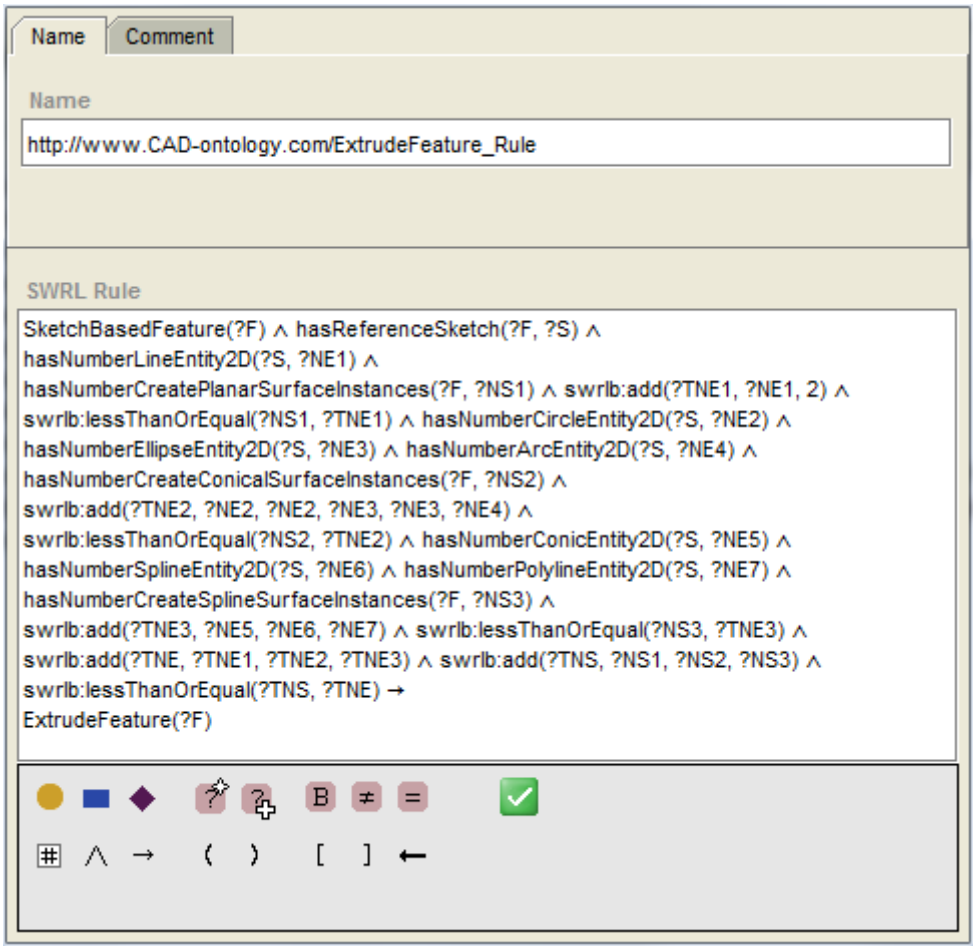

Figure 35: ExtrudeFeature SWRL Rule

Table 3: ExtrudeFeature SWRL Rule Overview

Test #	Statement	Comments
1	SketchBasedFeature(?F)	Assigns all instances of SketchBasedFeature to variable “F”
2	hasReferenceSketch(?F, ?S)	Assigns sketch of feature “F” to variable “S”
3	hasNumberLineEntity2D(?S, ?NE1)	Stores number of line entities in “S” to variable “NE1”
4	hasNumberCreatePlanarSurfaceInstances(?F, ?NS1)	Stores number of plane surfaces created by “F” to variable “NS1”
5	swrlb:add(?TNE1, ?NE1, 2)	Adds 2 to “NE1”, saves it as variable “TNE1”
6	swrlb:equal(?NS1, ?TNE1)	Checks to make sure the number of planar surfaces equals number of lines + 2
7	hasNumberCircleEntity2D(?S, ?NE2)	Sets number of circles to variable “NE2”
8	hasNumberArcEntity2D(?S, ?NE3)	Sets number of circular arcs to variable “NE3”
9	hasNumberCreateConicalSurfaceInstances(?F, ?NS2)	Sets number of conical surfaces to “NS2”
10	swrlb:add(?TNE2, ?NE2, ?NE2, ?NE3)	Adds 2 times the number of circles to the number of arcs, saves to “TNE2”
11	swrlb:equal(?NS2, ?TNE2)	Checks that number of arcs and circles equals number of conic surfaces
12	hasNumberEllipseEntity2D(?S, ?NE4)	Number of ellipses = “NE4”
13	hasNumberConicEntity2D(?S, ?NE5)	Number of conics = “NE5”
14	hasNumberSplineEntity2D(?S, ?NE6)	Number of splines = “NE6”
15	hasNumberPolylineEntity2D(?S, ?NE7)	Number of polylines =“NE7”
16	hasNumberCreateSplineSurfaceInstances(?F, ?NS3)	Number of spline surfaces created store as “NS3”
17	swrlb:add(?TNE3, ?NE4, ?NE4, ?NE5, ?NE6, ?NE7)	Adds 2 times the number of ellipses to conics, splines, and polylines, sets to “TNE3”
18	swrlb:equal(?NS3, ?TNE3)	Checks the number of spline surfaces equals “TNE3”
19	swrlb:add(?TNE, ?TNE1, ?TNE2, ?TNE3)	Adds the total number of entities + 2, saves as “TNE”
20	swrlb:add(?TNS, ?NS1, ?NS2, ?NS3)	Adds the total number of surfaces, saves as “TNS”
21	swrlb:equal(?TNS, ?TNE)	Confirms totals are equal
Result	**ExtrudeFeature(?F)**	**Makes “F” member of class ExtrudeFeature**

The comments in Table 3 clearly convey the process by which the SWRL rules determine if an instance of *SketchBasedFeature* is also an instance of *ExtrudeFeature*. The only thing worth noting further is that the number of surfaces created by circle and ellipse entities had to be doubled, because Pro/Engineer does not save 360 degree surfaces. Surfaces that wrap completely around, such as cylinders, are always divided into two halves. This could be problematic if other CAD systems do not follow the same convention, and it may be necessary to create a way to determine when adjacent faces share the same surface geometry. Also, while this example illustrates that SWRL rules can be used to classify features without relying solely on semantic data, it also demonstrates how difficult it is to work with SWRL. Needing that many different commands to determine if one variable is less than or equal to another variable plus two is not very efficient. In a more realistic example, the rule testing would be far more rigorous, and could not be implemented with this tool. It would likely be impossible to run the kinds of test that would be required for more advanced feature mapping using SWRL and OWL as they currently exist. However, since the SWRL rules are only making direct assertions about the class type if the rule evaluates to true, it should not be difficult to run these tests in a program that supports non-monotonic inferencing, so longs as it can read and interpret OWL properties. However, such a program is not readily available, and building one for this application would require sufficient work to base a second research thesis upon. This implementation may not show a full translation from one CAD system to another, but it does demonstrate that it is possible to classify features without having to rely solely upon semantic definitions. It shows that rule-based classification of features is sound, but hard to implement using currently available ontology tools. Full translation would require features from the target system to be expressed in a local ontology where the feature classes are defined with complete sets of necessary and sufficient classification rules. Once the features have been classified into

their appropriate classes, semantic matching based on similarity calculations would need to be used to resolve any unmatched attributes within class definition. Finally, once the data is fully mapped to the local CAD ontology, another program would be needed to take that data and rebuild it in the target CAD system. Similar work in feature matching based on semantic similarity has been done with the hybrid semantic feature model [42,43] using RDFS , so an approach based on the more expressive OWL language could include matching based on property type similarities would be more robust and should be possible with further development.

CHAPTER 8

SUMMARY AND FUTURE WORK

In this thesis, existing tools to enable data exchange among heterogeneous CAD systems are researched. Background on current geometry based interoperability solutions and feature based approaches was given. Previous ontology based approaches focused mostly on resolving semantic heterogeneity through use of a shared language and lacked a sufficient means to handle structural and conceptual differences without requiring ad hoc mapping processes. Previous work by our research group attempted to calculate semantic similarity based on name aliasing and type matching, but this method was found to be insufficient when encountering features with ambiguous semantics and similar graph structures. To overcome these limitations, a new approach was proposed which made use of the resulting boundary representation of a feature and expressed how geometric data could be related to the input attributes to embody the conceptualization of a feature operation through a series of rules. These rules would not only represent the convention by which humans visualize a feature, but also express the rules that the feature must follow to be a valid representation of that concept. By incorporating such rules into the mapping process as a classification step, the overall approach integrates techniques from both feature recognition and feature mapping research fields. This approach attempts to bridge the gap between newer, semantic based translation methods and more traditional geometry based feature recognition methods to provide a more comprehensive exchange format.

To achieve the goal of comprehensive representation, a new CAD ontology built off a hybrid ontology approach was proposed. This CAD ontology created a shared base vocabulary by which all CAD systems could exchange common data through a shared syntax and structure. In this hybrid ontology approach, local ontologies are used to define

the feature sets of the individual CAD systems in terms of the shared base ontology. To improve consistency in feature definitions, a three-branch CAD feature model was also proposed. This model would store features as combinations of the reference attributes by which it was related to existing data in the part file, parameter attributes which are defined in the feature definition, and by the B-Rep operations by which it alters the existing topology and geometry. The shared base ontology was built in the OWL language using the ontology editing tool Protégé-OWL to express classes of data designed to be universal to CAD programs by using established concepts. Limitations to OWL's class definition tools were described and overcome through use of SWRL rules.

Finally, description of implementation of the ontology was given. The shared base ontology for the CAD domain was created, as was a small local ontology to test feature classification. The creation of the three-branch CAD feature model and the CAD ontology was an important achievement because it provided a new framework that fully encapsulates both the defining attributes and resulting geometry of any CAD form feature in a uniform and unambiguous manner. By itself, such a framework would provide a useful neutral exchange format for types of direct mapping techniques, but by using the OWL language to create the ontology, additional expressiveness and reasoning abilities were added. To make use of these reasoning abilities, a set of simple SWRL rules was created to act as a proof-of-concept test and to demonstrate that an arbitrary feature may be classified reasonable and quickly without having to rely on similarity calculations. Lack of closed world and non-monotonic reasoning tool support in OWL and SWRL prevented more comprehensive classification rules from being implemented. However, the field of ontology research is rapidly advancing, and increased demand for more expressive languages may yield such tools in the near future. Computer programs to extract feature data from Pro/Engineer and to convert feature, sketch, and B-Rep data into the shared ontology format were developed and implemented. The approach to define a feature in terms of expected geometry at a class level, and then use that definition to

classify a feature without relying on semantic similarity calculations was tested and verified. By using such classification, features that convey the same shape concept can be related even if they have semantic or structural dissimilarities, which addresses a problem common amongst semantic mapping approaches.

To extend this work, several issues and implementation problems can be further addressed. One of the biggest problems is the inability to store B-Rep data after every feature operation, which currently limits the part model to a single feature. A way to store B-Rep after each feature creation in the construction history tree needs to be implemented in our Pro/Engineer export program. This may require a reversed construction history approach, where the B-Rep is exported for the final part, then the last feature is suppressed, and then B-Rep is exported again, until the process returns to the first feature. In a related problem, a means to correlate the B-Rep output at each feature creation to the resource identifiers used by other features as reference attributes is also needed. Programs like Pro/Engineer use their own internal resource identifiers which do not correlate to any of the B-Rep data that is exported. If multiple features are supported, then features that require geometry or topology entities created by previous features must have a way to know which existing entity is being referenced. Possible solutions to this may include creating a persistent naming convention for such entities. The API will have to be examined in greater detail to determine if a method to identify XML references exists. It may be necessary to assign an alias that correlates the reference ID in XML to the B-Rep object that has been calculated as having equivalent geometry.

Additional work should also be done to test the efficacy of this approach. Local feature ontologies should also be created for a different CAD system in the future and full translation between two systems should be demonstrated with this approach. The Pro/Engineer export code could be improved to export more features into OWL format and to store more data in the created OWL file. More research needs to be done to compare and contrast the effectiveness of this approach versus a purely semantic or ad

hoc approach. Tools that allow the use of local closed world assumptions and non-monotonic reasoning with OWL will also have to be implemented to advance research on this approach. As described before, the open world assumption only benefits databases where new information is being added. When using an ontology as an exchange format, all information that will ever be used in included in the file, so there is no fear of invalidation due to new information. To create a more comprehensive feature classification method, more complex logical operators than those currently available through OWL and SWRL will be required.

Finally, the shared base ontology could be improved with subclasses and properties that more completely describe concepts common amongst different CAD systems. For example, the B-Rep surface entity classes that are based on the ACIS data could be more expressive. ACIS does not have a separate class for cylindrical surfaces, which should be considered a subclass of the conic surface. Similarly, there is no specific class for circular curves, which should be considered a subset of the elliptical curve class. It would also be useful to distinguish between different types of spline surfaces, perhaps by using a set of subclasses based on the order of the curves which define the surface. The work done in this thesis is only the start of an effective approach to interoperable feature exchange. The limitations of this approach lie mostly in the researchers understanding of each CAD system when creating the local ontologies and the programming abilities needed to extract the necessary information using a system's API.

APPENDIX A

CAD DOMAIN SHARED BASE ONTOLOGY OWL FILE

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:swrlx="http://swrl.stanford.edu/ontologies/built-ins/3.3/swrlx.owl#"
    xmlns:swrlm="http://swrl.stanford.edu/ontologies/built-ins/3.4/swrlm.owl#"
    xmlns="http://www.CAD-ontology.com/CAD_ONTOLOGY.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#"
  xml:base="http://www.CAD-ontology.com/CAD_ONTOLOGY.owl">
  <owl:Ontology rdf:about="#">
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl"/>
    <owl:imports rdf:resource="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl"/>
  </owl:Ontology>
  <owl:Class rdf:ID="CylindricalSurface">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="ConicalSurface"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="ConstructionLineEntity2D">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="SketchEntity"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="PointEntity2D"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="EllipseEntity2D"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="ArcEntity2D"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="ConicEntity2D"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="CircleEntity2D"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="SplineEntity2D">
    <rdfs:subClassOf>
```

```
    <owl:Class rdf:about="#SketchEntity"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="InterpolatedCurve">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Curve"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="EdgeFeature">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Feature"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="ReferenceEdge"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasReferenceAttribute"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="SketchBasedFeature"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="ParameterSpaceCurve">
  <rdfs:subClassOf rdf:resource="#InterpolatedCurve"/>
</owl:Class>
<owl:Class rdf:ID="PolylineEntity2D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchEntity"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="IntersectingCornerRoundFeature">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="RoundFeature"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ReferenceAttribute">
  <owl:disjointWith>
    <rdf:Description rdf:about="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#Entity">
      <owl:disjointWith>
        <owl:Class rdf:ID="Partfile"/>
      </owl:disjointWith>
      <owl:disjointWith>
        <owl:Class rdf:ID="SketchComponent"/>
      </owl:disjointWith>
      <owl:disjointWith>
        <owl:Class rdf:ID="BRepEntity"/>
      </owl:disjointWith>
      <owl:disjointWith rdf:resource="#ReferenceAttribute"/>
      <owl:disjointWith>
        <owl:Class rdf:about="#Feature"/>
      </owl:disjointWith>
    </rdf:Description>
```

```
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Partfile"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#BRepEntity"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Feature"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#SketchComponent"/>
  </owl:disjointWith>
 </owl:Class>
 <owl:Class rdf:about="#EllipseEntity2D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchEntity"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#ConicEntity2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="LineEntity2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ArcEntity2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#PointEntity2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#CircleEntity2D"/>
  </owl:disjointWith>
 </owl:Class>
 <owl:Class rdf:ID="LineToLineDimension2D">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="SketchDimension"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="LineToPointDimension2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="isDimensionOfEntity"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#LineEntity2D"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="DiameterDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="ArcAngleDimension2D"/>
```

```
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="LineLengthDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="PointToPointDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="AngleBetweenLinesDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="RadiusDimension2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ArcToArcHorizTangentDimension2D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchDimension"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#ArcEntity2D"/>
            <owl:Class rdf:about="#CircleEntity2D"/>
          </owl:unionOf>
        </owl:Class>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#SketchBasedFeature">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Feature"/>
  </rdfs:subClassOf>
```

```
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
    <owl:onProperty>
     <owl:ObjectProperty rdf:ID="hasReferenceSketch"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#EdgeFeature"/>
 </owl:Class>
 <owl:Class rdf:about="#SketchComponent">
  <owl:disjointWith rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#Entity"/>
  <owl:disjointWith>
   <owl:Class rdf:about="#Partfile"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:about="#BRepEntity"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#ReferenceAttribute"/>
  <owl:disjointWith>
   <owl:Class rdf:about="#Feature"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:ID="Sketch"/>
  </owl:disjointWith>
 </owl:Class>
 <owl:Class rdf:about="#ConicEntity2D">
  <owl:disjointWith>
   <owl:Class rdf:about="#PointEntity2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#EllipseEntity2D"/>
  <owl:disjointWith>
   <owl:Class rdf:about="#CircleEntity2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:about="#LineEntity2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:about="#ArcEntity2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
   <owl:Class rdf:about="#SketchEntity"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="Face">
  <owl:disjointWith>
   <owl:Class rdf:ID="Shell"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:InverseFunctionalProperty rdf:ID="hasLoop"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
```

```
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="TopologyEntity"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="Point"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Surface"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Coedge"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Vertex"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Curve"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Loop"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasGeometryOfSurface"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="isFaceOf"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="Edge"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="PointOnEntityConstraint2D">
  <owl:disjointWith>
    <owl:Class rdf:ID="ParallelEntitiesConstraint2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="SketchConstraint"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="VerticalEntityConstraint2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
```

```
    <owl:Class rdf:ID="EqualRadiiConstraint2D"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="SamePointConstraint2D"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >2</owl:cardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="isConstraintOfEntity"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:ID="PerpendicularEntitiesConstraint2D"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="HorizontalEntityConstraint2D"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:hasValue>
      <owl:Class rdf:about="#PointEntity2D"/>
     </owl:hasValue>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:ID="EqualSegmentsConstraint2D"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="ColinearConstraint2D"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="TangentEntitiesConstraint2D"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#ParallelEntitiesConstraint2D">
   <owl:disjointWith>
    <owl:Class rdf:about="#EqualSegmentsConstraint2D"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#VerticalEntityConstraint2D"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#EqualRadiiConstraint2D"/>
   </owl:disjointWith>
   <owl:disjointWith rdf:resource="#PointOnEntityConstraint2D"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >2</owl:cardinality>
     <owl:onProperty>
```

```
      <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:about="#SamePointConstraint2D"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#PerpendicularEntitiesConstraint2D"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchConstraint"/>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:about="#HorizontalEntityConstraint2D"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#TangentEntitiesConstraint2D"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
     </owl:onProperty>
     <owl:allValuesFrom>
      <owl:Class rdf:about="#LineEntity2D"/>
     </owl:allValuesFrom>
    </owl:Restriction>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:about="#ColinearConstraint2D"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#ReferenceEdge">
   <rdfs:subClassOf>
    <owl:Class rdf:ID="ExisitngPartReference"/>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:ID="ReferenceShell"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="ReferenceFace"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="ReferenceVertex"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="ConicParameterDimension2D">
   <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchDimension"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
     </owl:onProperty>
```

```
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#ConicEntity2D"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Feature">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasBRepOperation"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasReferenceAttribute"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#Entity"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Partfile"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#BRepEntity"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#ReferenceAttribute"/>
  <owl:disjointWith rdf:resource="#SketchComponent"/>
</owl:Class>
<owl:Class rdf:about="#ReferenceVertex">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#ExisitngPartReference"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#ReferenceEdge"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#ReferenceShell"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ReferenceFace"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#BRepEntity">
```

```
  <owl:disjointWith rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#Entity"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Partfile"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#ReferenceAttribute"/>
  <owl:disjointWith rdf:resource="#Feature"/>
  <owl:disjointWith rdf:resource="#SketchComponent"/>
 </owl:Class>
 <owl:Class rdf:about="#ColinearConstraint2D">
  <owl:disjointWith>
    <owl:Class rdf:about="#HorizontalEntityConstraint2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#PerpendicularEntitiesConstraint2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#SamePointConstraint2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#ParallelEntitiesConstraint2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#LineEntity2D"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#EqualSegmentsConstraint2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchConstraint"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#VerticalEntityConstraint2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#PointOnEntityConstraint2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#EqualRadiiConstraint2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#TangentEntitiesConstraint2D"/>
  </owl:disjointWith>
 </owl:Class>
 <owl:Class rdf:ID="ReferenceDatumPlane">
```

```
  <owl:disjointWith>
    <owl:Class rdf:ID="ReferenceCoordSys"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Sketch"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Curve3D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ExisitngPartReference"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#ReferenceAttribute"/>
</owl:Class>
<owl:Class rdf:about="#LineEntity2D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchEntity"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#PointEntity2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#EllipseEntity2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#ArcEntity2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#ConicEntity2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#CircleEntity2D"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class>
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#LineEntity2D"/>
    <owl:Class rdf:about="#ConicEntity2D"/>
  </owl:unionOf>
</owl:Class>
<owl:Class rdf:ID="B-SplineCurve">
  <rdfs:subClassOf rdf:resource="#InterpolatedCurve"/>
</owl:Class>
<owl:Class rdf:ID="PlanarSurface">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Surface"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="ToroidalSurface"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ConicalSurface"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="SplineSurface"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="SphericalSurface"/>
  </owl:disjointWith>
</owl:Class>
```

```
<owl:Class rdf:ID="EllipseYRadiusDimension2D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchDimension"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#EllipseEntity2D"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#ReferenceShell">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#ExisitngPartReference"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#ReferenceEdge"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#ReferenceFace"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#ReferenceVertex"/>
</owl:Class>
<owl:Class rdf:ID="LinearCurve">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Curve"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PointToPointHorizDimension2D">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#PointEntity2D"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchDimension"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineLengthDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineToPointDimension2D"/>
```

```
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#AngleBetweenLinesDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#RadiusDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ArcAngleDimension2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#DiameterDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#LineToLineDimension2D"/>
</owl:Class>
<owl:Class rdf:about="#PointToPointDimension2D">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#PointEntity2D"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchDimension"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#LineToLineDimension2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#AngleBetweenLinesDimension2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#RadiusDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineLengthDimension2D"/>
```

```
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ArcAngleDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineToPointDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#DiameterDimension2D"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#ArcEntity2D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchEntity"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#CircleEntity2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#ConicEntity2D"/>
  <owl:disjointWith rdf:resource="#EllipseEntity2D"/>
  <owl:disjointWith rdf:resource="#LineEntity2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#PointEntity2D"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="SameYCoordConstraint2D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchConstraint"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#PointEntity2D"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#RadiusDimension2D">
  <owl:disjointWith>
    <owl:Class rdf:about="#LineToPointDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#LineToLineDimension2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#DiameterDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#AngleBetweenLinesDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#PointToPointDimension2D"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchDimension"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineLengthDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ArcAngleDimension2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#ArcEntity2D"/>
            <owl:Class rdf:about="#CircleEntity2D"/>
          </owl:unionOf>
        </owl:Class>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="EllipticalCurve">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Curve"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Curve3D">
  <rdfs:subClassOf rdf:resource="#ReferenceAttribute"/>
  <owl:disjointWith rdf:resource="#ReferenceDatumPlane"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#ReferenceCoordSys"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Sketch"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ExisitngPartReference"/>
  </owl:disjointWith>
```

```
</owl:Class>
<owl:Class rdf:about="#PerpendicularEntitiesConstraint2D">
 <rdfs:subClassOf>
  <owl:Class rdf:about="#SketchConstraint"/>
 </rdfs:subClassOf>
 <owl:disjointWith>
  <owl:Class rdf:about="#VerticalEntityConstraint2D"/>
 </owl:disjointWith>
 <owl:disjointWith>
  <owl:Class rdf:about="#HorizontalEntityConstraint2D"/>
 </owl:disjointWith>
 <owl:disjointWith>
  <owl:Class rdf:about="#SamePointConstraint2D"/>
 </owl:disjointWith>
 <owl:disjointWith rdf:resource="#ColinearConstraint2D"/>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
   >2</owl:cardinality>
   <owl:onProperty>
    <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
   </owl:onProperty>
  </owl:Restriction>
 </rdfs:subClassOf>
 <owl:disjointWith rdf:resource="#ParallelEntitiesConstraint2D"/>
 <owl:disjointWith>
  <owl:Class rdf:about="#EqualRadiiConstraint2D"/>
 </owl:disjointWith>
 <owl:disjointWith>
  <owl:Class rdf:about="#TangentEntitiesConstraint2D"/>
 </owl:disjointWith>
 <owl:disjointWith rdf:resource="#PointOnEntityConstraint2D"/>
 <owl:disjointWith>
  <owl:Class rdf:about="#EqualSegmentsConstraint2D"/>
 </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#EqualRadiiConstraint2D">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
   >2</owl:cardinality>
   <owl:onProperty>
    <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
   </owl:onProperty>
  </owl:Restriction>
 </rdfs:subClassOf>
 <owl:disjointWith rdf:resource="#PerpendicularEntitiesConstraint2D"/>
 <owl:disjointWith rdf:resource="#ParallelEntitiesConstraint2D"/>
 <owl:disjointWith>
  <owl:Class rdf:about="#SamePointConstraint2D"/>
 </owl:disjointWith>
 <rdfs:subClassOf>
  <owl:Class rdf:about="#SketchConstraint"/>
 </rdfs:subClassOf>
 <owl:disjointWith>
  <owl:Class rdf:about="#TangentEntitiesConstraint2D"/>
```

```
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#PointOnEntityConstraint2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#EqualSegmentsConstraint2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#VerticalEntityConstraint2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#ArcEntity2D"/>
            <owl:Class rdf:about="#CircleEntity2D"/>
          </owl:unionOf>
        </owl:Class>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#HorizontalEntityConstraint2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#ColinearConstraint2D"/>
</owl:Class>
<owl:Class rdf:about="#DiameterDimension2D">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#ArcEntity2D"/>
            <owl:Class rdf:about="#CircleEntity2D"/>
          </owl:unionOf>
        </owl:Class>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineToPointDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#PointToPointDimension2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineLengthDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ArcAngleDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#RadiusDimension2D"/>
  <owl:disjointWith rdf:resource="#LineToLineDimension2D"/>
```

```
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchDimension"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#AngleBetweenLinesDimension2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#ArcAngleDimension2D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchDimension"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#ArcEntity2D"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#PointToPointDimension2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineToPointDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#RadiusDimension2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#AngleBetweenLinesDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#DiameterDimension2D"/>
  <owl:disjointWith rdf:resource="#LineToLineDimension2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineLengthDimension2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Loop">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
```

```
      <owl:onProperty>
        <owl:InverseFunctionalProperty rdf:ID="containsCoedge"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:InverseFunctionalProperty rdf:ID="isLoopOf"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#TopologyEntity"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#Coedge"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Vertex"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Edge"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Shell"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Face"/>
</owl:Class>
<owl:Class rdf:about="#EqualSegmentsConstraint2D">
  <owl:disjointWith rdf:resource="#PointOnEntityConstraint2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#SamePointConstraint2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#PerpendicularEntitiesConstraint2D"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchConstraint"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#ColinearConstraint2D"/>
  <owl:disjointWith rdf:resource="#EqualRadiiConstraint2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#LineEntity2D"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#ParallelEntitiesConstraint2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#TangentEntitiesConstraint2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#HorizontalEntityConstraint2D"/>
```

```
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#VerticalEntityConstraint2D"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#VerticalEntityConstraint2D">
  <owl:disjointWith rdf:resource="#PointOnEntityConstraint2D"/>
  <owl:disjointWith rdf:resource="#ParallelEntitiesConstraint2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#SamePointConstraint2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#EqualSegmentsConstraint2D"/>
  <owl:disjointWith rdf:resource="#PerpendicularEntitiesConstraint2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#TangentEntitiesConstraint2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchConstraint"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#LineEntity2D"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#ColinearConstraint2D"/>
  <owl:disjointWith rdf:resource="#EqualRadiiConstraint2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#HorizontalEntityConstraint2D"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="HelicalCurve">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Curve"/>
  </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:about="#Partfile">
 <owl:disjointWith rdf:resource="#Feature"/>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty>
    <owl:ObjectProperty rdf:ID="hasFeature"/>
   </owl:onProperty>
   <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
   >1</owl:minCardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
 <owl:disjointWith rdf:resource="#ReferenceAttribute"/>
 <owl:disjointWith rdf:resource="#BRepEntity"/>
 <owl:disjointWith rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#Entity"/>
 <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty>
    <owl:ObjectProperty rdf:ID="hasReferenceCoordSys"/>
   </owl:onProperty>
   <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
   >1</owl:minCardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty>
    <owl:ObjectProperty rdf:ID="hasReferenceDatumPlane"/>
   </owl:onProperty>
   <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
   >3</owl:minCardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
 <owl:disjointWith rdf:resource="#SketchComponent"/>
</owl:Class>
<owl:Class rdf:about="#Sketch">
 <owl:disjointWith>
  <owl:Class rdf:about="#SketchEntity"/>
 </owl:disjointWith>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
   >1</owl:minCardinality>
   <owl:onProperty>
    <owl:ObjectProperty rdf:ID="hasSketchEntity"/>
   </owl:onProperty>
  </owl:Restriction>
 </rdfs:subClassOf>
 <owl:disjointWith>
  <owl:Class rdf:about="#SketchConstraint"/>
 </owl:disjointWith>
 <owl:disjointWith rdf:resource="#ReferenceDatumPlane"/>
 <owl:disjointWith>
  <owl:Class rdf:about="#SketchDimension"/>
 </owl:disjointWith>
 <owl:disjointWith>
```

```
    <owl:Class rdf:about="#ExisitngPartReference"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Curve3D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#ReferenceCoordSys"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#SketchComponent"/>
  <rdfs:subClassOf rdf:resource="#ReferenceAttribute"/>
 </owl:Class>
 <owl:Class rdf:about="#Coedge">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:InverseFunctionalProperty rdf:ID="hasOppositeCoedge"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#Vertex"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="isCoedgeInLoop"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="isCoedgeOf"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Loop"/>
  <owl:disjointWith rdf:resource="#Face"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:InverseFunctionalProperty rdf:ID="hasNextCoedge"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#TopologyEntity"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:InverseFunctionalProperty rdf:ID="hasPreviousCoedge"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#Edge"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Shell"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="CoordSys2D">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasPoint"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchEntity"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="RevolveFeature">
  <rdfs:subClassOf rdf:resource="#SketchBasedFeature"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasReferenceSketch"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Surface">
  <owl:disjointWith>
    <owl:Class rdf:about="#Point"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Curve"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Vertex"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Edge"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Face"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Shell"/>
```

```
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="GeometryEntity"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="isGeometryOfFace"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="LineToCircularTangentDimension2D">
  <owl:disjointWith rdf:resource="#RadiusDimension2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#ArcEntity2D"/>
            <owl:Class rdf:about="#CircleEntity2D"/>
          </owl:unionOf>
        </owl:Class>
      </owl:someValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#PointToPointDimension2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:resource="#LineEntity2D"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineLengthDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#ArcAngleDimension2D"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchDimension"/>
  </rdfs:subClassOf>
```

```
  <owl:disjointWith rdf:resource="#DiameterDimension2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#AngleBetweenLinesDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineToPointDimension2D"/>
  </owl:disjointWith>
 </owl:Class>
 <owl:Class rdf:about="#LineToPointDimension2D">
  <owl:disjointWith rdf:resource="#RadiusDimension2D"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchDimension"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#PointToPointDimension2D"/>
  <owl:disjointWith rdf:resource="#LineToLineDimension2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineLengthDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#DiameterDimension2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#AngleBetweenLinesDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#ArcAngleDimension2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue>
        <owl:Class rdf:about="#PointEntity2D"/>
      </owl:hasValue>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:resource="#LineEntity2D"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:about="#PointEntity2D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchEntity"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#LineEntity2D"/>
```

```
  <owl:disjointWith>
    <owl:Class rdf:about="#CircleEntity2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#ArcEntity2D"/>
  <owl:disjointWith rdf:resource="#ConicEntity2D"/>
  <owl:disjointWith rdf:resource="#EllipseEntity2D"/>
</owl:Class>
<owl:Class rdf:ID="SphereCornerRoundFeature">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#RoundFeature"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="AngleBetweenLineAndCurveEndTangentDimension2D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchDimension"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
      <owl:hasValue rdf:resource="#LineEntity2D"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class>
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#SketchConstraint"/>
    <owl:Class rdf:about="#SketchDimension"/>
  </owl:unionOf>
</owl:Class>
<owl:Class rdf:about="#SphericalSurface">
  <rdfs:subClassOf rdf:resource="#Surface"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#SplineSurface"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ConicalSurface"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ToroidalSurface"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#PlanarSurface"/>
</owl:Class>
<owl:Class rdf:about="#Point">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#GeometryEntity"/>
```

```
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="isGeometryOfVertex"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#Curve"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Surface"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Vertex"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Edge"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Face"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Shell"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#ExisitngPartReference">
  <rdfs:subClassOf rdf:resource="#ReferenceAttribute"/>
  <owl:disjointWith rdf:resource="#ReferenceDatumPlane"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#ReferenceCoordSys"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Sketch"/>
  <owl:disjointWith rdf:resource="#Curve3D"/>
</owl:Class>
<owl:Class rdf:about="#GeometryEntity">
  <rdfs:subClassOf rdf:resource="#BRepEntity"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#TopologyEntity"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#CircleEntity2D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchEntity"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#LineEntity2D"/>
  <owl:disjointWith rdf:resource="#EllipseEntity2D"/>
  <owl:disjointWith rdf:resource="#PointEntity2D"/>
  <owl:disjointWith rdf:resource="#ArcEntity2D"/>
  <owl:disjointWith rdf:resource="#ConicEntity2D"/>
</owl:Class>
<owl:Class rdf:ID="SweepFeature">
  <rdfs:subClassOf rdf:resource="#SketchBasedFeature"/>
</owl:Class>
<owl:Class rdf:about="#SplineSurface">
  <owl:disjointWith rdf:resource="#SphericalSurface"/>
  <owl:disjointWith>
```

```
    <owl:Class rdf:about="#ConicalSurface"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ToroidalSurface"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#PlanarSurface"/>
  <rdfs:subClassOf rdf:resource="#Surface"/>
</owl:Class>
<owl:Class rdf:about="#ToroidalSurface">
  <rdfs:subClassOf rdf:resource="#Surface"/>
  <owl:disjointWith rdf:resource="#SphericalSurface"/>
  <owl:disjointWith rdf:resource="#SplineSurface"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#ConicalSurface"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#PlanarSurface"/>
</owl:Class>
<owl:Class rdf:about="#AngleBetweenLinesDimension2D">
  <owl:disjointWith rdf:resource="#DiameterDimension2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchDimension"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#RadiusDimension2D"/>
  <owl:disjointWith rdf:resource="#ArcAngleDimension2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#LineEntity2D"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#LineToPointDimension2D"/>
  <owl:disjointWith rdf:resource="#LineToLineDimension2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#LineLengthDimension2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#PointToPointDimension2D"/>
</owl:Class>
<owl:Class rdf:about="#SketchDimension">
  <owl:disjointWith rdf:resource="#Sketch"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#SketchConstraint"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#SketchEntity"/>
  </owl:disjointWith>
```

```
  <rdfs:subClassOf rdf:resource="#SketchComponent"/>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
    </owl:onProperty>
    <owl:someValuesFrom>
     <owl:Class rdf:about="#SketchEntity"/>
    </owl:someValuesFrom>
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#TopologyEntity">
  <rdfs:subClassOf rdf:resource="#BRepEntity"/>
  <owl:disjointWith rdf:resource="#GeometryEntity"/>
</owl:Class>
<owl:Class rdf:ID="ArcToArcVertTangentDimension2D">
  <rdfs:subClassOf rdf:resource="#SketchDimension"/>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >2</owl:cardinality>
    <owl:onProperty>
     <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:allValuesFrom>
     <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#ArcEntity2D"/>
       <owl:Class rdf:about="#CircleEntity2D"/>
      </owl:unionOf>
     </owl:Class>
    </owl:allValuesFrom>
    <owl:onProperty>
     <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ChamferFeature">
  <rdfs:subClassOf rdf:resource="#EdgeFeature"/>
</owl:Class>
<owl:Class rdf:about="#Curve">
  <rdfs:subClassOf rdf:resource="#GeometryEntity"/>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:ObjectProperty rdf:ID="isGeometryOfEdge"/>
    </owl:onProperty>
    <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:minCardinality>
   </owl:Restriction>
```

```
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Point"/>
  <owl:disjointWith rdf:resource="#Surface"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Vertex"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Edge"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Face"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Shell"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="CircularCurve">
  <rdfs:subClassOf rdf:resource="#EllipticalCurve"/>
</owl:Class>
<owl:Class rdf:about="#RoundFeature">
  <rdfs:subClassOf rdf:resource="#EdgeFeature"/>
</owl:Class>
<owl:Class rdf:ID="ExtrudeFeature">
  <rdfs:subClassOf rdf:resource="#SketchBasedFeature"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasReferenceAttribute"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#ReferenceCoordSys">
  <rdfs:subClassOf rdf:resource="#ReferenceAttribute"/>
  <owl:disjointWith rdf:resource="#ReferenceDatumPlane"/>
  <owl:disjointWith rdf:resource="#Sketch"/>
  <owl:disjointWith rdf:resource="#Curve3D"/>
  <owl:disjointWith rdf:resource="#ExisitngPartReference"/>
</owl:Class>
<owl:Class rdf:about="#ReferenceFace">
  <rdfs:subClassOf rdf:resource="#ExisitngPartReference"/>
  <owl:disjointWith rdf:resource="#ReferenceEdge"/>
  <owl:disjointWith rdf:resource="#ReferenceShell"/>
  <owl:disjointWith rdf:resource="#ReferenceVertex"/>
</owl:Class>
<owl:Class rdf:about="#ConicalSurface">
  <rdfs:subClassOf rdf:resource="#Surface"/>
  <owl:disjointWith rdf:resource="#SphericalSurface"/>
  <owl:disjointWith rdf:resource="#SplineSurface"/>
  <owl:disjointWith rdf:resource="#ToroidalSurface"/>
  <owl:disjointWith rdf:resource="#PlanarSurface"/>
</owl:Class>
<owl:Class rdf:about="#Shell">
  <owl:disjointWith>
    <owl:Class rdf:about="#Edge"/>
  </owl:disjointWith>
```

```
  <owl:disjointWith rdf:resource="#Face"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Vertex"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Coedge"/>
  <owl:disjointWith rdf:resource="#Surface"/>
  <rdfs:subClassOf rdf:resource="#TopologyEntity"/>
  <owl:disjointWith rdf:resource="#Point"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:InverseFunctionalProperty rdf:ID="hasFace"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Curve"/>
  <owl:disjointWith rdf:resource="#Loop"/>
</owl:Class>
<owl:Class rdf:about="#LineLengthDimension2D">
  <owl:disjointWith rdf:resource="#LineToPointDimension2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#LineEntity2D"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#LineToLineDimension2D"/>
  <owl:disjointWith rdf:resource="#DiameterDimension2D"/>
  <rdfs:subClassOf rdf:resource="#SketchDimension"/>
  <owl:disjointWith rdf:resource="#AngleBetweenLinesDimension2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#RadiusDimension2D"/>
  <owl:disjointWith rdf:resource="#ArcAngleDimension2D"/>
  <owl:disjointWith rdf:resource="#PointToPointDimension2D"/>
</owl:Class>
<owl:Class rdf:about="#SketchEntity">
  <owl:disjointWith rdf:resource="#Sketch"/>
  <owl:disjointWith rdf:resource="#SketchDimension"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#SketchConstraint"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#SketchComponent"/>
</owl:Class>
<owl:Class rdf:about="#HorizontalEntityConstraint2D">
```

```
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchConstraint"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#TangentEntitiesConstraint2D"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#PerpendicularEntitiesConstraint2D"/>
  <owl:disjointWith rdf:resource="#VerticalEntityConstraint2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#SamePointConstraint2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#LineEntity2D"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#EqualRadiiConstraint2D"/>
  <owl:disjointWith rdf:resource="#PointOnEntityConstraint2D"/>
  <owl:disjointWith rdf:resource="#ColinearConstraint2D"/>
  <owl:disjointWith rdf:resource="#ParallelEntitiesConstraint2D"/>
  <owl:disjointWith rdf:resource="#EqualSegmentsConstraint2D"/>
</owl:Class>
<owl:Class rdf:ID="CoordAxisEntity2D">
  <owl:disjointWith rdf:resource="#PointEntity2D"/>
  <owl:disjointWith rdf:resource="#EllipseEntity2D"/>
  <owl:disjointWith rdf:resource="#ArcEntity2D"/>
  <owl:disjointWith rdf:resource="#ConicEntity2D"/>
  <owl:disjointWith rdf:resource="#CircleEntity2D"/>
  <rdfs:subClassOf rdf:resource="#SketchEntity"/>
</owl:Class>
<owl:Class rdf:about="#Edge">
  <owl:disjointWith rdf:resource="#Shell"/>
  <owl:disjointWith rdf:resource="#Surface"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="hasStartVertex"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >2</owl:cardinality>
    <owl:onProperty>
      <owl:TransitiveProperty rdf:ID="hasVertex"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<owl:disjointWith rdf:resource="#Face"/>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="hasReverseCoedge"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:InverseFunctionalProperty rdf:ID="hasCoedge"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >2</owl:cardinality>
  </owl:Restriction>
</rdfs:subClassOf>
<owl:disjointWith>
  <owl:Class rdf:about="#Vertex"/>
</owl:disjointWith>
<owl:disjointWith rdf:resource="#Curve"/>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="hasForwardCoedge"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="hasEndVertex"/>
    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
  </owl:Restriction>
</rdfs:subClassOf>
<owl:disjointWith rdf:resource="#Coedge"/>
<owl:disjointWith rdf:resource="#Point"/>
<owl:disjointWith rdf:resource="#Loop"/>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="hasGeometryOfCurve"/>
    </owl:onProperty>
```

```
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#TopologyEntity"/>
</owl:Class>
<owl:Class rdf:ID="EllipseXRadiusDimension2D">
  <rdfs:subClassOf rdf:resource="#SketchDimension"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#EllipseEntity2D"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#TangentEntitiesConstraint2D">
  <owl:disjointWith rdf:resource="#HorizontalEntityConstraint2D"/>
  <owl:disjointWith rdf:resource="#EqualRadiiConstraint2D"/>
  <owl:disjointWith rdf:resource="#PointOnEntityConstraint2D"/>
  <owl:disjointWith rdf:resource="#PerpendicularEntitiesConstraint2D"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#SamePointConstraint2D"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchConstraint"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#ColinearConstraint2D"/>
  <owl:disjointWith rdf:resource="#EqualSegmentsConstraint2D"/>
  <owl:disjointWith rdf:resource="#ParallelEntitiesConstraint2D"/>
  <owl:disjointWith rdf:resource="#VerticalEntityConstraint2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="SameXCoordConstraint2D">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#SketchConstraint"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
```

```
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
     </owl:onProperty>
     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >2</owl:cardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
     </owl:onProperty>
     <owl:allValuesFrom rdf:resource="#PointEntity2D"/>
    </owl:Restriction>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#SketchConstraint">
   <owl:disjointWith rdf:resource="#Sketch"/>
   <owl:disjointWith rdf:resource="#SketchDimension"/>
   <owl:disjointWith rdf:resource="#SketchEntity"/>
   <rdfs:subClassOf rdf:resource="#SketchComponent"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:someValuesFrom rdf:resource="#SketchEntity"/>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="PointToPointVertDimension2D">
   <owl:disjointWith rdf:resource="#LineToLineDimension2D"/>
   <owl:disjointWith rdf:resource="#ArcAngleDimension2D"/>
   <owl:disjointWith rdf:resource="#LineLengthDimension2D"/>
   <rdfs:subClassOf rdf:resource="#SketchDimension"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:allValuesFrom rdf:resource="#PointEntity2D"/>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <owl:disjointWith rdf:resource="#LineToPointDimension2D"/>
   <owl:disjointWith rdf:resource="#AngleBetweenLinesDimension2D"/>
   <owl:disjointWith rdf:resource="#DiameterDimension2D"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >2</owl:cardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:about="#isDimensionOfEntity"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
```

```
  <owl:disjointWith rdf:resource="#RadiusDimension2D"/>
</owl:Class>
<owl:Class rdf:about="#Vertex">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="isVertexOf"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Point"/>
  <owl:disjointWith rdf:resource="#Loop"/>
  <owl:disjointWith rdf:resource="#Curve"/>
  <rdfs:subClassOf rdf:resource="#TopologyEntity"/>
  <owl:disjointWith rdf:resource="#Face"/>
  <owl:disjointWith rdf:resource="#Coedge"/>
  <owl:disjointWith rdf:resource="#Shell"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasGeometryOfPoint"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Surface"/>
  <owl:disjointWith rdf:resource="#Edge"/>
</owl:Class>
<owl:Class rdf:ID="SurfaceIntersectionCurve">
  <rdfs:subClassOf rdf:resource="#InterpolatedCurve"/>
</owl:Class>
<owl:Class rdf:ID="PointArray2D">
  <rdfs:subClassOf rdf:resource="#SketchEntity"/>
</owl:Class>
<owl:Class rdf:about="#SamePointConstraint2D">
  <owl:disjointWith rdf:resource="#ParallelEntitiesConstraint2D"/>
  <owl:disjointWith rdf:resource="#PointOnEntityConstraint2D"/>
  <owl:disjointWith rdf:resource="#VerticalEntityConstraint2D"/>
  <owl:disjointWith rdf:resource="#PerpendicularEntitiesConstraint2D"/>
  <rdfs:subClassOf rdf:resource="#SketchConstraint"/>
  <owl:disjointWith rdf:resource="#TangentEntitiesConstraint2D"/>
  <owl:disjointWith rdf:resource="#HorizontalEntityConstraint2D"/>
  <owl:disjointWith rdf:resource="#ColinearConstraint2D"/>
  <owl:disjointWith rdf:resource="#EqualRadiiConstraint2D"/>
  <owl:disjointWith rdf:resource="#EqualSegmentsConstraint2D"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >2</owl:cardinality>
    </owl:Restriction>
```

```
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#PointEntity2D"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#isConstraintOfEntity"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasTopologyOperation">
  <rdfs:range rdf:resource="#TopologyEntity"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#hasBRepOperation"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCenterPoint">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#CircleEntity2D"/>
        <owl:Class rdf:about="#EllipseEntity2D"/>
        <owl:Class rdf:about="#ArcEntity2D"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#hasPoint"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasSketchEntity">
  <rdfs:range rdf:resource="#SketchEntity"/>
  <rdfs:subPropertyOf>
    <owl:InverseFunctionalProperty rdf:ID="hasSketchComponent"/>
  </rdfs:subPropertyOf>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="isSketchEntityOf"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#Sketch"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesHelicalCurve">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="deletesCurve"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="#HelicalCurve"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsCircularCurve">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="#CircularCurve"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="createsEllipticalCurve"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="hasCoordSys2D">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdfs:subPropertyOf rdf:resource="#hasSketchEntity"/>
  <rdfs:range rdf:resource="#CoordSys2D"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesGeometry">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="hasGeometryOperation"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="#GeometryEntity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsToroidalSurface">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="createsSurface"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#ToroidalSurface"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesB-SplineCurve">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="deletesInterpolatedCurve"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#B-SplineCurve"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsHelicalCurve">
  <rdfs:range rdf:resource="#HelicalCurve"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="createsCurve"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isGeometryOfVertex">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
  <rdfs:subPropertyOf>
    <owl:InverseFunctionalProperty rdf:ID="isGeometryOf"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#Vertex"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#hasGeometryOfPoint"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#Point"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesPoint">
  <rdfs:subPropertyOf rdf:resource="#deletesGeometry"/>
  <rdfs:range rdf:resource="#Point"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isSketchEntityOf">
  <rdfs:domain rdf:resource="#SketchEntity"/>
  <owl:inverseOf rdf:resource="#hasSketchEntity"/>
  <rdfs:subPropertyOf>
    <owl:FunctionalProperty rdf:ID="isSketchComponentOf"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="#isDimensionOfEntity">
  <rdfs:domain rdf:resource="#SketchDimension"/>
  <rdfs:range rdf:resource="#SketchEntity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesSphericalSurface">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="deletesSurface"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#SphericalSurface"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasEllipseEntity2D">
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdfs:subPropertyOf rdf:resource="#hasSketchEntity"/>
  <rdfs:range rdf:resource="#EllipseEntity2D"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isGeometryOfEdge">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
  <rdfs:range rdf:resource="#Edge"/>
  <rdfs:domain rdf:resource="#Curve"/>
  <rdfs:subPropertyOf>
    <owl:InverseFunctionalProperty rdf:about="#isGeometryOf"/>
  </rdfs:subPropertyOf>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#hasGeometryOfCurve"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPolylineEntity2D">
  <rdfs:subPropertyOf rdf:resource="#hasSketchEntity"/>
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdfs:range rdf:resource="#PolylineEntity2D"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsPlanarSurface">
  <rdfs:range rdf:resource="#PlanarSurface"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#createsSurface"/>
  </rdfs:subPropertyOf>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#ChamferFeature"/>
        <owl:Class rdf:about="#Feature"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesSurfaceIntersectionCurve">
  <rdfs:range rdf:resource="#SurfaceIntersectionCurve"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#deletesInterpolatedCurve"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesEdge">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="deletesTopology"/>
```

```
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#Edge"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsParameterSpaceCurve">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="#ParameterSpaceCurve"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="createsInterpolatedCurve"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesLinearCurve">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="#LinearCurve"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#deletesCurve"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasExistingPartReference">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#hasReferenceAttribute"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="#ExisitngPartReference"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasBRepOperation">
  <rdfs:range rdf:resource="#BRepEntity"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#createsSurface">
  <rdfs:range rdf:resource="#Surface"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="createsGeometry"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSecondaryReferenceDatumPlane">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#hasReferenceDatumPlane"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#ReferenceDatumPlane"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasReferenceFace">
  <rdfs:range rdf:resource="#ReferenceFace"/>
  <rdfs:subPropertyOf rdf:resource="#hasExistingPartReference"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsSurfaceIntersectionCurve">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#createsInterpolatedCurve"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#SurfaceIntersectionCurve"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasGeometryOfSurface">
  <rdfs:range rdf:resource="#Surface"/>
  <owl:inverseOf>
```

```
    <owl:ObjectProperty rdf:about="#isGeometryOfFace"/>
  </owl:inverseOf>
  <rdfs:subPropertyOf>
    <owl:FunctionalProperty rdf:ID="hasGeometryOf"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Face"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasConstructionLineEntity2D">
  <rdfs:subPropertyOf rdf:resource="#hasSketchEntity"/>
  <rdfs:range rdf:resource="#ConstructionLineEntity2D"/>
  <rdfs:domain rdf:resource="#Sketch"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsEdge">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="createsTopology"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#Edge"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesEllipticalCurve">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#deletesCurve"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#EllipticalCurve"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsShericalSurface">
  <rdfs:range rdf:resource="#SphericalSurface"/>
  <rdfs:subPropertyOf rdf:resource="#createsSurface"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#createsGeometry">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#hasGeometryOperation"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#GeometryEntity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesSplineSurface">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#deletesSurface"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#SplineSurface"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasPoint">
  <rdfs:range rdf:resource="#PointEntity2D"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#LineEntity2D"/>
        <owl:Class rdf:about="#CircleEntity2D"/>
        <owl:Class rdf:about="#ConicEntity2D"/>
        <owl:Class rdf:about="#EllipseEntity2D"/>
        <owl:Class rdf:about="#ArcEntity2D"/>
        <owl:Class rdf:about="#PointArray2D"/>
        <owl:Class rdf:about="#CoordSys2D"/>
```

```
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasReferenceEdge">
  <rdfs:subPropertyOf rdf:resource="#hasExistingPartReference"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="#ReferenceEdge"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasConicEntity2D">
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdfs:range rdf:resource="#ConicEntity2D"/>
  <rdfs:subPropertyOf rdf:resource="#hasSketchEntity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsB-SplineCurve">
  <rdfs:range rdf:resource="#B-SplineCurve"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#createsInterpolatedCurve"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsLinearCurve">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#createsCurve"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="#LinearCurve"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasShoulderPoint">
  <rdfs:subPropertyOf rdf:resource="#hasPoint"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#ConicEntity2D"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesConicalSurface">
  <rdfs:range rdf:resource="#ConicalSurface"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#deletesSurface"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasReferenceSketch">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#SketchBasedFeature"/>
        <owl:Class rdf:about="#Feature"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#hasReferenceAttribute"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#Sketch"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#deletesInterpolatedCurve">
  <rdfs:subPropertyOf>
```

```
    <owl:ObjectProperty rdf:about="#deletesCurve"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#InterpolatedCurve"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isSketchConstraintOf">
  <rdfs:subPropertyOf>
    <owl:FunctionalProperty rdf:about="#isSketchComponentOf"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#SketchConstraint"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="hasSketchConstraint"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasEndPoint">
  <rdfs:subPropertyOf rdf:resource="#hasPoint"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#LineEntity2D"/>
        <owl:Class rdf:about="#ConicEntity2D"/>
        <owl:Class rdf:about="#ConstructionLineEntity2D"/>
        <owl:Class rdf:about="#CoordAxisEntity2D"/>
        <owl:Class rdf:about="#ArcEntity2D"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsFace">
  <rdfs:range rdf:resource="#Face"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#createsTopology"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasReferenceAttribute">
  <rdfs:range rdf:resource="#ReferenceAttribute"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Feature"/>
        <owl:Class rdf:about="#Partfile"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesParameterSpaceCurve">
  <rdfs:subPropertyOf rdf:resource="#deletesInterpolatedCurve"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="#ParameterSpaceCurve"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesLoop">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#deletesTopology"/>
```

```
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#Loop"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesCoedge">
  <rdfs:range rdf:resource="#Coedge"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#deletesTopology"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isConstraintOfEntity">
  <rdfs:range rdf:resource="#SketchEntity"/>
  <rdfs:domain rdf:resource="#SketchConstraint"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#deletesCurve">
  <rdfs:range rdf:resource="#Curve"/>
  <rdfs:subPropertyOf rdf:resource="#deletesGeometry"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasSketchConstraint">
  <rdfs:subPropertyOf>
    <owl:InverseFunctionalProperty rdf:about="#hasSketchComponent"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#SketchConstraint"/>
  <owl:inverseOf rdf:resource="#isSketchConstraintOf"/>
  <rdfs:domain rdf:resource="#Sketch"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsShell">
  <rdfs:range rdf:resource="#Shell"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#createsTopology"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isGeometryOfFace">
  <rdfs:domain rdf:resource="#Surface"/>
  <owl:inverseOf rdf:resource="#hasGeometryOfSurface"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
  <rdfs:range rdf:resource="#Face"/>
  <rdfs:subPropertyOf>
    <owl:InverseFunctionalProperty rdf:about="#isGeometryOf"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesPlanarSurface">
  <rdfs:range rdf:resource="#PlanarSurface"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#deletesSurface"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#deletesSurface">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#deletesGeometry"/>
  <rdfs:range rdf:resource="#Surface"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#createsInterpolatedCurve">
  <rdfs:domain rdf:resource="#Feature"/>
```

```
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#createsCurve"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#InterpolatedCurve"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasReferenceDatumPlane">
  <rdfs:subPropertyOf rdf:resource="#hasReferenceAttribute"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Feature"/>
        <owl:Class rdf:about="#Partfile"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#ReferenceDatumPlane"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsPoint">
  <rdfs:range rdf:resource="#Point"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#createsGeometry"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSplineEntity2D">
  <rdfs:subPropertyOf rdf:resource="#hasSketchEntity"/>
  <rdfs:range rdf:resource="#SplineEntity2D"/>
  <rdfs:domain rdf:resource="#Sketch"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasArcEntity2D">
  <rdfs:range rdf:resource="#ArcEntity2D"/>
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdfs:subPropertyOf rdf:resource="#hasSketchEntity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsCylindricalSurface">
  <rdfs:range rdf:resource="#CylindricalSurface"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="createsConicalSurface"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsLoop">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#createsTopology"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#Loop"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasFeature">
  <rdfs:range rdf:resource="#Feature"/>
  <rdfs:domain rdf:resource="#Partfile"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesFace">
  <rdfs:range rdf:resource="#Face"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#deletesTopology"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="#hasGeometryOfCurve">
  <rdfs:subPropertyOf>
    <owl:FunctionalProperty rdf:about="#hasGeometryOf"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Edge"/>
  <rdfs:range rdf:resource="#Curve"/>
  <owl:inverseOf rdf:resource="#isGeometryOfEdge"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSketchDimension">
  <rdfs:range rdf:resource="#SketchDimension"/>
  <rdfs:domain rdf:resource="#Sketch"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="isSketchDimensionOf"/>
  </owl:inverseOf>
  <rdfs:subPropertyOf>
    <owl:InverseFunctionalProperty rdf:about="#hasSketchComponent"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCircleEntity2D">
  <rdfs:range rdf:resource="#CircleEntity2D"/>
  <rdfs:subPropertyOf rdf:resource="#hasSketchEntity"/>
  <rdfs:domain rdf:resource="#Sketch"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasLineEntity2D">
  <rdfs:subPropertyOf rdf:resource="#hasSketchEntity"/>
  <rdfs:range rdf:resource="#LineEntity2D"/>
  <rdfs:domain rdf:resource="#Sketch"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#deletesTopology">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasTopologyOperation"/>
  <rdfs:range rdf:resource="#TopologyEntity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCurve3D">
  <rdfs:range rdf:resource="#Curve3D"/>
  <rdfs:subPropertyOf rdf:resource="#hasReferenceAttribute"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsCoedge">
  <rdfs:range rdf:resource="#Coedge"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#createsTopology"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPrimaryReferenceDatumPlane">
  <rdfs:range rdf:resource="#ReferenceDatumPlane"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasReferenceDatumPlane"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#createsCurve">
  <rdfs:range rdf:resource="#Curve"/>
  <rdfs:subPropertyOf rdf:resource="#createsGeometry"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCoordAxisEntity2D">
```

```
  <rdfs:range rdf:resource="#CoordAxisEntity2D"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdfs:subPropertyOf rdf:resource="#hasSketchEntity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPointEntity2D">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasSketchEntity"/>
  <rdfs:range rdf:resource="#PointEntity2D"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#createsConicalSurface">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="#ConicalSurface"/>
  <rdfs:subPropertyOf rdf:resource="#createsSurface"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasReferenceCoordSys">
  <rdfs:range rdf:resource="#ReferenceCoordSys"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Feature"/>
        <owl:Class rdf:about="#Partfile"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:subPropertyOf rdf:resource="#hasReferenceAttribute"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsSplineSurface">
  <rdfs:range rdf:resource="#SplineSurface"/>
  <rdfs:subPropertyOf rdf:resource="#createsSurface"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesCylindricalSurface">
  <rdfs:range rdf:resource="#CylindricalSurface"/>
  <rdfs:subPropertyOf rdf:resource="#deletesConicalSurface"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasReferenceShell">
  <rdfs:range rdf:resource="#ReferenceShell"/>
  <rdfs:subPropertyOf rdf:resource="#hasExistingPartReference"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isVertexOf">
  <owl:inverseOf>
    <owl:TransitiveProperty rdf:about="#hasVertex"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#Vertex"/>
  <rdfs:range rdf:resource="#Edge"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isSketchDimensionOf">
  <rdfs:domain rdf:resource="#SketchDimension"/>
  <owl:inverseOf rdf:resource="#hasSketchDimension"/>
  <rdfs:subPropertyOf>
    <owl:FunctionalProperty rdf:about="#isSketchComponentOf"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesCircularCurve">
  <rdfs:range rdf:resource="#CircularCurve"/>
  <rdfs:subPropertyOf rdf:resource="#deletesEllipticalCurve"/>
```

```
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasReferenceVertex">
  <rdfs:range rdf:resource="#ReferenceVertex"/>
  <rdfs:subPropertyOf rdf:resource="#hasExistingPartReference"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPointArray">
  <rdfs:range rdf:resource="#PointArray2D"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#PolylineEntity2D"/>
        <owl:Class rdf:about="#SplineEntity2D"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasStartPoint">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#LineEntity2D"/>
        <owl:Class rdf:about="#ConicEntity2D"/>
        <owl:Class rdf:about="#ConstructionLineEntity2D"/>
        <owl:Class rdf:about="#CoordAxisEntity2D"/>
        <owl:Class rdf:about="#ArcEntity2D"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:subPropertyOf rdf:resource="#hasPoint"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesToroidalSurface">
  <rdfs:range rdf:resource="#ToroidalSurface"/>
  <rdfs:subPropertyOf rdf:resource="#deletesSurface"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#createsTopology">
  <rdfs:subPropertyOf rdf:resource="#hasTopologyOperation"/>
  <rdfs:range rdf:resource="#TopologyEntity"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasGeometryOperation">
  <rdfs:range rdf:resource="#GeometryEntity"/>
  <rdfs:subPropertyOf rdf:resource="#hasBRepOperation"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="createsVertex">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="#Vertex"/>
  <rdfs:subPropertyOf rdf:resource="#createsTopology"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#createsEllipticalCurve">
  <rdfs:subPropertyOf rdf:resource="#createsCurve"/>
  <rdfs:range rdf:resource="#EllipticalCurve"/>
  <rdfs:domain rdf:resource="#Feature"/>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasGeometryOfPoint">
  <rdfs:subPropertyOf>
    <owl:FunctionalProperty rdf:about="#hasGeometryOf"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <owl:inverseOf rdf:resource="#isGeometryOfVertex"/>
  <rdfs:domain rdf:resource="#Vertex"/>
  <rdfs:range rdf:resource="#Point"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesVertex">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="#Vertex"/>
  <rdfs:subPropertyOf rdf:resource="#deletesTopology"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deletesShell">
  <rdfs:range rdf:resource="#Shell"/>
  <rdfs:subPropertyOf rdf:resource="#deletesTopology"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="hasNumberContructionLineEntity2D">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:ID="hasNumberSketchEntity"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Sketch"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasLengthDimensionAttribute">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:ID="hasDimensionAttribute"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberDeleteSplineSurfaceInstances">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:ID="hasNumberGeometryOperationInstances"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberReferenceVertexInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:ID="hasNumberReferenceAttributeInstances"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberCreateFaceInstances">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:ID="hasNumberTopologyOperationInstances"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasDescriptiveAttribute">
  <rdfs:subPropertyOf>
```

```
    <owl:DatatypeProperty rdf:ID="hasParameterAttribute"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberDescriptiveAttributeInstances">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:ID="hasNumberParameterAttributeInstances"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberEllipseEntity2D">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberSketchEntity"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Sketch"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberDeleteVertexInstances">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberTopologyOperationInstances"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberCreateSphericalSurfaceInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberGeometryOperationInstances"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberBooleanAttributeInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberParameterAttributeInstances"/>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#hasNumberParameterAttributeInstances">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:ID="hasNumberInstances"/>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberIntegerAttributeInstances">
  <rdfs:subPropertyOf rdf:resource="#hasNumberParameterAttributeInstances"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberLineEntity2D">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberSketchEntity"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberCoordSys2D">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
```

```
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberSketchEntity"/>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberArcEntity2D">
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberSketchEntity"/>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberCreateVertexInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberTopologyOperationInstances"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberCreateEdgeInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberTopologyOperationInstances"/>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasThicknessDimension">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasLengthDimensionAttribute"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberBRepOperationInstances">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberInstances"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#hasDimensionAttribute">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasParameterAttribute"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberSplineEntity2D">
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberSketchEntity"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasDiameterDimension">
  <rdfs:subPropertyOf rdf:resource="#hasLengthDimensionAttribute"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasBooleanAttribute">
  <rdfs:subPropertyOf>
```

```
    <owl:DatatypeProperty rdf:about="#hasParameterAttribute"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberCreateShellInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberTopologyOperationInstances"/>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberCreateSplineSurfaceInstances">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberGeometryOperationInstances"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#hasNumberTopologyOperationInstances">
  <rdfs:subPropertyOf rdf:resource="#hasNumberBRepOperationInstances"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberCreatePlanarSurfaceInstances">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberGeometryOperationInstances"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberPolylineEntity2D">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberSketchEntity"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Sketch"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberPointEntity2D">
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberSketchEntity"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberReferenceDatumInstances">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberReferenceAttributeInstances"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberReferenceShellInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberReferenceAttributeInstances"/>
  </rdfs:subPropertyOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
```

```
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberDeleteFaceInstances">
  <rdfs:subPropertyOf rdf:resource="#hasNumberTopologyOperationInstances"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberDeleteEdgeInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberTopologyOperationInstances"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#hasParameterAttribute">
  <rdfs:domain rdf:resource="#Feature"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#hasNumberGeometryOperationInstances">
  <rdfs:subPropertyOf rdf:resource="#hasNumberBRepOperationInstances"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberDeletePlanarSurfaceInstances">
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberReferenceFaceInstances">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberReferenceAttributeInstances"/>
  </rdfs:subPropertyOf>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberReferenceCoordSysInstances">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberReferenceAttributeInstances"/>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#hasNumberReferenceAttributeInstances">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberInstances"/>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberReferenceEdgeInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberReferenceAttributeInstances"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberReferenceSketchInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberReferenceAttributeInstances"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasIntergerAttribute">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasParameterAttribute"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasDepthDimension">
```

```
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasLengthDimensionAttribute"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#hasNumberSketchEntity">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:about="#hasNumberInstances"/>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberDimensionAttributeInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberParameterAttributeInstances"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#hasNumberInstances">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberCircleEntity2D">
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberSketchEntity"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberConicEntity2D">
  <rdfs:subPropertyOf rdf:resource="#hasNumberSketchEntity"/>
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasRadiusDimension">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasLengthDimensionAttribute"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberDeleteSphericalSurfaceInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberCurve3DInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberReferenceAttributeInstances"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasAngularDimensionAttribute">
  <rdfs:subPropertyOf rdf:resource="#hasDimensionAttribute"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberDeleteToroidalSurfaceInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberCreateToroidalSurfaceInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberDeleteShellInstances">
  <rdfs:subPropertyOf rdf:resource="#hasNumberTopologyOperationInstances"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
```

```
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasNumberCoordAxisEntity2D">
  <rdfs:subPropertyOf rdf:resource="#hasNumberSketchEntity"/>
  <rdfs:domain rdf:resource="#Sketch"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:TransitiveProperty rdf:about="#hasVertex">
  <rdfs:range rdf:resource="#Vertex"/>
  <owl:inverseOf rdf:resource="#isVertexOf"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Edge"/>
</owl:TransitiveProperty>
<owl:FunctionalProperty rdf:ID="hasEndAngle">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#ArcEntity2D"/>
        <owl:Class rdf:about="#SplineEntity2D"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberCreateLinearCurveInstances">
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberCreateSurfaceInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberExistingPartReferenceInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberReferenceAttributeInstances"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#hasReverseCoedge">
  <rdfs:domain rdf:resource="#Edge"/>
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:ID="isReverseCoedgeOf"/>
  </owl:inverseOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:subPropertyOf>
    <owl:InverseFunctionalProperty rdf:about="#hasCoedge"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#Coedge"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasCoordSysType">
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:rest rdf:parseType="Resource">
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
        >Cylindrical</rdf:first>
        <rdf:rest rdf:parseType="Resource">
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Spherical</rdf:first>
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
        </rdf:rest>
      </rdf:rest>
      <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Cartesian</rdf:first>
    </owl:oneOf>
  </owl:DataRange>
 </rdfs:range>
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
 <rdfs:domain rdf:resource="#ReferenceCoordSys"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasCoordSysLocationType">
 <rdfs:domain rdf:resource="#ReferenceCoordSys"/>
 <rdfs:range>
  <owl:DataRange>
   <owl:oneOf rdf:parseType="Resource">
    <rdf:rest rdf:parseType="Resource">
     <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
     <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >Intersection</rdf:first>
    </rdf:rest>
    <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Origin</rdf:first>
   </owl:oneOf>
  </owl:DataRange>
 </rdfs:range>
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasConicParameter">
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
 <rdfs:domain rdf:resource="#ConicEntity2D"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberCreateConicalSurfaceInstances">
 <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
 <rdfs:domain rdf:resource="#Feature"/>
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasStartAngle">
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#ArcEntity2D"/>
    <owl:Class rdf:about="#SplineEntity2D"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="isAtYCoordinate">
 <rdfs:domain rdf:resource="#Point"/>
```

```
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasXCoord">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#PointEntity2D"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasDatumPlaneType">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#ReferenceDatumPlane"/>
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >X_Plane</rdf:first>
        <rdf:rest rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Z_Plane</rdf:first>
          </rdf:rest>
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Y_Plane</rdf:first>
        </rdf:rest>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberCreateInterpolatedCurveInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberDeleteLoopInstances">
  <rdfs:subPropertyOf rdf:resource="#hasNumberTopologyOperationInstances"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberCreateLoopInstances">
  <rdfs:subPropertyOf rdf:resource="#hasNumberTopologyOperationInstances"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberDeleteConicalSurfaceInstances">
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#isCoedgeInLoop">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Coedge"/>
  <rdfs:range rdf:resource="#Loop"/>
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:about="#containsCoedge"/>
  </owl:inverseOf>
```

```
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberCreateHelicalCurveInstances">
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#hasGeometryOf">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Face"/>
        <owl:Class rdf:about="#Edge"/>
        <owl:Class rdf:about="#Vertex"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:about="#isGeometryOf"/>
  </owl:inverseOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#GeometryEntity"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberDeleteInterpolatedCurveInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#hasEndVertex">
  <rdfs:range rdf:resource="#Vertex"/>
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:ID="isEndVertexOf"/>
  </owl:inverseOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasVertex"/>
  <rdfs:domain rdf:resource="#Edge"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberCreateCurveInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#isSketchComponentOf">
  <rdfs:domain rdf:resource="#SketchComponent"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Sketch"/>
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:about="#hasSketchComponent"/>
  </owl:inverseOf>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberDeleteCurveInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#isFaceOf">
  <rdfs:domain rdf:resource="#Face"/>
```

```
  <rdfs:range rdf:resource="#Shell"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:about="#hasFace"/>
  </owl:inverseOf>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberDeletePointInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberDeleteEllipticalCurveInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasDimensionValue">
  <rdfs:domain rdf:resource="#SketchDimension"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#isCoedgeOf">
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:about="#hasCoedge"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#Coedge"/>
  <rdfs:range rdf:resource="#Edge"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="isThinShellPart">
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</rdf:first>
        <rdf:rest rdf:parseType="Resource">
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >true</rdf:first>
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
        </rdf:rest>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasBooleanAttribute"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Feature"/>
        <owl:Class rdf:about="#ExtrudeFeature"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="isAtZCoordinate">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
```

```
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain rdf:resource="#Point"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasN_Points">
  <rdfs:domain rdf:resource="#PointArray2D"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#hasForwardCoedge">
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:ID="isForwardCoedgeOf"/>
  </owl:inverseOf>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Coedge"/>
  <rdfs:domain rdf:resource="#Edge"/>
  <rdfs:subPropertyOf>
    <owl:InverseFunctionalProperty rdf:about="#hasCoedge"/>
  </rdfs:subPropertyOf>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberCreateEllipticalCurveInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberDeleteHelicalCurveInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="isAtXCoordinate">
  <rdfs:domain rdf:resource="#Point"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberDeleteSurfaceInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberDeleteCoedgeInstances">
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberTopologyOperationInstances"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberCreatePointInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="removesMaterial">
  <rdfs:subPropertyOf rdf:resource="#hasBooleanAttribute"/>
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:rest rdf:parseType="Resource">
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
```

```
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >true</rdf:first>
       </rdf:rest>
       <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
       >false</rdf:first>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Feature"/>
        <owl:Class rdf:about="#ExtrudeFeature"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberDeleteLinearCurveInstances">
  <rdfs:subPropertyOf rdf:resource="#hasNumberGeometryOperationInstances"/>
  <rdfs:domain rdf:resource="#Feature"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasXRadius">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain rdf:resource="#EllipseEntity2D"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasRadius">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#CircleEntity2D"/>
        <owl:Class rdf:about="#ArcEntity2D"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasYRadius">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#EllipseEntity2D"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasNumberCreateCoedgeInstances">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasNumberTopologyOperationInstances"/>
  <rdfs:domain rdf:resource="#Feature"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasExtSurfaceType">
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Solid</rdf:first>
```

```
      <rdf:rest rdf:parseType="Resource">
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Surface</rdf:first>
      </rdf:rest>
    </owl:oneOf>
   </owl:DataRange>
  </rdfs:range>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hasDescriptiveAttribute"/>
  <rdfs:domain>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Feature"/>
     <owl:Class rdf:about="#ExtrudeFeature"/>
    </owl:unionOf>
   </owl:Class>
  </rdfs:domain>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasYCoord">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain rdf:resource="#PointEntity2D"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#hasStartVertex">
  <rdfs:range rdf:resource="#Vertex"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <owl:inverseOf>
   <owl:InverseFunctionalProperty rdf:ID="isStartVertexOf"/>
  </owl:inverseOf>
  <rdfs:subPropertyOf rdf:resource="#hasVertex"/>
  <rdfs:domain rdf:resource="#Edge"/>
</owl:FunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#hasOppositeCoedge">
  <rdfs:domain rdf:resource="#Coedge"/>
  <rdfs:range rdf:resource="#Coedge"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <owl:inverseOf rdf:resource="#hasOppositeCoedge"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#isReverseCoedgeOf">
  <rdfs:domain rdf:resource="#Coedge"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Edge"/>
  <rdfs:subPropertyOf rdf:resource="#isCoedgeOf"/>
  <owl:inverseOf rdf:resource="#hasReverseCoedge"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#isGeometryOf">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Vertex"/>
     <owl:Class rdf:about="#Face"/>
     <owl:Class rdf:about="#Edge"/>
```

```
    </owl:unionOf>
   </owl:Class>
  </rdfs:range>
  <rdfs:domain rdf:resource="#GeometryEntity"/>
  <owl:inverseOf rdf:resource="#hasGeometryOf"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#hasLoop">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <owl:inverseOf>
   <owl:InverseFunctionalProperty rdf:about="#isLoopOf"/>
  </owl:inverseOf>
  <rdfs:range rdf:resource="#Loop"/>
  <rdfs:domain rdf:resource="#Face"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#hasFace">
  <rdfs:range rdf:resource="#Face"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Shell"/>
  <owl:inverseOf rdf:resource="#isFaceOf"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#containsCoedge">
  <rdfs:range rdf:resource="#Coedge"/>
  <rdfs:domain rdf:resource="#Loop"/>
  <owl:inverseOf rdf:resource="#isCoedgeInLoop"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#isEndVertexOf">
  <rdfs:subPropertyOf rdf:resource="#isVertexOf"/>
  <rdfs:range rdf:resource="#Edge"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Vertex"/>
  <owl:inverseOf rdf:resource="#hasEndVertex"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#isStartVertexOf">
  <rdfs:subPropertyOf rdf:resource="#isVertexOf"/>
  <owl:inverseOf rdf:resource="#hasStartVertex"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Edge"/>
  <rdfs:domain rdf:resource="#Vertex"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#isForwardCoedgeOf">
  <rdfs:domain rdf:resource="#Coedge"/>
  <rdfs:subPropertyOf rdf:resource="#isCoedgeOf"/>
  <owl:inverseOf rdf:resource="#hasForwardCoedge"/>
  <rdfs:range rdf:resource="#Edge"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#isLoopOf">
  <rdfs:domain rdf:resource="#Loop"/>
  <owl:inverseOf rdf:resource="#hasLoop"/>
  <rdfs:range rdf:resource="#Face"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#hasCoedge">
```

```
  <rdfs:range rdf:resource="#Coedge"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Edge"/>
  <owl:inverseOf rdf:resource="#isCoedgeOf"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#hasSketchComponent">
  <rdfs:domain rdf:resource="#Sketch"/>
  <owl:inverseOf rdf:resource="#isSketchComponentOf"/>
  <rdfs:range rdf:resource="#SketchComponent"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#hasPreviousCoedge">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Coedge"/>
  <owl:inverseOf>
    <owl:InverseFunctionalProperty rdf:about="#hasNextCoedge"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#Coedge"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:about="#hasNextCoedge">
  <rdfs:domain rdf:resource="#Coedge"/>
  <owl:inverseOf rdf:resource="#hasPreviousCoedge"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Coedge"/>
</owl:InverseFunctionalProperty>
<swrl:Imp rdf:about="http://www.CAD-ontology.com/BRepRule-6_ShellHasFaces">
  <swrl:head>
    <swrl:AtomList>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      <rdf:first>
        <swrl:IndividualPropertyAtom>
          <swrl:argument2>
            <swrl:Variable rdf:ID="a"/>
          </swrl:argument2>
          <swrl:argument1>
            <swrl:Variable rdf:ID="b"/>
          </swrl:argument1>
          <swrl:propertyPredicate rdf:resource="#hasFace"/>
        </swrl:IndividualPropertyAtom>
      </rdf:first>
    </swrl:AtomList>
  </swrl:head>
  <swrl:body>
    <swrl:AtomList>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      <rdf:first>
        <swrl:IndividualPropertyAtom>
          <swrl:argument2 rdf:resource="#b"/>
          <swrl:argument1 rdf:resource="#a"/>
          <swrl:propertyPredicate rdf:resource="#isFaceOf"/>
        </swrl:IndividualPropertyAtom>
      </rdf:first>
    </swrl:AtomList>
  </swrl:body>
```

```
</swrl:Imp>
<swrl:Imp rdf:about="http://www.CAD-ontology.com/ExtrudeFeature_Rule">
 <swrl:head>
  <swrl:AtomList>
   <rdf:first>
    <swrl:ClassAtom>
     <swrl:argument1>
      <swrl:Variable rdf:ID="F"/>
     </swrl:argument1>
     <swrl:classPredicate rdf:resource="#ExtrudeFeature"/>
    </swrl:ClassAtom>
   </rdf:first>
   <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
  </swrl:AtomList>
 </swrl:head>
 <swrl:body>
  <swrl:AtomList>
   <rdf:first>
    <swrl:ClassAtom>
     <swrl:argument1 rdf:resource="#F"/>
     <swrl:classPredicate rdf:resource="#SketchBasedFeature"/>
    </swrl:ClassAtom>
   </rdf:first>
   <rdf:rest>
    <swrl:AtomList>
     <rdf:rest>
      <swrl:AtomList>
       <rdf:rest>
        <swrl:AtomList>
         <rdf:rest>
          <swrl:AtomList>
           <rdf:rest>
            <swrl:AtomList>
             <rdf:first>
              <swrl:BuiltinAtom>
               <swrl:arguments>
                <rdf:List>
                 <rdf:rest>
                  <rdf:List>
                   <rdf:first>
                    <swrl:Variable rdf:ID="TNE1"/>
                   </rdf:first>
                   <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                  </rdf:List>
                 </rdf:rest>
                 <rdf:first>
                  <swrl:Variable rdf:ID="NS1"/>
                 </rdf:first>
                </rdf:List>
               </swrl:arguments>
               <swrl:builtin rdf:resource="http://www.w3.org/2003/11/swrlb#lessThanOrEqual"/>
              </swrl:BuiltinAtom>
             </rdf:first>
             <rdf:rest>
              <swrl:AtomList>
               <rdf:first>
```

<swrl:DatavaluedPropertyAtom>
<swrl:argument2>
<swrl:Variable rdf:ID="NE2"/>
</swrl:argument2>
<swrl:argument1>
<swrl:Variable rdf:ID="S"/>
</swrl:argument1>
<swrl:propertyPredicate rdf:resource="#hasNumberCircleEntity2D"/>
</swrl:DatavaluedPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:DatavaluedPropertyAtom>
<swrl:argument1 rdf:resource="#S"/>
<swrl:argument2>
<swrl:Variable rdf:ID="NE3"/>
</swrl:argument2>
<swrl:propertyPredicate rdf:resource="#hasNumberEllipseEntity2D"/>
</swrl:DatavaluedPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:DatavaluedPropertyAtom>
<swrl:argument1 rdf:resource="#S"/>
<swrl:argument2>
<swrl:Variable rdf:ID="NE4"/>
</swrl:argument2>
<swrl:propertyPredicate rdf:resource="#hasNumberArcEntity2D"/>
</swrl:DatavaluedPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:DatavaluedPropertyAtom>
<swrl:argument2>
<swrl:Variable rdf:ID="NS2"/>
</swrl:argument2>
<swrl:argument1 rdf:resource="#F"/>
<swrl:propertyPredicate
rdf:resource="#hasNumberCreateConicalSurfaceInstances"/>
</swrl:DatavaluedPropertyAtom>
</rdf:first>
<rdf:rest>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:BuiltinAtom>
<swrl:arguments>
<rdf:List>
<rdf:first rdf:resource="#NS2"/>
<rdf:rest>
<rdf:List>
<rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>

```
    <rdf:first>
      <swrl:Variable rdf:ID="TNE2"/>
    </rdf:first>                                    </rdf:List>
                                                  </rdf:rest>
                                                 </rdf:List>
                                                </swrl:arguments>
                                                <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#lessThanOrEqual"/>
                                                 </swrl:BuiltinAtom>
                                               </rdf:first>
                                               <rdf:rest>
                                                 <swrl:AtomList>
                                                   <rdf:first>
                                                     <swrl:DatavaluedPropertyAtom>
                                                       <swrl:argument2>
                                                         <swrl:Variable rdf:ID="NE5"/>
                                                       </swrl:argument2>
                                                       <swrl:argument1 rdf:resource="#S"/>
                                                       <swrl:propertyPredicate rdf:resource="#hasNumberConicEntity2D"/>
                                                     </swrl:DatavaluedPropertyAtom>
                                                   </rdf:first>
                                                   <rdf:rest>
                                                     <swrl:AtomList>
                                                       <rdf:first>
                                                         <swrl:DatavaluedPropertyAtom>
    <swrl:argument2>
      <swrl:Variable rdf:ID="NE6"/>
    </swrl:argument2>
    <swrl:argument1 rdf:resource="#S"/>
    <swrl:propertyPredicate rdf:resource="#hasNumberSplineEntity2D"/>
                                                           </swrl:DatavaluedPropertyAtom>
                                                       </rdf:first>
                                                       <rdf:rest>
                                                         <swrl:AtomList>
    <rdf:first>
      <swrl:DatavaluedPropertyAtom>
        <swrl:argument1 rdf:resource="#S"/>
        <swrl:argument2>
          <swrl:Variable rdf:ID="NE7"/>
        </swrl:argument2>
        <swrl:propertyPredicate rdf:resource="#hasNumberPolylineEntity2D"/>
      </swrl:DatavaluedPropertyAtom>
    </rdf:first>
    <rdf:rest>
      <swrl:AtomList>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:BuiltinAtom>
                    <swrl:arguments>
                      <rdf:List>
                        <rdf:first>
                          <swrl:Variable rdf:ID="NS3"/>
                        </rdf:first>
```

```
      <rdf:rest>
       <rdf:List>
        <rdf:first>
         <swrl:Variable rdf:ID="TNE3"/>
        </rdf:first>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
       </rdf:List>
      </rdf:rest>
     </rdf:List>
    </swrl:arguments>
    <swrl:builtin rdf:resource="http://www.w3.org/2003/11/swrlb#lessThanOrEqual"/>
   </swrl:BuiltinAtom>
  </rdf:first>
  <rdf:rest>
   <swrl:AtomList>
    <rdf:rest>
     <swrl:AtomList>
      <rdf:first>
       <swrl:BuiltinAtom>
        <swrl:arguments>
         <rdf:List>
          <rdf:first>
           <swrl:Variable rdf:ID="TNS"/>
          </rdf:first>
          <rdf:rest>
           <rdf:List>
            <rdf:rest>
             <rdf:List>
              <rdf:first rdf:resource="#NS2"/>
              <rdf:rest>
               <rdf:List>
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                <rdf:first rdf:resource="#NS3"/>
               </rdf:List>
              </rdf:rest>
             </rdf:List>
            </rdf:rest>
            <rdf:first rdf:resource="#NS1"/>
           </rdf:List>
          </rdf:rest>
         </rdf:List>
        </swrl:arguments>
        <swrl:builtin rdf:resource="http://www.w3.org/2003/11/swrlb#add"/>
       </swrl:BuiltinAtom>
      </rdf:first>
      <rdf:rest>
       <swrl:AtomList>
        <rdf:first>
         <swrl:BuiltinAtom>
          <swrl:arguments>
           <rdf:List>
            <rdf:first rdf:resource="#TNS"/>
            <rdf:rest>
             <rdf:List>
              <rdf:first>
               <swrl:Variable rdf:ID="TNE"/>
```

```
                    </rdf:first>
                    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                   </rdf:List>
                  </rdf:rest>
                 </rdf:List>
                </swrl:arguments>
                <swrl:builtin rdf:resource="http://www.w3.org/2003/11/swrlb#lessThanOrEqual"/>
               </swrl:BuiltinAtom>
              </rdf:first>
              <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
             </swrl:AtomList>
            </rdf:rest>
           </swrl:AtomList>
          </rdf:rest>
          <rdf:first>
           <swrl:BuiltinAtom>
            <swrl:builtin rdf:resource="http://www.w3.org/2003/11/swrlb#add"/>
            <swrl:arguments>
             <rdf:List>
              <rdf:rest>
               <rdf:List>
                <rdf:rest>
                 <rdf:List>
                  <rdf:first rdf:resource="#TNE2"/>
                  <rdf:rest>
                   <rdf:List>
                    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                    <rdf:first rdf:resource="#TNE3"/>
                   </rdf:List>
                  </rdf:rest>
                 </rdf:List>
                </rdf:rest>
                <rdf:first rdf:resource="#TNE1"/>
               </rdf:List>
              </rdf:rest>
              <rdf:first rdf:resource="#TNE"/>
             </rdf:List>
            </swrl:arguments>
           </swrl:BuiltinAtom>
          </rdf:first>
         </swrl:AtomList>
        </rdf:rest>
       </swrl:AtomList>
      </rdf:rest>
      <rdf:first>
       <swrl:BuiltinAtom>
        <swrl:arguments>
         <rdf:List>
          <rdf:first rdf:resource="#TNE3"/>
          <rdf:rest>
           <rdf:List>
            <rdf:rest>
             <rdf:List>
              <rdf:first rdf:resource="#NE6"/>
              <rdf:rest>
               <rdf:List>
```

<rdf:first rdf:resource="#NE7"/>
<rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
</rdf:List>
</rdf:rest>
</rdf:List>
</rdf:rest>
<rdf:first rdf:resource="#NE5"/>
</rdf:List>
</rdf:rest>
</rdf:List>
</swrl:arguments>
<swrl:builtin rdf:resource="http://www.w3.org/2003/11/swrlb#add"/>
</swrl:BuiltinAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:DatavaluedPropertyAtom>
<swrl:propertyPredicate rdf:resource="#hasNumberCreateSplineSurfaceInstances"/>
<swrl:argument1 rdf:resource="#F"/>
<swrl:argument2 rdf:resource="#NS3"/>
</swrl:DatavaluedPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest> </swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:BuiltinAtom>
<swrl:arguments>
<rdf:List>
<rdf:first rdf:resource="#TNE2"/>
<rdf:rest>
<rdf:List>
<rdf:rest>
<rdf:List>
<rdf:first rdf:resource="#NE2"/>
<rdf:rest>
<rdf:List>
<rdf:first rdf:resource="#NE3"/>
<rdf:rest>
<rdf:List>
<rdf:first rdf:resource="#NE3"/>
<rdf:rest>
<rdf:List>
<rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
<rdf:first rdf:resource="#NE4"/>
</rdf:List>
</rdf:rest>
</rdf:List>
</rdf:rest>

```
  </rdf:List>
</rdf:rest>                                    </rdf:List>
                                             </rdf:rest>
                                             <rdf:first rdf:resource="#NE2"/>
                                            </rdf:List>
                                           </rdf:rest>
                                          </rdf:List>
                                         </swrl:arguments>
                                         <swrl:builtin rdf:resource="http://www.w3.org/2003/11/swrlb#add"/>
                                        </swrl:BuiltinAtom>
                                       </rdf:first>
                                      </swrl:AtomList>
                                     </rdf:rest>
                                    </swrl:AtomList>
                                   </rdf:rest>
                                  </swrl:AtomList>
                                 </rdf:rest>
                                </swrl:AtomList>
                               </rdf:rest>
                              </swrl:AtomList>
                             </rdf:rest>
                            </swrl:AtomList>
                           </rdf:rest>
                           <rdf:first>
                            <swrl:BuiltinAtom>
                             <swrl:builtin rdf:resource="http://www.w3.org/2003/11/swrlb#add"/>
                             <swrl:arguments>
                              <rdf:List>
                               <rdf:rest>
                                <rdf:List>
                                 <rdf:first>
                                  <swrl:Variable rdf:ID="NE1"/>
                                 </rdf:first>
                                 <rdf:rest>
                                  <rdf:List>
                                   <rdf:first rdf:datatype=
                                    "http://www.w3.org/2001/XMLSchema#long"
                                    >2</rdf:first>
                                   <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                  </rdf:List>
                                 </rdf:rest>
                                </rdf:List>
                               </rdf:rest>
                               <rdf:first rdf:resource="#TNE1"/>
                              </rdf:List>
                             </swrl:arguments>
                            </swrl:BuiltinAtom>
                           </rdf:first>
                          </swrl:AtomList>
                         </rdf:rest>
                         <rdf:first>
                          <swrl:DatavaluedPropertyAtom>
                           <swrl:propertyPredicate rdf:resource="#hasNumberCreatePlanarSurfaceInstances"/>
                           <swrl:argument1 rdf:resource="#F"/>
                           <swrl:argument2 rdf:resource="#NS1"/>
                          </swrl:DatavaluedPropertyAtom>
```

```
            </rdf:first>
           </swrl:AtomList>
          </rdf:rest>
          <rdf:first>
           <swrl:DatavaluedPropertyAtom>
            <swrl:propertyPredicate rdf:resource="#hasNumberLineEntity2D"/>
            <swrl:argument2 rdf:resource="#NE1"/>
            <swrl:argument1 rdf:resource="#S"/>
           </swrl:DatavaluedPropertyAtom>
          </rdf:first>
         </swrl:AtomList>
        </rdf:rest>
        <rdf:first>
         <swrl:IndividualPropertyAtom>
          <swrl:argument1 rdf:resource="#F"/>
          <swrl:argument2 rdf:resource="#S"/>
          <swrl:propertyPredicate rdf:resource="#hasReferenceSketch"/>
         </swrl:IndividualPropertyAtom>
        </rdf:first>
       </swrl:AtomList>
      </rdf:rest>
     </swrl:AtomList>
    </swrl:body>
</swrl:Imp>
<swrl:Imp rdf:about="http://www.CAD-ontology.com/EdgeFeatureRule">
  <swrl:body>
   <swrl:AtomList>
    <rdf:first>
     <swrl:ClassAtom>
      <swrl:argument1 rdf:resource="#F"/>
      <swrl:classPredicate rdf:resource="#Feature"/>
     </swrl:ClassAtom>
    </rdf:first>
    <rdf:rest>
     <swrl:AtomList>
      <rdf:rest>
       <swrl:AtomList>
        <rdf:first>
         <swrl:DatavaluedPropertyAtom>
          <swrl:propertyPredicate rdf:resource="#hasNumberReferenceAttributeInstances"/>
          <swrl:argument1 rdf:resource="#F"/>
          <swrl:argument2>
           <swrl:Variable rdf:ID="N2"/>
          </swrl:argument2>
         </swrl:DatavaluedPropertyAtom>
        </rdf:first>
        <rdf:rest>
         <swrl:AtomList>
          <rdf:rest>
           <swrl:AtomList>
            <rdf:first>
             <swrl:BuiltinAtom>
              <swrl:arguments>
               <rdf:List>
                <rdf:first>
                 <swrl:Variable rdf:ID="N"/>
```

```
                        </rdf:first>
                        <rdf:rest>
                          <rdf:List>
                            <rdf:first rdf:resource="#N2"/>
                            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                          </rdf:List>
                        </rdf:rest>
                      </rdf:List>
                    </swrl:arguments>
                    <swrl:builtin rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
                  </swrl:BuiltinAtom>
                </rdf:first>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:first>
                      <swrl:DatavaluedPropertyAtom>
                        <swrl:propertyPredicate rdf:resource="#hasNumberCreateSurfaceInstances"/>
                        <swrl:argument1 rdf:resource="#F"/>
                        <swrl:argument2>
                          <swrl:Variable rdf:ID="N3"/>
                        </swrl:argument2>
                      </swrl:DatavaluedPropertyAtom>
                    </rdf:first>
                    <rdf:rest>
                      <swrl:AtomList>
                        <rdf:first>
                          <swrl:BuiltinAtom>
                            <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#greaterThanOrEqual"/>
                            <swrl:arguments>
                              <rdf:List>
                                <rdf:first rdf:resource="#N3"/>
                                <rdf:rest>
                                  <rdf:List>
                                    <rdf:first rdf:resource="#N"/>
                                    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                  </rdf:List>
                                </rdf:rest>
                              </rdf:List>
                            </swrl:arguments>
                          </swrl:BuiltinAtom>
                        </rdf:first>
                        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                      </swrl:AtomList>
                    </rdf:rest>
                  </swrl:AtomList>
                </rdf:rest>
                <rdf:first>
                  <swrl:BuiltinAtom>
                    <swrl:arguments>
                      <rdf:List>
                        <rdf:rest>
                          <rdf:List>
                            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
```

<rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#long"
>1</rdf:first>
</rdf:List>
</rdf:rest>
<rdf:first rdf:resource="#N"/>
</rdf:List>
</swrl:arguments>
<swrl:builtin rdf:resource="http://www.w3.org/2003/11/swrlb#greaterThanOrEqual"/>
</swrl:BuiltinAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</rdf:rest>
<rdf:first>
<swrl:DatavaluedPropertyAtom>
<swrl:argument1 rdf:resource="#F"/>
<swrl:argument2 rdf:resource="#N"/>
<swrl:propertyPredicate rdf:resource="#hasNumberReferenceEdgeInstances"/>
</swrl:DatavaluedPropertyAtom>
</rdf:first>
</swrl:AtomList>
</rdf:rest>
</swrl:AtomList>
</swrl:body>
<swrl:head>
<swrl:AtomList>
<rdf:first>
<swrl:ClassAtom>
<swrl:classPredicate rdf:resource="#EdgeFeature"/>
<swrl:argument1 rdf:resource="#F"/>
</swrl:ClassAtom>
</rdf:first>
<rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
</swrl:AtomList>
</swrl:head>
</swrl:Imp>
<swrl:Imp rdf:about="http://www.CAD-ontology.com/SketchFeature_Rule">
<swrl:head>
<swrl:AtomList>
<rdf:first>
<swrl:ClassAtom>
<swrl:classPredicate rdf:resource="#SketchBasedFeature"/>
<swrl:argument1 rdf:resource="#F"/>
</swrl:ClassAtom>
</rdf:first>
<rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
</swrl:AtomList>
</swrl:head>
<swrl:body>
<swrl:AtomList>
<rdf:rest>
<swrl:AtomList>
<rdf:first>
<swrl:DatavaluedPropertyAtom>
<swrl:argument2 rdf:datatype="http://www.w3.org/2001/XMLSchema#long"

```
            >1</swrl:argument2>
            <swrl:argument1 rdf:resource="#F"/>
            <swrl:propertyPredicate rdf:resource="#hasNumberReferenceSketchInstances"/>
           </swrl:DatavaluedPropertyAtom>
          </rdf:first>
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
         </swrl:AtomList>
        </rdf:rest>
        <rdf:first>
         <swrl:ClassAtom>
          <swrl:classPredicate rdf:resource="#Feature"/>
          <swrl:argument1 rdf:resource="#F"/>
         </swrl:ClassAtom>
        </rdf:first>
       </swrl:AtomList>
      </swrl:body>
     </swrl:Imp>
     <swrl:Imp rdf:about="http://www.CAD-ontology.com/BRepRule-7_PointOfVertex">
      <swrl:body>
       <swrl:AtomList>
        <rdf:first>
         <swrl:IndividualPropertyAtom>
          <swrl:argument1 rdf:resource="#a"/>
          <swrl:argument2 rdf:resource="#b"/>
          <swrl:propertyPredicate rdf:resource="#hasGeometryOfPoint"/>
         </swrl:IndividualPropertyAtom>
        </rdf:first>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
       </swrl:AtomList>
      </swrl:body>
      <swrl:head>
       <swrl:AtomList>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
        <rdf:first>
         <swrl:IndividualPropertyAtom>
          <swrl:argument2 rdf:resource="#a"/>
          <swrl:argument1 rdf:resource="#b"/>
          <swrl:propertyPredicate rdf:resource="#isGeometryOfVertex"/>
         </swrl:IndividualPropertyAtom>
        </rdf:first>
       </swrl:AtomList>
      </swrl:head>
     </swrl:Imp>
     <swrl:Imp rdf:about="http://www.CAD-ontology.com/BRepRule-5_LoopContainsCoedge">
      <swrl:head>
       <swrl:AtomList>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
        <rdf:first>
         <swrl:IndividualPropertyAtom>
          <swrl:propertyPredicate rdf:resource="#containsCoedge"/>
          <swrl:argument2 rdf:resource="#a"/>
          <swrl:argument1 rdf:resource="#b"/>
         </swrl:IndividualPropertyAtom>
        </rdf:first>
       </swrl:AtomList>
      </swrl:head>
```

```
  <swrl:body>
   <swrl:AtomList>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    <rdf:first>
     <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="#isCoedgeInLoop"/>
      <swrl:argument2 rdf:resource="#b"/>
      <swrl:argument1 rdf:resource="#a"/>
     </swrl:IndividualPropertyAtom>
    </rdf:first>
   </swrl:AtomList>
  </swrl:body>
</swrl:Imp>
<swrl:Imp rdf:about="http://www.CAD-ontology.com/BRepRule-4_ReverseCoedgeInverse">
  <swrl:body>
   <swrl:AtomList>
    <rdf:first>
     <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="#isReverseCoedgeOf"/>
      <swrl:argument2 rdf:resource="#b"/>
      <swrl:argument1 rdf:resource="#a"/>
     </swrl:IndividualPropertyAtom>
    </rdf:first>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
   </swrl:AtomList>
  </swrl:body>
  <swrl:head>
   <swrl:AtomList>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    <rdf:first>
     <swrl:IndividualPropertyAtom>
      <swrl:argument2 rdf:resource="#a"/>
      <swrl:argument1 rdf:resource="#b"/>
      <swrl:propertyPredicate rdf:resource="#hasReverseCoedge"/>
     </swrl:IndividualPropertyAtom>
    </rdf:first>
   </swrl:AtomList>
  </swrl:head>
</swrl:Imp>
<swrl:Variable rdf:ID="size8"/>
<swrl:Variable rdf:ID="set8"/>
<swrl:Variable rdf:ID="p8"/>
<swrl:Variable rdf:ID="set7"/>
<swrl:Imp rdf:about="http://www.CAD-ontology.com/BRepRule-9_SurfaceOfFace">
  <swrl:body>
   <swrl:AtomList>
    <rdf:first>
     <swrl:IndividualPropertyAtom>
      <swrl:argument2 rdf:resource="#b"/>
      <swrl:argument1 rdf:resource="#a"/>
      <swrl:propertyPredicate rdf:resource="#hasGeometryOfSurface"/>
     </swrl:IndividualPropertyAtom>
    </rdf:first>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
   </swrl:AtomList>
  </swrl:body>
```

```
  <swrl:head>
   <swrl:AtomList>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    <rdf:first>
     <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="#isGeometryOfFace"/>
      <swrl:argument1 rdf:resource="#b"/>
      <swrl:argument2 rdf:resource="#a"/>
     </swrl:IndividualPropertyAtom>
    </rdf:first>
   </swrl:AtomList>
  </swrl:head>
</swrl:Imp>
<swrl:Variable rdf:ID="set6"/>
<swrl:Variable rdf:ID="size5"/>
<swrl:Variable rdf:ID="p5"/>
<swrl:Variable rdf:ID="p4"/>
<swrl:Variable rdf:ID="p3"/>
<swrl:Variable rdf:ID="size2"/>
<swrl:Variable rdf:ID="p1"/>
<swrl:Variable rdf:ID="set1"/>
<swrl:Imp rdf:about="http://www.CAD-ontology.com/BRepRule-3_ForwardCoedgeInverse">
  <swrl:head>
   <swrl:AtomList>
    <rdf:first>
     <swrl:IndividualPropertyAtom>
      <swrl:argument1 rdf:resource="#b"/>
      <swrl:argument2 rdf:resource="#a"/>
      <swrl:propertyPredicate rdf:resource="#hasForwardCoedge"/>
     </swrl:IndividualPropertyAtom>
    </rdf:first>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
   </swrl:AtomList>
  </swrl:head>
  <swrl:body>
   <swrl:AtomList>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    <rdf:first>
     <swrl:IndividualPropertyAtom>
      <swrl:argument1 rdf:resource="#a"/>
      <swrl:argument2 rdf:resource="#b"/>
      <swrl:propertyPredicate rdf:resource="#isForwardCoedgeOf"/>
     </swrl:IndividualPropertyAtom>
    </rdf:first>
   </swrl:AtomList>
  </swrl:body>
</swrl:Imp>
<swrl:Imp rdf:about="http://www.CAD-ontology.com/BRepRule-1_StartVertexInverse">
  <swrl:body>
   <swrl:AtomList>
    <rdf:first>
     <swrl:IndividualPropertyAtom>
      <swrl:argument2 rdf:resource="#b"/>
      <swrl:argument1 rdf:resource="#a"/>
      <swrl:propertyPredicate rdf:resource="#hasStartVertex"/>
     </swrl:IndividualPropertyAtom>
```

```
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </swrl:AtomList>
  </swrl:body>
  <swrl:head>
    <swrl:AtomList>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      <rdf:first>
        <swrl:IndividualPropertyAtom>
          <swrl:propertyPredicate rdf:resource="#isStartVertexOf"/>
          <swrl:argument1 rdf:resource="#b"/>
          <swrl:argument2 rdf:resource="#a"/>
        </swrl:IndividualPropertyAtom>
      </rdf:first>
    </swrl:AtomList>
  </swrl:head>
</swrl:Imp>
<swrl:Imp rdf:about="http://www.CAD-ontology.com/BRepRule-8_CurveOfEdge">
  <swrl:head>
    <swrl:AtomList>
      <rdf:first>
        <swrl:IndividualPropertyAtom>
          <swrl:argument1 rdf:resource="#b"/>
          <swrl:propertyPredicate rdf:resource="#isGeometryOfEdge"/>
          <swrl:argument2 rdf:resource="#a"/>
        </swrl:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </swrl:AtomList>
  </swrl:head>
  <swrl:body>
    <swrl:AtomList>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      <rdf:first>
        <swrl:IndividualPropertyAtom>
          <swrl:propertyPredicate rdf:resource="#hasGeometryOfCurve"/>
          <swrl:argument2 rdf:resource="#b"/>
          <swrl:argument1 rdf:resource="#a"/>
        </swrl:IndividualPropertyAtom>
      </rdf:first>
    </swrl:AtomList>
  </swrl:body>
</swrl:Imp>
<swrl:Variable rdf:ID="set5"/>
<swrl:Imp rdf:about="http://www.CAD-ontology.com/BRepRule-2_EndVertexInverse">
  <swrl:head>
    <swrl:AtomList>
      <rdf:first>
        <swrl:IndividualPropertyAtom>
          <swrl:argument2 rdf:resource="#a"/>
          <swrl:argument1 rdf:resource="#b"/>
          <swrl:propertyPredicate rdf:resource="#isEndVertexOf"/>
        </swrl:IndividualPropertyAtom>
      </rdf:first>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </swrl:AtomList>
```

```
    </swrl:head>
    <swrl:body>
      <swrl:AtomList>
        <rdf:first>
          <swrl:IndividualPropertyAtom>
            <swrl:argument2 rdf:resource="#b"/>
            <swrl:argument1 rdf:resource="#a"/>
            <swrl:propertyPredicate rdf:resource="#hasEndVertex"/>
          </swrl:IndividualPropertyAtom>
        </rdf:first>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      </swrl:AtomList>
    </swrl:body>
  </swrl:Imp>
  <swrl:Variable rdf:ID="set4"/>
  <swrl:Variable rdf:ID="set3"/>
  <swrl:Variable rdf:ID="size9"/>
  <swrl:Variable rdf:ID="set2"/>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.4.4, Build 579)  http://protege.stanford.edu -->
```

APPENDIX B

SAMPLE FEATURE XML FILE EXPORTED BY PRO/ENGINEER

```
<?xml version="1.0" encoding="UTF-8"?>
<PRO_E_FEATURE_TREE  AppName="Pro/ENGINEER" AppVersion="Wildfire_4.0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="ProTKFeature.xsd" type="compound">
  <PRO_E_FEATURE_TYPE type="int">PRO_FEAT_PROTRUSION</PRO_E_FEATURE_TYPE>
  <PRO_E_FEATURE_FORM type="int">PRO_EXTRUDE</PRO_E_FEATURE_FORM>
  <PRO_E_STD_FEATURE_NAME type="wstring">EXTRUDE_1</PRO_E_STD_FEATURE_NAME>
  <PRO_E_EXT_SURF_CUT_SOLID_TYPE
type="int">917</PRO_E_EXT_SURF_CUT_SOLID_TYPE>
  <PRO_E_REMOVE_MATERIAL type="int">-1</PRO_E_REMOVE_MATERIAL>
  <PRO_E_IS_SMT_CUT type="int">0</PRO_E_IS_SMT_CUT>
  <PRO_E_SMT_CUT_NORMAL_DIR type="int">0</PRO_E_SMT_CUT_NORMAL_DIR>
  <PRO_E_SKETCHER></PRO_E_SKETCHER>
  <PRO_E_STD_SECTION type="compound">
    <PRO_E_STD_SEC_METHOD type="int">0</PRO_E_STD_SEC_METHOD>
    <PRO_E_SEC_USE_SKETCH type="selection"></PRO_E_SEC_USE_SKETCH>
    <PRO_E_STD_SEC_SELECT type="compound">
      <PRO_E_STD_CURVE_COLLECTION_APPL type="collection">
        <PRO_XML_COLLECTION type="curve">
        </PRO_XML_COLLECTION>
      </PRO_E_STD_CURVE_COLLECTION_APPL>
      <PRO_E_SURF_CHAIN_CMPND type="compound">
        <PRO_E_SURF_CHAIN_METHOD type="int">0</PRO_E_SURF_CHAIN_METHOD>
        <PRO_E_SURF_CHAIN_REF_SURFS type="array">
          <PRO_E_SURF_CHAIN_SURF type="compound">
            <PRO_E_SURF_CHAIN_REF type="selection"></PRO_E_SURF_CHAIN_REF>
          </PRO_E_SURF_CHAIN_SURF>
        </PRO_E_SURF_CHAIN_REF_SURFS>
      </PRO_E_SURF_CHAIN_CMPND>
      <PRO_E_STD_SEC_BLN_VERTS type="selection"></PRO_E_STD_SEC_BLN_VERTS>
    </PRO_E_STD_SEC_SELECT>
    <PRO_E_STD_SEC_SETUP_PLANE type="compound">
      <PRO_E_STD_SEC_PLANE type="selection">
        <PRO_XML_REFERENCE type="reference">
          <PRO_XML_REFERENCE_OWNER
type="owner">ONTOLOGY_DEMO_1.prt</PRO_XML_REFERENCE_OWNER>
          <PRO_XML_REFERENCE_ID type="id">6</PRO_XML_REFERENCE_ID>
          <PRO_XML_REFERENCE_TYPE
type="protype">PRO_SURFACE</PRO_XML_REFERENCE_TYPE>
        </PRO_XML_REFERENCE>
      </PRO_E_STD_SEC_PLANE>
      <PRO_E_STD_SEC_PLANE_VIEW_DIR type="int">1</PRO_E_STD_SEC_PLANE_VIEW_DIR>
      <PRO_E_STD_SEC_PLANE_ORIENT_DIR
type="int">4</PRO_E_STD_SEC_PLANE_ORIENT_DIR>
      <PRO_E_STD_SEC_PLANE_ORIENT_REF type="selection">
        <PRO_XML_REFERENCE type="reference">
```

<PRO_XML_REFERENCE_OWNER type="owner">ONTOLOGY_DEMO_1.prt</PRO_XML_REFERENCE_OWNER>
<PRO_XML_REFERENCE_ID type="id">2</PRO_XML_REFERENCE_ID>
<PRO_XML_REFERENCE_TYPE type="protype">PRO_SURFACE</PRO_XML_REFERENCE_TYPE>
</PRO_XML_REFERENCE>
</PRO_E_STD_SEC_PLANE_ORIENT_REF>
</PRO_E_STD_SEC_SETUP_PLANE>
<PRO_E_SKETCHER type="pointer">**</PRO_E_SKETCHER>
</PRO_E_STD_SECTION>
<PRO_E_FEAT_FORM_IS_THIN type="int">0</PRO_E_FEAT_FORM_IS_THIN>
<PRO_E_STD_MATRLSIDE type="int">0</PRO_E_STD_MATRLSIDE>
<PRO_E_THICKNESS type="double">0.000000</PRO_E_THICKNESS>
<PRO_E_SRF_END_ATTRIBUTES type="int">0</PRO_E_SRF_END_ATTRIBUTES>
<PRO_E_TRIM_QUILT type="selection"></PRO_E_TRIM_QUILT>
<PRO_E_TRIM_QLT_SIDE type="int">0</PRO_E_TRIM_QLT_SIDE>
<PRO_E_STD_DIRECTION type="int">1</PRO_E_STD_DIRECTION>
<PRO_E_STD_EXT_DEPTH type="compound">
<PRO_E_EXT_DEPTH_FROM type="compound">
<PRO_E_EXT_DEPTH_FROM_TYPE type="int">4096</PRO_E_EXT_DEPTH_FROM_TYPE>
<PRO_E_EXT_DEPTH_FROM_VALUE type="double">0.000000</PRO_E_EXT_DEPTH_FROM_VALUE>
<PRO_E_EXT_DEPTH_FROM_REF type="selection"></PRO_E_EXT_DEPTH_FROM_REF>
</PRO_E_EXT_DEPTH_FROM>
<PRO_E_EXT_DEPTH_TO type="compound">
<PRO_E_EXT_DEPTH_TO_TYPE type="int">262144</PRO_E_EXT_DEPTH_TO_TYPE>
<PRO_E_EXT_DEPTH_TO_VALUE type="double">100.000000</PRO_E_EXT_DEPTH_TO_VALUE>
<PRO_E_EXT_DEPTH_TO_REF type="selection"></PRO_E_EXT_DEPTH_TO_REF>
</PRO_E_EXT_DEPTH_TO>
</PRO_E_STD_EXT_DEPTH>
<PRO_E_INT_PARTS></PRO_E_INT_PARTS>
<PRO_E_PATTERN></PRO_E_PATTERN>
<PRO_E_STD_SMT_THICKNESS type="double">0.000000</PRO_E_STD_SMT_THICKNESS>
<PRO_E_STD_SMT_SWAP_DRV_SIDE type="int">0</PRO_E_STD_SMT_SWAP_DRV_SIDE>
<PRO_E_SMT_WALL_SHARPS_TO_BENDS type="int">0</PRO_E_SMT_WALL_SHARPS_TO_BENDS>
<PRO_E_SMT_FILLETS type="compound">
<PRO_E_SMT_FILLETS_SIDE type="int">0</PRO_E_SMT_FILLETS_SIDE>
<PRO_E_SMT_FILLETS_VALUE type="double">0.000000</PRO_E_SMT_FILLETS_VALUE>
</PRO_E_SMT_FILLETS>
<PRO_E_SMT_DEV_LEN_CALCULATION type="compound">
<PRO_E_SMT_DEV_LEN_SOURCE type="int">0</PRO_E_SMT_DEV_LEN_SOURCE>
<PRO_E_SMT_DEV_LEN_Y_FACTOR type="compound">
<PRO_E_SMT_DEV_LEN_Y_FACTOR_TYPE type="int">0</PRO_E_SMT_DEV_LEN_Y_FACTOR_TYPE>
<PRO_E_SMT_DEV_LEN_Y_FACTOR_VALUE type="double">0.000000</PRO_E_SMT_DEV_LEN_Y_FACTOR_VALUE>
</PRO_E_SMT_DEV_LEN_Y_FACTOR>
<PRO_E_SMT_DEV_LEN_BEND_TABLE type="int">0</PRO_E_SMT_DEV_LEN_BEND_TABLE>
</PRO_E_SMT_DEV_LEN_CALCULATION>
</PRO_E_FEATURE_TREE>

APPENDIX C

SAMPLE PART FILE STORED IN SHARED BASE ONTOLOGY FORMAT

```
<?xml version="1.0" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#" xmlns:xsp="http://www.owl-
ontologies.com/2005/08/07/xsp.owl#" xmlns:swrlx="http://swrl.stanford.edu/ontologies/built-
ins/3.3/swrlx.owl#" xmlns:swrlm="http://swrl.stanford.edu/ontologies/built-ins/3.4/swrlm.owl#"
xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:CAD="http://www.CAD-
ontology.com/CAD_ONTOLOGY.owl#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:swrl="http://www.w3.org/2003/11/swrl#" xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns="http://www.CAD-
ontology.com/OntologyFile.owl#" xml:base="http://www.CAD-ontology.com/OntologyFile.owl">
   <!--  Ontology Information -->
   <owl:Ontology rdf:about="">
      <owl:imports rdf:resource="http://www.CAD-ontology.com/CAD_ONTOLOGY.owl#" />
   </owl:Ontology>
   <CAD:Partfile rdf:ID="ONTOLOGY_DEMO_1">
      <CAD:hasReferenceDatumPlane>
         <CAD:ReferenceDatumPlane rdf:ID="RIGHT">
            <CAD:hasDatumPlaneType
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">X_Plane</CAD:hasDatumPlaneType>
         </CAD:ReferenceDatumPlane>
      </CAD:hasReferenceDatumPlane>
      <CAD:hasReferenceDatumPlane>
         <CAD:ReferenceDatumPlane rdf:ID="TOP">
            <CAD:hasDatumPlaneType
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Y_Plane</CAD:hasDatumPlaneType>
         </CAD:ReferenceDatumPlane>
      </CAD:hasReferenceDatumPlane>
      <CAD:hasReferenceDatumPlane>
         <CAD:ReferenceDatumPlane rdf:ID="FRONT">
            <CAD:hasDatumPlaneType
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Z_Plane</CAD:hasDatumPlaneType>
         </CAD:ReferenceDatumPlane>
      </CAD:hasReferenceDatumPlane>
      <CAD:hasReferenceCoordSys>
         <CAD:ReferenceCoordSys rdf:ID="PRT_CSYS_DEF">
            <CAD:hasCoordSysLocationType
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Origin</CAD:hasCoordSysLocationType>
            <CAD:hasCoordSysType
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Cartesian</CAD:hasCoordSysType>
         </CAD:ReferenceCoordSys>
      </CAD:hasReferenceCoordSys>
      <CAD:hasFeature>
         <CAD:Feature rdf:ID="EXTRUDE_1">
            <CAD:hasReferenceSketch>
               <CAD:Sketch rdf:ID="S2D0002">
```

<CAD:hasCoordAxisEntity2D>
<CAD:CoordAxisEntity2D rdf:ID="Section_S2D0002_Entity_0">
<CAD:hasStartPoint>
<CAD:PointEntity2D rdf:ID="Section_S2D0002_Entity_0_StartPoint">
<CAD:hasXCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasXCoord>
<CAD:hasYCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasYCoord>
</CAD:PointEntity2D>
</CAD:hasStartPoint>
<CAD:hasEndPoint>
<CAD:PointEntity2D rdf:ID="Section_S2D0002_Entity_0_EndPoint">
<CAD:hasXCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasXCoord>
<CAD:hasYCoord rdf:datatype="http://www.w3.org/2001/XMLSchema#float">-
100.00</CAD:hasYCoord>
</CAD:PointEntity2D>
</CAD:hasEndPoint>
</CAD:CoordAxisEntity2D>
</CAD:hasCoordAxisEntity2D>
<CAD:hasCoordAxisEntity2D>
<CAD:CoordAxisEntity2D rdf:ID="Section_S2D0002_Entity_1">
<CAD:hasStartPoint>
<CAD:PointEntity2D rdf:ID="Section_S2D0002_Entity_1_StartPoint">
<CAD:hasXCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasXCoord>
<CAD:hasYCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasYCoord>
</CAD:PointEntity2D>
</CAD:hasStartPoint>
<CAD:hasEndPoint>
<CAD:PointEntity2D rdf:ID="Section_S2D0002_Entity_1_EndPoint">
<CAD:hasXCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100.00</CAD:hasXCoord>
<CAD:hasYCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasYCoord>
</CAD:PointEntity2D>
</CAD:hasEndPoint>
</CAD:CoordAxisEntity2D>
</CAD:hasCoordAxisEntity2D>
<CAD:hasLineEntity2D>
<CAD:LineEntity2D rdf:ID="Section_S2D0002_Entity_4">
<CAD:hasStartPoint>
<CAD:PointEntity2D rdf:ID="Section_S2D0002_Entity_4_StartPoint">
<CAD:hasXCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasXCoord>
<CAD:hasYCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasYCoord>
</CAD:PointEntity2D>
</CAD:hasStartPoint>
<CAD:hasEndPoint>
<CAD:PointEntity2D rdf:ID="Section_S2D0002_Entity_4_EndPoint">
<CAD:hasXCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">200.00</CAD:hasXCoord>
<CAD:hasYCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasYCoord>

```
                    </CAD:PointEntity2D>
                  </CAD:hasEndPoint>
                </CAD:LineEntity2D>
              </CAD:hasLineEntity2D>
              <CAD:hasLineEntity2D>
                <CAD:LineEntity2D rdf:ID="Section_S2D0002_Entity_5">
                  <CAD:hasStartPoint>
                    <CAD:PointEntity2D rdf:ID="Section_S2D0002_Entity_5_StartPoint">
                      <CAD:hasXCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">200.00</CAD:hasXCoord>
                      <CAD:hasYCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasYCoord>
                    </CAD:PointEntity2D>
                  </CAD:hasStartPoint>
                  <CAD:hasEndPoint>
                    <CAD:PointEntity2D rdf:ID="Section_S2D0002_Entity_5_EndPoint">
                      <CAD:hasXCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">200.00</CAD:hasXCoord>
                      <CAD:hasYCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100.00</CAD:hasYCoord>
                    </CAD:PointEntity2D>
                  </CAD:hasEndPoint>
                </CAD:LineEntity2D>
              </CAD:hasLineEntity2D>
              <CAD:hasLineEntity2D>
                <CAD:LineEntity2D rdf:ID="Section_S2D0002_Entity_6">
                  <CAD:hasStartPoint>
                    <CAD:PointEntity2D rdf:ID="Section_S2D0002_Entity_6_StartPoint">
                      <CAD:hasXCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">200.00</CAD:hasXCoord>
                      <CAD:hasYCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100.00</CAD:hasYCoord>
                    </CAD:PointEntity2D>
                  </CAD:hasStartPoint>
                  <CAD:hasEndPoint>
                    <CAD:PointEntity2D rdf:ID="Section_S2D0002_Entity_6_EndPoint">
                      <CAD:hasXCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasXCoord>
                      <CAD:hasYCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100.00</CAD:hasYCoord>
                    </CAD:PointEntity2D>
                  </CAD:hasEndPoint>
                </CAD:LineEntity2D>
              </CAD:hasLineEntity2D>
              <CAD:hasLineEntity2D>
                <CAD:LineEntity2D rdf:ID="Section_S2D0002_Entity_7">
                  <CAD:hasStartPoint>
                    <CAD:PointEntity2D rdf:ID="Section_S2D0002_Entity_7_StartPoint">
                      <CAD:hasXCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasXCoord>
                      <CAD:hasYCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100.00</CAD:hasYCoord>
                    </CAD:PointEntity2D>
                  </CAD:hasStartPoint>
                  <CAD:hasEndPoint>
                    <CAD:PointEntity2D rdf:ID="Section_S2D0002_Entity_7_EndPoint">
```

<CAD:hasXCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasXCoord>
<CAD:hasYCoord
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.00</CAD:hasYCoord>
</CAD:PointEntity2D>
</CAD:hasEndPoint>
</CAD:LineEntity2D>
</CAD:hasLineEntity2D>
<CAD:hasNumberCoordAxisEntity2D
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2</CAD:hasNumberCoordAxisEntity2D>
<CAD:hasNumberPointEntity2D
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberPointEntity2D>
<CAD:hasNumberCoordSys2D
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberCoordSys2D>
<CAD:hasNumberLineEntity2D
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">4</CAD:hasNumberLineEntity2D>
<CAD:hasNumberConstructionLineEntity2D
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberConstructionLineEntity2D
>
<CAD:hasNumberArcEntity2D
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberArcEntity2D>
<CAD:hasNumberCircleEntity2D
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberCircleEntity2D>
<CAD:hasNumberPolylineEntity2D
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberPolylineEntity2D>
<CAD:hasNumberSplineEntity2D
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberSplineEntity2D>
<CAD:hasNumberEllipseEntity2D
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberEllipseEntity2D>
<CAD:hasNumberConicEntity2D
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberConicEntity2D>
<CAD:hasSketchConstraint>
<CAD:SamePointConstraint2D rdf:ID="Section_S2D0002_Constraint_0">
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_4_EndPoint" />
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_5_StartPoint" />
</CAD:SamePointConstraint2D>
</CAD:hasSketchConstraint>
<CAD:hasSketchConstraint>
<CAD:SamePointConstraint2D rdf:ID="Section_S2D0002_Constraint_1">
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_5_EndPoint" />
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_6_StartPoint" />
</CAD:SamePointConstraint2D>
</CAD:hasSketchConstraint>
<CAD:hasSketchConstraint>
<CAD:SamePointConstraint2D rdf:ID="Section_S2D0002_Constraint_2">
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_6_EndPoint" />
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_7_StartPoint" />
</CAD:SamePointConstraint2D>
</CAD:hasSketchConstraint>
<CAD:hasSketchConstraint>
<CAD:SamePointConstraint2D rdf:ID="Section_S2D0002_Constraint_3">
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_7_EndPoint" />
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_4_StartPoint" />
</CAD:SamePointConstraint2D>
</CAD:hasSketchConstraint>
<CAD:hasSketchConstraint>

<CAD:HorizontalEntityConstraint2D rdf:ID="Section_S2D0002_Constraint_4">
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_4" />
</CAD:HorizontalEntityConstraint2D>
</CAD:hasSketchConstraint>
<CAD:hasSketchConstraint>
<CAD:HorizontalEntityConstraint2D rdf:ID="Section_S2D0002_Constraint_5">
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_6" />
</CAD:HorizontalEntityConstraint2D>
</CAD:hasSketchConstraint>
<CAD:hasSketchConstraint>
<CAD:VerticalEntityConstraint2D rdf:ID="Section_S2D0002_Constraint_6">
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_5" />
</CAD:VerticalEntityConstraint2D>
</CAD:hasSketchConstraint>
<CAD:hasSketchConstraint>
<CAD:VerticalEntityConstraint2D rdf:ID="Section_S2D0002_Constraint_7">
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_7" />
</CAD:VerticalEntityConstraint2D>
</CAD:hasSketchConstraint>
<CAD:hasSketchConstraint>
<CAD:PointOnEntityConstraint2D rdf:ID="Section_S2D0002_Constraint_8">
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_0" />
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_4_StartPoint" />
</CAD:PointOnEntityConstraint2D>
</CAD:hasSketchConstraint>
<CAD:hasSketchConstraint>
<CAD:PointOnEntityConstraint2D rdf:ID="Section_S2D0002_Constraint_9">
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_1" />
<CAD:isConstraintOfEntity rdf:resource="#Section_S2D0002_Entity_4_StartPoint" />
</CAD:PointOnEntityConstraint2D>
</CAD:hasSketchConstraint>
<CAD:hasSketchDimension>
<CAD:LineLengthDimension2D rdf:ID="Section_S2D0002_Dimension_0">
<CAD:hasDimensionValue
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100.00</CAD:hasDimensionValue>
<CAD:isDimensionOfEntity rdf:resource="#Section_S2D0002_Entity_5" />
</CAD:LineLengthDimension2D>
</CAD:hasSketchDimension>
<CAD:hasSketchDimension>
<CAD:LineLengthDimension2D rdf:ID="Section_S2D0002_Dimension_1">
<CAD:hasDimensionValue
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">200.00</CAD:hasDimensionValue>
<CAD:isDimensionOfEntity rdf:resource="#Section_S2D0002_Entity_4" />
</CAD:LineLengthDimension2D>
</CAD:hasSketchDimension>
</CAD:Sketch>
</CAD:hasReferenceSketch>
<CAD:hasExtSurfaceType
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Solid</CAD:hasExtSurfaceType>
<CAD:removesMaterial
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">False</CAD:removesMaterial>
<CAD:isThinShellPart
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">False</CAD:isThinShellPart>
<CAD:hasDepthDimension
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100.000000</CAD:hasDepthDimension>
<CAD:hasPrimaryReferenceDatumPlane rdf:resource="#FRONT" />

<CAD:hasSecondaryReferenceDatumPlane rdf:resource="#RIGHT" />
<CAD:hasNumberReferenceSketchInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</CAD:hasNumberReferenceSketchInstances>
<CAD:createsShell>
<CAD:Shell rdf:ID="BRepID-3" />
</CAD:createsShell>
<CAD:createsFace>
<CAD:Face rdf:ID="BRepID-4">
<CAD:hasLoop rdf:resource="#BRepID-6" />
<CAD:isFaceOf rdf:resource="#BRepID-3" />
<CAD:hasGeometryOfSurface rdf:resource="#BRepID-7" />
</CAD:Face>
</CAD:createsFace>
<CAD:createsFace>
<CAD:Face rdf:ID="BRepID-5">
<CAD:hasLoop rdf:resource="#BRepID-9" />
<CAD:isFaceOf rdf:resource="#BRepID-3" />
<CAD:hasGeometryOfSurface rdf:resource="#BRepID-10" />
</CAD:Face>
</CAD:createsFace>
<CAD:createsLoop>
<CAD:Loop rdf:ID="BRepID-6">
<CAD:isLoopOf rdf:resource="#BRepID-4" />
</CAD:Loop>
</CAD:createsLoop>
<CAD:createsPlanarSurface>
<CAD:PlanarSurface rdf:ID="BRepID-7" />
</CAD:createsPlanarSurface>
<CAD:createsFace>
<CAD:Face rdf:ID="BRepID-8">
<CAD:hasLoop rdf:resource="#BRepID-13" />
<CAD:isFaceOf rdf:resource="#BRepID-3" />
<CAD:hasGeometryOfSurface rdf:resource="#BRepID-14" />
</CAD:Face>
</CAD:createsFace>
<CAD:createsLoop>
<CAD:Loop rdf:ID="BRepID-9">
<CAD:isLoopOf rdf:resource="#BRepID-5" />
</CAD:Loop>
</CAD:createsLoop>
<CAD:createsPlanarSurface>
<CAD:PlanarSurface rdf:ID="BRepID-10" />
</CAD:createsPlanarSurface>
<CAD:createsCoedge>
<CAD:Coedge rdf:ID="BRepID-11">
<CAD:hasNextCoedge rdf:resource="#BRepID-16" />
<CAD:hasPreviousCoedge rdf:resource="#BRepID-17" />
<CAD:hasOppositeCoedge rdf:resource="#BRepID-18" />
<CAD:isReverseCoedgeOf rdf:resource="#BRepID-19" />
<CAD:isCoedgeInLoop rdf:resource="#BRepID-6" />
</CAD:Coedge>
</CAD:createsCoedge>
<CAD:createsFace>
<CAD:Face rdf:ID="BRepID-12">
<CAD:hasLoop rdf:resource="#BRepID-21" />

```
      <CAD:isFaceOf rdf:resource="#BRepID-3" />
      <CAD:hasGeometryOfSurface rdf:resource="#BRepID-22" />
    </CAD:Face>
  </CAD:createsFace>
  <CAD:createsLoop>
    <CAD:Loop rdf:ID="BRepID-13">
      <CAD:isLoopOf rdf:resource="#BRepID-8" />
    </CAD:Loop>
  </CAD:createsLoop>
  <CAD:createsPlanarSurface>
    <CAD:PlanarSurface rdf:ID="BRepID-14" />
  </CAD:createsPlanarSurface>
  <CAD:createsCoedge>
    <CAD:Coedge rdf:ID="BRepID-15">
      <CAD:hasNextCoedge rdf:resource="#BRepID-24" />
      <CAD:hasPreviousCoedge rdf:resource="#BRepID-25" />
      <CAD:hasOppositeCoedge rdf:resource="#BRepID-26" />
      <CAD:isForwardCoedgeOf rdf:resource="#BRepID-27" />
      <CAD:isCoedgeInLoop rdf:resource="#BRepID-9" />
    </CAD:Coedge>
  </CAD:createsCoedge>
  <CAD:createsCoedge>
    <CAD:Coedge rdf:ID="BRepID-16">
      <CAD:hasNextCoedge rdf:resource="#BRepID-28" />
      <CAD:hasPreviousCoedge rdf:resource="#BRepID-11" />
      <CAD:hasOppositeCoedge rdf:resource="#BRepID-29" />
      <CAD:isReverseCoedgeOf rdf:resource="#BRepID-30" />
      <CAD:isCoedgeInLoop rdf:resource="#BRepID-6" />
    </CAD:Coedge>
  </CAD:createsCoedge>
  <CAD:createsCoedge>
    <CAD:Coedge rdf:ID="BRepID-17">
      <CAD:hasNextCoedge rdf:resource="#BRepID-11" />
      <CAD:hasPreviousCoedge rdf:resource="#BRepID-28" />
      <CAD:hasOppositeCoedge rdf:resource="#BRepID-25" />
      <CAD:isReverseCoedgeOf rdf:resource="#BRepID-31" />
      <CAD:isCoedgeInLoop rdf:resource="#BRepID-6" />
    </CAD:Coedge>
  </CAD:createsCoedge>
  <CAD:createsCoedge>
    <CAD:Coedge rdf:ID="BRepID-18">
      <CAD:hasNextCoedge rdf:resource="#BRepID-23" />
      <CAD:hasPreviousCoedge rdf:resource="#BRepID-32" />
      <CAD:hasOppositeCoedge rdf:resource="#BRepID-11" />
      <CAD:isForwardCoedgeOf rdf:resource="#BRepID-19" />
      <CAD:isCoedgeInLoop rdf:resource="#BRepID-13" />
    </CAD:Coedge>
  </CAD:createsCoedge>
  <CAD:createsEdge>
    <CAD:Edge rdf:ID="BRepID-19">
      <CAD:hasStartVertex rdf:resource="#BRepID-34" />
      <CAD:hasEndVertex rdf:resource="#BRepID-35" />
      <CAD:hasGeometryOfCurve rdf:resource="#BRepID-36" />
    </CAD:Edge>
  </CAD:createsEdge>
  <CAD:createsFace>
```

```
    <CAD:Face rdf:ID="BRepID-20">
      <CAD:hasLoop rdf:resource="#BRepID-38" />
      <CAD:isFaceOf rdf:resource="#BRepID-3" />
      <CAD:hasGeometryOfSurface rdf:resource="#BRepID-39" />
    </CAD:Face>
  </CAD:createsFace>
  <CAD:createsLoop>
    <CAD:Loop rdf:ID="BRepID-21">
      <CAD:isLoopOf rdf:resource="#BRepID-12" />
    </CAD:Loop>
  </CAD:createsLoop>
  <CAD:createsPlanarSurface>
    <CAD:PlanarSurface rdf:ID="BRepID-22" />
  </CAD:createsPlanarSurface>
  <CAD:createsCoedge>
    <CAD:Coedge rdf:ID="BRepID-23">
      <CAD:hasNextCoedge rdf:resource="#BRepID-41" />
      <CAD:hasPreviousCoedge rdf:resource="#BRepID-18" />
      <CAD:hasOppositeCoedge rdf:resource="#BRepID-42" />
      <CAD:isForwardCoedgeOf rdf:resource="#BRepID-43" />
      <CAD:isCoedgeInLoop rdf:resource="#BRepID-13" />
    </CAD:Coedge>
  </CAD:createsCoedge>
  <CAD:createsCoedge>
    <CAD:Coedge rdf:ID="BRepID-24">
      <CAD:hasNextCoedge rdf:resource="#BRepID-42" />
      <CAD:hasPreviousCoedge rdf:resource="#BRepID-15" />
      <CAD:hasOppositeCoedge rdf:resource="#BRepID-44" />
      <CAD:isReverseCoedgeOf rdf:resource="#BRepID-45" />
      <CAD:isCoedgeInLoop rdf:resource="#BRepID-9" />
    </CAD:Coedge>
  </CAD:createsCoedge>
  <CAD:createsCoedge>
    <CAD:Coedge rdf:ID="BRepID-25">
      <CAD:hasNextCoedge rdf:resource="#BRepID-15" />
      <CAD:hasPreviousCoedge rdf:resource="#BRepID-42" />
      <CAD:hasOppositeCoedge rdf:resource="#BRepID-17" />
      <CAD:isForwardCoedgeOf rdf:resource="#BRepID-31" />
      <CAD:isCoedgeInLoop rdf:resource="#BRepID-9" />
    </CAD:Coedge>
  </CAD:createsCoedge>
  <CAD:createsCoedge>
    <CAD:Coedge rdf:ID="BRepID-26">
      <CAD:hasNextCoedge rdf:resource="#BRepID-46" />
      <CAD:hasPreviousCoedge rdf:resource="#BRepID-47" />
      <CAD:hasOppositeCoedge rdf:resource="#BRepID-15" />
      <CAD:isReverseCoedgeOf rdf:resource="#BRepID-27" />
      <CAD:isCoedgeInLoop rdf:resource="#BRepID-38" />
    </CAD:Coedge>
  </CAD:createsCoedge>
  <CAD:createsEdge>
    <CAD:Edge rdf:ID="BRepID-27">
      <CAD:hasStartVertex rdf:resource="#BRepID-49" />
      <CAD:hasEndVertex rdf:resource="#BRepID-50" />
      <CAD:hasGeometryOfCurve rdf:resource="#BRepID-51" />
    </CAD:Edge>
```

```
</CAD:createsEdge>
<CAD:createsCoedge>
  <CAD:Coedge rdf:ID="BRepID-28">
    <CAD:hasNextCoedge rdf:resource="#BRepID-17" />
    <CAD:hasPreviousCoedge rdf:resource="#BRepID-16" />
    <CAD:hasOppositeCoedge rdf:resource="#BRepID-46" />
    <CAD:isReverseCoedgeOf rdf:resource="#BRepID-52" />
    <CAD:isCoedgeInLoop rdf:resource="#BRepID-6" />
  </CAD:Coedge>
</CAD:createsCoedge>
<CAD:createsCoedge>
  <CAD:Coedge rdf:ID="BRepID-29">
    <CAD:hasNextCoedge rdf:resource="#BRepID-40" />
    <CAD:hasPreviousCoedge rdf:resource="#BRepID-53" />
    <CAD:hasOppositeCoedge rdf:resource="#BRepID-16" />
    <CAD:isForwardCoedgeOf rdf:resource="#BRepID-30" />
    <CAD:isCoedgeInLoop rdf:resource="#BRepID-21" />
  </CAD:Coedge>
</CAD:createsCoedge>
<CAD:createsEdge>
  <CAD:Edge rdf:ID="BRepID-30">
    <CAD:hasStartVertex rdf:resource="#BRepID-55" />
    <CAD:hasEndVertex rdf:resource="#BRepID-34" />
    <CAD:hasGeometryOfCurve rdf:resource="#BRepID-56" />
  </CAD:Edge>
</CAD:createsEdge>
<CAD:createsEdge>
  <CAD:Edge rdf:ID="BRepID-31">
    <CAD:hasStartVertex rdf:resource="#BRepID-35" />
    <CAD:hasEndVertex rdf:resource="#BRepID-49" />
    <CAD:hasGeometryOfCurve rdf:resource="#BRepID-58" />
  </CAD:Edge>
</CAD:createsEdge>
<CAD:createsCoedge>
  <CAD:Coedge rdf:ID="BRepID-32">
    <CAD:hasNextCoedge rdf:resource="#BRepID-18" />
    <CAD:hasPreviousCoedge rdf:resource="#BRepID-41" />
    <CAD:hasOppositeCoedge rdf:resource="#BRepID-40" />
    <CAD:isReverseCoedgeOf rdf:resource="#BRepID-59" />
    <CAD:isCoedgeInLoop rdf:resource="#BRepID-13" />
  </CAD:Coedge>
</CAD:createsCoedge>
<CAD:createsVertex>
  <CAD:Vertex rdf:ID="BRepID-34">
    <CAD:hasGeometryOfPoint rdf:resource="#BRepID-60" />
  </CAD:Vertex>
</CAD:createsVertex>
<CAD:createsVertex>
  <CAD:Vertex rdf:ID="BRepID-35">
    <CAD:hasGeometryOfPoint rdf:resource="#BRepID-61" />
  </CAD:Vertex>
</CAD:createsVertex>
<CAD:createsLinearCurve>
  <CAD:LinearCurve rdf:ID="BRepID-36" />
</CAD:createsLinearCurve>
<CAD:createsFace>
```

```
    <CAD:Face rdf:ID="BRepID-37">
      <CAD:hasLoop rdf:resource="#BRepID-62" />
      <CAD:isFaceOf rdf:resource="#BRepID-3" />
      <CAD:hasGeometryOfSurface rdf:resource="#BRepID-63" />
    </CAD:Face>
  </CAD:createsFace>
  <CAD:createsLoop>
    <CAD:Loop rdf:ID="BRepID-38">
      <CAD:isLoopOf rdf:resource="#BRepID-20" />
    </CAD:Loop>
  </CAD:createsLoop>
  <CAD:createsPlanarSurface>
    <CAD:PlanarSurface rdf:ID="BRepID-39" />
  </CAD:createsPlanarSurface>
  <CAD:createsCoedge>
    <CAD:Coedge rdf:ID="BRepID-40">
      <CAD:hasNextCoedge rdf:resource="#BRepID-65" />
      <CAD:hasPreviousCoedge rdf:resource="#BRepID-29" />
      <CAD:hasOppositeCoedge rdf:resource="#BRepID-32" />
      <CAD:isForwardCoedgeOf rdf:resource="#BRepID-59" />
      <CAD:isCoedgeInLoop rdf:resource="#BRepID-21" />
    </CAD:Coedge>
  </CAD:createsCoedge>
  <CAD:createsCoedge>
    <CAD:Coedge rdf:ID="BRepID-41">
      <CAD:hasNextCoedge rdf:resource="#BRepID-32" />
      <CAD:hasPreviousCoedge rdf:resource="#BRepID-23" />
      <CAD:hasOppositeCoedge rdf:resource="#BRepID-66" />
      <CAD:isReverseCoedgeOf rdf:resource="#BRepID-67" />
      <CAD:isCoedgeInLoop rdf:resource="#BRepID-13" />
    </CAD:Coedge>
  </CAD:createsCoedge>
  <CAD:createsCoedge>
    <CAD:Coedge rdf:ID="BRepID-42">
      <CAD:hasNextCoedge rdf:resource="#BRepID-25" />
      <CAD:hasPreviousCoedge rdf:resource="#BRepID-24" />
      <CAD:hasOppositeCoedge rdf:resource="#BRepID-23" />
      <CAD:isReverseCoedgeOf rdf:resource="#BRepID-43" />
      <CAD:isCoedgeInLoop rdf:resource="#BRepID-9" />
    </CAD:Coedge>
  </CAD:createsCoedge>
  <CAD:createsEdge>
    <CAD:Edge rdf:ID="BRepID-43">
      <CAD:hasStartVertex rdf:resource="#BRepID-35" />
      <CAD:hasEndVertex rdf:resource="#BRepID-69" />
      <CAD:hasGeometryOfCurve rdf:resource="#BRepID-70" />
    </CAD:Edge>
  </CAD:createsEdge>
  <CAD:createsCoedge>
    <CAD:Coedge rdf:ID="BRepID-44">
      <CAD:hasNextCoedge rdf:resource="#BRepID-71" />
      <CAD:hasPreviousCoedge rdf:resource="#BRepID-66" />
      <CAD:hasOppositeCoedge rdf:resource="#BRepID-24" />
      <CAD:isForwardCoedgeOf rdf:resource="#BRepID-45" />
      <CAD:isCoedgeInLoop rdf:resource="#BRepID-62" />
    </CAD:Coedge>
```

```
</CAD:createsCoedge>
<CAD:createsEdge>
  <CAD:Edge rdf:ID="BRepID-45">
    <CAD:hasStartVertex rdf:resource="#BRepID-69" />
    <CAD:hasEndVertex rdf:resource="#BRepID-50" />
    <CAD:hasGeometryOfCurve rdf:resource="#BRepID-73" />
  </CAD:Edge>
</CAD:createsEdge>
<CAD:createsCoedge>
  <CAD:Coedge rdf:ID="BRepID-46">
    <CAD:hasNextCoedge rdf:resource="#BRepID-64" />
    <CAD:hasPreviousCoedge rdf:resource="#BRepID-26" />
    <CAD:hasOppositeCoedge rdf:resource="#BRepID-28" />
    <CAD:isForwardCoedgeOf rdf:resource="#BRepID-52" />
    <CAD:isCoedgeInLoop rdf:resource="#BRepID-38" />
  </CAD:Coedge>
</CAD:createsCoedge>
<CAD:createsCoedge>
  <CAD:Coedge rdf:ID="BRepID-47">
    <CAD:hasNextCoedge rdf:resource="#BRepID-26" />
    <CAD:hasPreviousCoedge rdf:resource="#BRepID-64" />
    <CAD:hasOppositeCoedge rdf:resource="#BRepID-71" />
    <CAD:isReverseCoedgeOf rdf:resource="#BRepID-74" />
    <CAD:isCoedgeInLoop rdf:resource="#BRepID-38" />
  </CAD:Coedge>
</CAD:createsCoedge>
<CAD:createsVertex>
  <CAD:Vertex rdf:ID="BRepID-49">
    <CAD:hasGeometryOfPoint rdf:resource="#BRepID-75" />
  </CAD:Vertex>
</CAD:createsVertex>
<CAD:createsVertex>
  <CAD:Vertex rdf:ID="BRepID-50">
    <CAD:hasGeometryOfPoint rdf:resource="#BRepID-76" />
  </CAD:Vertex>
</CAD:createsVertex>
<CAD:createsLinearCurve>
  <CAD:LinearCurve rdf:ID="BRepID-51" />
</CAD:createsLinearCurve>
<CAD:createsEdge>
  <CAD:Edge rdf:ID="BRepID-52">
    <CAD:hasStartVertex rdf:resource="#BRepID-49" />
    <CAD:hasEndVertex rdf:resource="#BRepID-55" />
    <CAD:hasGeometryOfCurve rdf:resource="#BRepID-78" />
  </CAD:Edge>
</CAD:createsEdge>
<CAD:createsCoedge>
  <CAD:Coedge rdf:ID="BRepID-53">
    <CAD:hasNextCoedge rdf:resource="#BRepID-29" />
    <CAD:hasPreviousCoedge rdf:resource="#BRepID-65" />
    <CAD:hasOppositeCoedge rdf:resource="#BRepID-64" />
    <CAD:isReverseCoedgeOf rdf:resource="#BRepID-79" />
    <CAD:isCoedgeInLoop rdf:resource="#BRepID-21" />
  </CAD:Coedge>
</CAD:createsCoedge>
<CAD:createsVertex>
```

```
            <CAD:Vertex rdf:ID="BRepID-55">
              <CAD:hasGeometryOfPoint rdf:resource="#BRepID-80" />
            </CAD:Vertex>
          </CAD:createsVertex>
          <CAD:createsLinearCurve>
            <CAD:LinearCurve rdf:ID="BRepID-56" />
          </CAD:createsLinearCurve>
          <CAD:createsLinearCurve>
            <CAD:LinearCurve rdf:ID="BRepID-58" />
          </CAD:createsLinearCurve>
          <CAD:createsEdge>
            <CAD:Edge rdf:ID="BRepID-59">
              <CAD:hasStartVertex rdf:resource="#BRepID-34" />
              <CAD:hasEndVertex rdf:resource="#BRepID-82" />
              <CAD:hasGeometryOfCurve rdf:resource="#BRepID-83" />
            </CAD:Edge>
          </CAD:createsEdge>
          <CAD:createsPoint>
            <CAD:Point rdf:ID="BRepID-60">
              <CAD:isAtXCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</CAD:isAtXCoordinate>
              <CAD:isAtYCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100</CAD:isAtYCoordinate>
              <CAD:isAtZCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</CAD:isAtZCoordinate>
            </CAD:Point>
          </CAD:createsPoint>
          <CAD:createsPoint>
            <CAD:Point rdf:ID="BRepID-61">
              <CAD:isAtXCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</CAD:isAtXCoordinate>
              <CAD:isAtYCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</CAD:isAtYCoordinate>
              <CAD:isAtZCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</CAD:isAtZCoordinate>
            </CAD:Point>
          </CAD:createsPoint>
          <CAD:createsLoop>
            <CAD:Loop rdf:ID="BRepID-62">
              <CAD:isLoopOf rdf:resource="#BRepID-37" />
            </CAD:Loop>
          </CAD:createsLoop>
          <CAD:createsPlanarSurface>
            <CAD:PlanarSurface rdf:ID="BRepID-63" />
          </CAD:createsPlanarSurface>
          <CAD:createsCoedge>
            <CAD:Coedge rdf:ID="BRepID-64">
              <CAD:hasNextCoedge rdf:resource="#BRepID-47" />
              <CAD:hasPreviousCoedge rdf:resource="#BRepID-46" />
              <CAD:hasOppositeCoedge rdf:resource="#BRepID-53" />
              <CAD:isForwardCoedgeOf rdf:resource="#BRepID-79" />
              <CAD:isCoedgeInLoop rdf:resource="#BRepID-38" />
            </CAD:Coedge>
          </CAD:createsCoedge>
          <CAD:createsCoedge>
            <CAD:Coedge rdf:ID="BRepID-65">
```

<CAD:hasNextCoedge rdf:resource="#BRepID-53" />
<CAD:hasPreviousCoedge rdf:resource="#BRepID-40" />
<CAD:hasOppositeCoedge rdf:resource="#BRepID-84" />
<CAD:isReverseCoedgeOf rdf:resource="#BRepID-85" />
<CAD:isCoedgeInLoop rdf:resource="#BRepID-21" />
</CAD:Coedge>
</CAD:createsCoedge>
<CAD:createsCoedge>
<CAD:Coedge rdf:ID="BRepID-66">
<CAD:hasNextCoedge rdf:resource="#BRepID-44" />
<CAD:hasPreviousCoedge rdf:resource="#BRepID-84" />
<CAD:hasOppositeCoedge rdf:resource="#BRepID-41" />
<CAD:isForwardCoedgeOf rdf:resource="#BRepID-67" />
<CAD:isCoedgeInLoop rdf:resource="#BRepID-62" />
</CAD:Coedge>
</CAD:createsCoedge>
<CAD:createsEdge>
<CAD:Edge rdf:ID="BRepID-67">
<CAD:hasStartVertex rdf:resource="#BRepID-82" />
<CAD:hasEndVertex rdf:resource="#BRepID-69" />
<CAD:hasGeometryOfCurve rdf:resource="#BRepID-87" />
</CAD:Edge>
</CAD:createsEdge>
<CAD:createsVertex>
<CAD:Vertex rdf:ID="BRepID-69">
<CAD:hasGeometryOfPoint rdf:resource="#BRepID-88" />
</CAD:Vertex>
</CAD:createsVertex>
<CAD:createsLinearCurve>
<CAD:LinearCurve rdf:ID="BRepID-70" />
</CAD:createsLinearCurve>
<CAD:createsCoedge>
<CAD:Coedge rdf:ID="BRepID-71">
<CAD:hasNextCoedge rdf:resource="#BRepID-84" />
<CAD:hasPreviousCoedge rdf:resource="#BRepID-44" />
<CAD:hasOppositeCoedge rdf:resource="#BRepID-47" />
<CAD:isForwardCoedgeOf rdf:resource="#BRepID-74" />
<CAD:isCoedgeInLoop rdf:resource="#BRepID-62" />
</CAD:Coedge>
</CAD:createsCoedge>
<CAD:createsLinearCurve>
<CAD:LinearCurve rdf:ID="BRepID-73" />
</CAD:createsLinearCurve>
<CAD:createsEdge>
<CAD:Edge rdf:ID="BRepID-74">
<CAD:hasStartVertex rdf:resource="#BRepID-50" />
<CAD:hasEndVertex rdf:resource="#BRepID-90" />
<CAD:hasGeometryOfCurve rdf:resource="#BRepID-91" />
</CAD:Edge>
</CAD:createsEdge>
<CAD:createsPoint>
<CAD:Point rdf:ID="BRepID-75">
<CAD:isAtXCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">200</CAD:isAtXCoordinate>
<CAD:isAtYCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</CAD:isAtYCoordinate>

<CAD:isAtZCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</CAD:isAtZCoordinate>
</CAD:Point>
</CAD:createsPoint>
<CAD:createsPoint>
<CAD:Point rdf:ID="BRepID-76">
<CAD:isAtXCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">200</CAD:isAtXCoordinate>
<CAD:isAtYCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</CAD:isAtYCoordinate>
<CAD:isAtZCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100</CAD:isAtZCoordinate>
</CAD:Point>
</CAD:createsPoint>
<CAD:createsLinearCurve>
<CAD:LinearCurve rdf:ID="BRepID-78" />
</CAD:createsLinearCurve>
<CAD:createsEdge>
<CAD:Edge rdf:ID="BRepID-79">
<CAD:hasStartVertex rdf:resource="#BRepID-55" />
<CAD:hasEndVertex rdf:resource="#BRepID-90" />
<CAD:hasGeometryOfCurve rdf:resource="#BRepID-93" />
</CAD:Edge>
</CAD:createsEdge>
<CAD:createsPoint>
<CAD:Point rdf:ID="BRepID-80">
<CAD:isAtXCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">200</CAD:isAtXCoordinate>
<CAD:isAtYCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100</CAD:isAtYCoordinate>
<CAD:isAtZCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</CAD:isAtZCoordinate>
</CAD:Point>
</CAD:createsPoint>
<CAD:createsVertex>
<CAD:Vertex rdf:ID="BRepID-82">
<CAD:hasGeometryOfPoint rdf:resource="#BRepID-94" />
</CAD:Vertex>
</CAD:createsVertex>
<CAD:createsLinearCurve>
<CAD:LinearCurve rdf:ID="BRepID-83" />
</CAD:createsLinearCurve>
<CAD:createsCoedge>
<CAD:Coedge rdf:ID="BRepID-84">
<CAD:hasNextCoedge rdf:resource="#BRepID-66" />
<CAD:hasPreviousCoedge rdf:resource="#BRepID-71" />
<CAD:hasOppositeCoedge rdf:resource="#BRepID-65" />
<CAD:isForwardCoedgeOf rdf:resource="#BRepID-85" />
<CAD:isCoedgeInLoop rdf:resource="#BRepID-62" />
</CAD:Coedge>
</CAD:createsCoedge>
<CAD:createsEdge>
<CAD:Edge rdf:ID="BRepID-85">
<CAD:hasStartVertex rdf:resource="#BRepID-90" />
<CAD:hasEndVertex rdf:resource="#BRepID-82" />
<CAD:hasGeometryOfCurve rdf:resource="#BRepID-96" />

```
            </CAD:Edge>
          </CAD:createsEdge>
          <CAD:createsLinearCurve>
            <CAD:LinearCurve rdf:ID="BRepID-87" />
          </CAD:createsLinearCurve>
          <CAD:createsPoint>
            <CAD:Point rdf:ID="BRepID-88">
              <CAD:isAtXCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</CAD:isAtXCoordinate>
              <CAD:isAtYCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</CAD:isAtYCoordinate>
              <CAD:isAtZCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100</CAD:isAtZCoordinate>
            </CAD:Point>
          </CAD:createsPoint>
          <CAD:createsVertex>
            <CAD:Vertex rdf:ID="BRepID-90">
              <CAD:hasGeometryOfPoint rdf:resource="#BRepID-97" />
            </CAD:Vertex>
          </CAD:createsVertex>
          <CAD:createsLinearCurve>
            <CAD:LinearCurve rdf:ID="BRepID-91" />
          </CAD:createsLinearCurve>
          <CAD:createsLinearCurve>
            <CAD:LinearCurve rdf:ID="BRepID-93" />
          </CAD:createsLinearCurve>
          <CAD:createsPoint>
            <CAD:Point rdf:ID="BRepID-94">
              <CAD:isAtXCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</CAD:isAtXCoordinate>
              <CAD:isAtYCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100</CAD:isAtYCoordinate>
              <CAD:isAtZCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100</CAD:isAtZCoordinate>
            </CAD:Point>
          </CAD:createsPoint>
          <CAD:createsLinearCurve>
            <CAD:LinearCurve rdf:ID="BRepID-96" />
          </CAD:createsLinearCurve>
          <CAD:createsPoint>
            <CAD:Point rdf:ID="BRepID-97">
              <CAD:isAtXCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">200</CAD:isAtXCoordinate>
              <CAD:isAtYCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100</CAD:isAtYCoordinate>
              <CAD:isAtZCoordinate
rdf:datatype="http://www.w3.org/2001/XMLSchema#float">100</CAD:isAtZCoordinate>
            </CAD:Point>
          </CAD:createsPoint>
          <CAD:hasNumberCreatePointInstances
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">8</CAD:hasNumberCreatePointInstances>
          <CAD:hasNumberCreateCurveInstances
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">12</CAD:hasNumberCreateCurveInstances>
          <CAD:hasNumberCreateLinearCurveInstances
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">12</CAD:hasNumberCreateLinearCurveInstan
ces>
```

<CAD:hasNumberCreateEllipticalCurveInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberCreateEllipticalCurveInstances>
<CAD:hasNumberCreateHelicalCurveInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberCreateHelicalCurveInstances>
<CAD:hasNumberCreateInterpolatedCurveInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberCreateInterpolatedCurveInstances>
<CAD:hasNumberCreateSurfaceInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">6</CAD:hasNumberCreateSurfaceInstances>
<CAD:hasNumberCreatePlanarSurfaceInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">6</CAD:hasNumberCreatePlanarSurfaceInstances>
<CAD:hasNumberCreateConicalSurfaceInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberCreateConicalSurfaceInstances>
<CAD:hasNumberCreateSphericalSurfaceInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberCreateSphericalSurfaceInstances>
<CAD:hasNumberCreateSplineSurfaceInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberCreateSplineSurfaceInstances>
<CAD:hasNumberCreateToroidalSurfaceInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">0</CAD:hasNumberCreateToroidalSurfaceInstances>
<CAD:hasNumberCreateVertexInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">8</CAD:hasNumberCreateVertexInstances>
<CAD:hasNumberCreateEdgeInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">12</CAD:hasNumberCreateEdgeInstances>
<CAD:hasNumberCreateCoedgeInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">24</CAD:hasNumberCreateCoedgeInstances>
<CAD:hasNumberCreateLoopInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">6</CAD:hasNumberCreateLoopInstances>
<CAD:hasNumberCreateFaceInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">6</CAD:hasNumberCreateFaceInstances>
<CAD:hasNumberCreateShellInstances rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</CAD:hasNumberCreateShellInstances>
</CAD:Feature>
</CAD:hasFeature>
</CAD:Partfile>
</rdf:RDF>

REFERENCES

1 Shah, J.J., Mäntylä, M., 1995. Parametric and Feature-Based CAD/CAM. Wiley-Interscience, 1995. ISBN-10: 0471002143

2 Braid, I.C. 1979. "Notes on a Geometric Modeler". CAD Group Document 101, Cambridge University Computer Laboratory

3 Requicha, A. and Voelcker, H. 1977. "Constructive Solid Geometry" Tech. Memo. 25, Production Automation Project, Univ. Rochester, N.Y.

4 Requicha, A. 1980. "Representations of Rigid Solids: Theory, Methods, and Systems". ACM Computing Surveys. Vol. 12. Issue 4

5 Shah, J. and Rogers, M. 1988 "Functional Requirements and Conceptual design of the Feature-Based Modeling System" Computer Aided Engineering Vol. 5 Issue 1 pp. 9-15

6 Shah, J.J. 1991 " Assessment of Features Technology". Computer Aided Design Vol. 23 Issue 5

7 Luby, S.C., Dixon J.R. and Simmons M.K. 1986 "Creating and Using a Features data base". Computers in Mechanical Engineering Vol. 5 Issue 3

8 Cunningham, J. and Dixon, J. 1988 "Designing with Features: The Origin of Features". Proc. ASME Computers in Engineering Conf. San Francisco (Jul/Aug 1988)

9 Kyprianou, L. 1980. "Shape Classification in Computer Aided Design". Ph.D. dissertation, University of Cambridge.

10 Shah J.J., Anderson D., Kim, Y., and Joshi, S. 2001 "A Discourse on Geometric Feature Recognition From CAD Models". Journal of Computing and Information Science in Engineering Vol. 1 Issue 1, 41 (2001), DOI:10.1115/1.1345522

11 Babic B., Nesic N., and Miljkovic Z., 2008 "A review of automated feature recognition with rule-based pattern recognition". Computers in Industry Vol. 59 Issue 4, 321-337

12 Henderson, M., and Anderson, D., 1984, ''Computer Recognition and Extraction of Form Features: A CAD/CAM link,'' Comput. Ind., 5, pp. 329–339.

13 Probhakar, S., and Henderson, M. 1992. “Automatic Form-Feature Recognition using Neural-Network-Based techniques on Boundary Representations of Solid Models”. Coputer-Aided Design 24 (7): 381-393

14 Shah, J.J. 1988 “Feature Transformations between Application-Specific Feature Spaces”. Computer-Aided Engineering Vol. 5 Issue 6 pp. 247-255

15 Bettig, B. and Shah, J.J. 2001 “Derivation of a Standard Set of Geometric Constraints for Parametric Modeling and Data Exchange”. Computer Aided Design 2001. 33 pp 17-33

16 Ovtcharoca, J., Pahl, G. and Rix, J., 1992 “A Proposal for Feature Classification in Feature-Based Design”. Computers & Graphics Vol. 16 Issue 2, pp 187-195

17 Rosen, D. W. and Dixon, J. R., 1992. "Languages for Feature-Based Design and Manufacturability Evaluation," International Journal of Systems Automation: Research and Applications, 2(4), pp. 353-373.

18 Rosen, D. W. and Peters, T. J., 1992 "Topological Properties that Model Feature-Based Representation Conversions within Concurrent Engineering," Research in Engineering Design, 4(3), pp. 147-158.

19 Bettig, B., Shah, J., and Summers J., 2000 “Domain Independent Characterization of Parametric and Geometric Problems in Embodiment Design”. Proc. ASME Design Automation Conference, Baltimore, September

20 Bettig, B., Summers J., and Shah, J., 2000 “Geometric Exemplars: A Bridge between CAD and AI”. IFIP Conference of Knowledge Intensive CAD, KIC-4 Parma, Itally, May.

21 Chen, X., and Hoffmann, C. 1995. “On Editability of Feature-based Design”. Comp.-Aided Des., 27(12), pp. 905–914.

22 Hoffmann, C., 1997. “CAD Systems Development”. Springer, Berlin, ch. EREP Project Overview, pp. 32–40.

23 Shih, C., and Anderson, B., 1997. “A Design/Constraint Model to Capture Design Intent”. Proc. 4th ACM Symp. on Solid Modeling & Applications, 27(12), pp. 255–264.

24 Kim, J., Pratt, M. J., Iyer, R. G., and Sriram, R. D., 2008.“Standardized data exchange of CAD models with design intent”. Comp.-Aided Des., 40(7), pp. 760–777.

25 Patil L, Dutta D, Sriram R., 2005. “Ontology-based exchange of product data semantics”. IEEE J Automation Sci Eng 2005;2:213–25.

26 Rappoport A. 2003 “An architecture for universal CAD data exchange.” In: Proc. 2003 ACM solid modeling symposium. New York: ACM Press; p. 266–269.

27 Rappoport, A., Spitz, S., and Etzion, M., 2006. Lecture Notes in Computer Science. Springer,Berlin, ch. Two-Dimensional Selections for Feature-Based Data Exchange, pp. 325–342.

28 Choi, G. H.; Mun, D.; Han, S., 2002, “Exchange of CAD part models based on macro-parametric approach”, International Journal of CAD/CAM, 2(1), pp 13-21.

29 Mun D, Han S, Kim J, Oh Y., 2003. “A set of standard modeling commands for the history-based parametric approach.” Comput Aided Des 35:1171–9.

30 Seo T-S, Lee Y, Cheon S-U, Han S, Patil L, Dutta D., 2005. “Sharing CAD models based on feature ontology of commands history”. Int J CAD/CAM 2005;5.

31 Song, I. and Han, S., 2010. “Parametric CAD Data Exchange Using Geometry-Based Neutral Macro File”. Cooperative Design, Visualization, and Engineering. Springer Berlin / Heidelberg, pp. 145-152

32 Li, M., Gao, S., Wang, C.C.L., 2007. “Real-Time collaborative Design With Heterogeneous CAD Systems Based on Neutral Modeling Commands,” ASME J. Comp. Inf. Sci. Eng., 7, pp. 113-125.

33 Cai, X., He, F., Li, X., and Chen, X., 2008. "A Direct Feature Retrieval and Reuse Method for Heterogeneous Collaborative CAD Systems," Pervasive Computing and Applications, 2008. ICPCA 2008. Third International Conference on , vol.2, no., pp.718-723

34 Li, X., He, F., Cai, X., Chen, Y., and Liu, H., 2009. "Using procedure recovery approach to exchange feature-based data among heterogeneous CAD systems," Computer Supported Cooperative Work in Design, 2009. CSCWD 2009. 13th International Conference on , vol., no., pp.716-721

35 Li, X., Cai, X., He, F., and Huang, Z., 2010. "Retrieval and reconstruction of heterogeneous feature data for collaborative design," Computer Supported Cooperative Work in Design (CSCWD), 2010 14th International Conference on , vol., no., pp.553-558, 14-16 April 2010

36 Zhang, Y. and Luo, X., 2009 "Design Intent Information Exchange of Feature-Based CAD Models," Computer Science and Information Engineering, 2009 WRI World Congress on Computer Science and Information Engineering, Vol.3, pp.11-15

37 Sun, W., Ma, T. Q., and Huang, Y. J., 2010. "Research on method of constraint conversion in feature-based data exchange between heterogeneous CAD systems". Journal of Mechanical Science and Technology 23(1) pp. 246-253.

38 Dartigues, C., Ghodous, P., Gruninger, M., Pallez, D., and Sriram, R., 2007, "CAD/CAPP Integration using Feature Ontology." Concurrent Engineering, 15(2):237-249.

39 Zhan, P., Jayaram, U., Kim, O., and Zhu, L., 2010, "Knowledge Representation and Ontology Mapping Methods for Product Data in Engineering Applications". Journal of Computing and Information Science in Engineering, 10(2):021004.

40 Zhu, L., Jayaram, U., Jayaram, S., and Kim, O., 2009, "Ontology-Driven Integration of CAD/CAE Applications: 10 Copyright © 2011 by ASME Strategies and Comparisons". ASME Conference Proceedings, 2009(48999):1461-1472.

41 Kim, O., Jayaram, U., Jayaram, S., and Zhu, L., 2009, "An Ontology Mapping Application Using a Shared Ontology Approach and a Bridge Ontology". ASME Conference Proceedings, 2009(48999):431-441.

42 Hanayneh, L., Wang, Y., Wang, Y., Wileden, J., and Qureshi, K., 2008. "Feature Mapping Automation for CAD Data Exchange". 2008 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE2008) (DETC2008-49671).

43 Altidor, J., Wileden, J., Wang, Y., Hanayneh, L., and Wang, Y., 2009. "Analyzing and Implementing a Feature Mapping Approach to CAD System Interoperability". 2009 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE2009)(DETC2009-87539).

44 Stroud, I. 2006. Boundary Representation Modelling Techniques. London: Springer, 2006, p. 31 ISBN-10: 0-387-84628-312-4

45 Spacial Corp, 2010. "Tutorial:ACIS Tutorials (Topology)" Available online at http://doc.spatial.com/index.php/Tutorial:ACIS_Tutorials_(Topology)

46 Spacial Corp, 2010. "Tutorial:ACIS Tutorials (Geometry)" Available online at http://doc.spatial.com/index.php/Tutorial:ACIS_Tutorials_(Geometry)

47 Braid, I. C., Hillyard, R. C., Stroud, I. A., “Stepwise construction of polyhedral in geometric modelling”, in “Mathematical Methods in Computer Graphics and Design”, ed. K. W. Brodlie, Academic Press, 1980.

48 Wache, H., Vogele, T.J., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hubner, S., 2001, 11 Copyright © 2009 by ASME “Ontology-based integration of information – a survey of existing approaches”, Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing, Seattle, WA.

49 W3C, 2004, "OWL Web Ontology Language Overview". Available online at http://www.w3.org/TR/owl-features

50 W3C, 2004, “Resource Description Framework (RDF)”. Available online at http://www.w3.org/RDF/

51 W3C, 2004, “RDF Vocabulary Description Language 1.0: RDF Schema”. Available online at http://www.w3.org/TR/rdf-schema/.

52 Horrocks, I., Patel-Schneider, P., and Harmelen, F. 2003 "From SHIQ and RDF to OWL: the making of a Web Ontology Language". Web Semantics: Science, Services and Agents on the World Wide Web Vol. 1, Issue 1, December 2003, Pages 7-26

53 Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., 2002. The Description Logic Handbook. Cambridge University Press, Cambridge, London.

54 Horrocks, I., Sattler, U., and Tobies, S. 2000. “Practical reasoning for very expressive Description Logics”. J. Interest Group Pure Appl. Logic 8 (3) (2000) 239–264.

55 Horrocks, I., Sattler, U., and Tobies, S. 1999 “Practical reasoning for expressive Description Logics”, in: H. Ganzinger, D. McAllester, A. Voronkov (Eds.), Proceedings of the Sixth International Conference on Logic for Programming and Automated Reasoning (LPAR’99), Lecture Notes in Artificial Intelligence, vol. 1705, Springer, Berlin, 1999, pp. 161–180.

56 Horrocks, I., and Sattler, U. 2001. “Ontology reasoning in the SHOQ(**D**) Description Logic”, in: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI’2001), Seattle, Washington, USA, August 4–10, 2001, pp. 199–204.

57 Horrocks, I. Patel-Schneider, P. Boley, H., Tabet, Grosof, B. and Dean, M., 2004, “SWRL: A Semantic Web Rule Language Combining OWL and RuleML” Available online at http://www.w3.org/Submission/SWRL/

58 Grimm, S., Motik, B., and Preist, C. 2006. “Matching Semantic Service Descriptions with Local Closed-World Reasoning”. Lecture Notes in Computer Science, Volume 4011

59 Grimm, S. and Hitzler, P. 2008. “Semantic Matchmaking of Web Resources with Local Closed-World Reasoning” International Journal of Electronic Commerce. Vol. 12 Issue 2 pp. 89-126

60 Katz, Y. and Parsia, B., 2005 “Towards a nonmonotonic extension to OWL,” Proc. Workshop on OWL Experiences and Directions, 2005.

61 Hosain, S., Jamil, H. 2009. “Empowering OWL with overriding inheritance, conflict resolution and non-monotonic reasoning”. In AAAI Spring Symposium on Social Semantic Web: Where Web 2.0 meets Web 3.0 (2009)

62 Knorr, M., Alferes, J., and Hitzler, P. 2011 “Local closed world reasoning with description logics under the well-founded semantics”. Artificial Intelligence, Vol. 175, Issues 9-10, June 2011, Pages 1528-1554

63 Stanford Center for Biomedical Informatics, 2010, “What is Protégé-OWL?” Available online at http://protege.stanford.edu/overview/protege-owl.html

64 Friedman-Hill, E., 2008 “Jess, the Rule Engine for the Java Platform”. Sandia National Laboratories. Available online at http://www.jessrules.com